Pattern Recognition Letters 49 (2014) 99–106

Contents lists available at ScienceDirect

Pattern Recognition Letters

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

Efficient classification using the Euler characteristic ☆

Eitan Richardson *, Michael Werman

School of Computer Science, The Hebrew University of Jerusalem, Jerusalem 91904, Israel

ARTICLE INFO

Article history:
Received 19 January 2014
Available online 11 July 2014

Keywords:
Euler
Topology
Classification
Image
3D

ABSTRACT

This paper introduces an object descriptor for classification based on the Euler characteristic of subsets created by thresholding a function at multiple levels (sub-level filtration). We demonstrate the effectiveness of this basic topological invariant of sets, the Euler characteristic, and use it to compute descriptors in two different domains – images and 3D mesh surfaces. The descriptors used as input to linear SVMs achieve state of the art classification results on various public data sets. Moreover, these descriptors are extremely fast to compute. We present linear time methods to calculate the Euler characteristic for multiple threshold values and to compute the Euler characteristic in a sliding window.

1. Introduction

Supervised object classification entails two main elements – features (descriptors) and learning algorithms. In recent years, much effort has been invested in developing features that yield good classification. Different features are developed for different domains such as images and 3D objects. Some features are even object specific, e.g. faces or texture.

For good classification, features should be rich, descriptive and discriminative, and on the other hand, invariant to different transformations and robust enough to allow intra-class variation. The focus in recent years shifted from *global features* that describe the object as a whole, to statistical descriptors of *local features*. The statistical descriptor of low-dimensional local features is simply their distribution (e.g. color histogram). For more complex local features (e.g. SIFT), a *Bag-of-Words* (BoW) scheme is used.

The main criticism for statistical descriptors of local features is the loss of all spatial information: "because these methods disregard all information about the spatial layout of the features, they have severely limited descriptive ability" [14]. For example, in one object class, the different values of the local feature might be evenly distributed, and in another class, the different values are clustered. Several approaches for putting the local features into some spatial (or spatio-temporal) context have been suggested. Some examples are spatial pyramids [14] and hierarchical neighborhood features [12].

This paper presents a new descriptor for supervised classification, which is based on simple local features, but instead of using their distribution, we propose to threshold the feature at multiple levels and calculate the Euler characteristic (EC) values of the resulting subsets of the domain. The vector of Euler Numbers – the Euler Characteristic Graph (or EC Graph) is then used as an object descriptor.

The EC is a global topological invariant which in the case of a two-dimensional set is the number of connected components minus the number of holes. The EC Graph feature therefore encodes information about the spatial distribution of the local property, information which is missing in many statistical descriptors. One of the nice algorithmic properties of EC Graph is that it can be easily computed by counting local elements. The EC is invariant to all topological transformations, including rotation and scale. Section 6.2.2 shows that the EC Graph descriptor has better invariance to image transformations than the global distribution descriptor [21].

We evaluate the performance of the EC Graph descriptor using images and 3D mesh objects, based on different local properties. In both domains we use publicly available object classification datasets and show that the EC Graph feature achieves state of the art results at very low computation time.

The rest of the paper is organized as follows: In Section 2 we review existing methods for calculating the EC and for its use as a descriptor. The EC Graph descriptor is introduced in Section 3. Sections 4 and 5 present our efficient methods for calculating the EC Graph on the entire domain and in a sliding window. In Section 6 we provide experimental results for the efficient calculation methods and evaluate the performance of the EC Graph feature for classification. We provide our conclusions in Section 7.

☆ This paper has been recommended for acceptance by A. Shokoufandeh.

* Corresponding author. Tel.: +972 54 443 1168; fax: +972 8 971 9268.
E-mail address: eitan.richardson@gmail.com (E. Richardson).

http://dx.doi.org/10.1016/j.patrec.2014.07.001
0167-8655/

The contributions of this paper:

- EC Graph – A new descriptor for supervised classification based on the Euler characteristic of simple local features.
- An algorithm for efficient calculation of the EC Graph for multiple thresholds, for regular and non-regular domains.
- An algorithm for efficient calculation of the EC Graph in a sliding window (can be used for object detection).
- Evaluation of the EC Graph feature for classification of images and 3D mesh objects.

2. Related work

2.1. The Euler characteristic

The Euler characteristic (or Euler number) dates back to Leonhard Euler (1707–1783) who observed that in simple polyhedra, $V - E + F = 2$ i.e. the number of vertices minus the number of edges plus the number of faces always equals 2. This was later generalized to the topological invariant: $\chi = V - E + F$, for any object constructed from 0, 1 and 2-dimensional cells (vertices, edges and faces). In general, the EC χ of a set S is equal to the alternating sum of the cardinalities of the open k-dimensional cells in any partition of S.

It is important to note that the Euler characteristic of an object is independent of the cell-decomposition (also termed *triangulation*) being used. An example for a cell-decomposition is a uniform square lattice, in which a 2D planar set (e.g. an image) is constructed of open squares (pixels), their edges and corners. Another example is a triangle-mesh representing a surface in 3D, which is constructed from triangular faces, edges and vertices. In addition, it should be noted that the Euler characteristic is additive, obeying the inclusion–exclusion principle.

2.2. The Euler characteristic as an object descriptor

Several papers suggest using the Euler characteristic of a binary image for recognition or classification. For example, Anagnostopoulos et al. [2] use the EC of a binary image of a license plate for OCR. Mery et al. [16] suggest using the EC of corn tortillas to evaluate their quality (a high-quality tortilla is expected to have one connected component and no holes). In both papers, a single Euler number is calculated for a binary image that was created by segmenting the original image.

The concept of calculating the EC of subsets of the domain defined by multiple threshold values of a density function over the domain is suggested by Worsley [20]. The resulting graph is manually compared to the expected EC graph of a proposed stochastic model [1].

Huber et al. [10,22] use the Minkowski Functionals (MFs) of grayscale images at multiple threshold levels for classification. In both cases, the EC is calculated on the original grayscale image. We propose a more general descriptor, which is based on the EC graph of different local features instead of using the original image. We will show that this approach yields much better results and is applicable to different domains and not just to images.

2.3. Calculating the Euler characteristic

The Euler characteristic of a binary image on a continuous 2D plane and on a lattice (a discretized image) is defined by Gray [8]. Over the years, several methods were suggested for efficient calculation of the EC of a binary image (an image with pixel values of '0' and '1'). Gray [8] proposes a calculation method based on counting 2×2 pixel patterns (in case of a square lattice). Bribiesca [3] proposes a method based on the Contact Perimeter (length of edges adjacent to two pixels).

For grayscale images, simply repeating the EC calculation for multiple threshold levels will result in $O(NT)$ time complexity, where N is the number of pixels in the image and T is the number of threshold values. Snidaro and Foresti [19] and Conaire [5] propose a method that operates at $O(N + T)$, but since it is based on Gray's 2×2 pixel pattern, it is applicable to regular grids only. The method we propose has a time complexity of $O(N + T)$ as well, and is applicable also to non-regular domains, such as 3D mesh surfaces.

The Minkowski Functionals (which include the EC) are suggested as a feature for texture analysis, an application that requires calculating the MFs in multiple image sub-windows [15]. The authors describe a bias in the MFs calculation and propose an approximated solution by averaging MF values calculated using 8 different traversing directions. In Section 5 we propose a closed and efficient solution for calculating the EC in multiple sub-windows. Our method is based on triangulation (counting vertices, edges and faces) and on a modified Integral Image [6] calculation.

3. The Euler Characteristic Graph feature

Fig. 1 demonstrates the EC Graph feature for a two-dimensional image. A more formal definition is provided in the next section. The input image shown in Fig. 1(a) was generated by applying a Gaussian filter on a random image. Fig. 1(b)–(f) show how the original object is segmented by thresholding the input. As we increase the threshold, holes (shown in black) begin to appear and χ becomes negative. When we continue raising the threshold, the segmented object begins breaking up into components. χ will reach 0 again when the number of components is equal to the number of holes (Fig. 1(d)). χ continues to rise as we increase the threshold, until the object consists of many separate 'islands'. As we continue increasing the threshold, the islands will start to disappear and χ will move towards 0 again. Fig. 1(g) shows the resulting EC Graph.

4. Efficient calculation of the Euler Characteristic Graph

A set $X \subset \mathbb{R}^n$ can be represented, non uniquely, as a union of open k-cells (open elements of dimension $k, k \leqslant n$):

$$X = \bigcup_{k=0}^{n} \bigcup_{i=1}^{M(k)} x_i^{(k)} \tag{1}$$

where $x_i^{(k)}$ is an open cell of dimension k and $M(k)$ is the number of k-cells. The Euler characteristic χ of the set X is:

$$\chi = \sum_{k=0}^{n} (-1)^k M(k) \tag{2}$$

In some cases, as in most applications discussed in this paper, we need to find the EC of a subset S of the original domain, defined by a function over the domain. S can be represented as a union of open k-cells, as defined by (1). In this section we assume that S itself is closed.

We first discuss the case of a binary function: Let $f_d(x^{(d)}) \in \{0, 1\}$ be a binary function defined over the open cells of the highest dimension d of which X is constructed. For example, if X is a mesh surface in $\mathbb{R}^3$, f will be defined over the triangular faces ($d = 2$). The functions over the lower-dimension elements are not given as an input and are derived from f_d. We define $f_k(x^{(k)}), k < d$ to be 1 for all k-cells on the boundary of "1" d-cells, in which $f_d(x^{(d)}) = 1$:

$$f_k(x^{(k)}) = \mathbf{1}_{[\exists x^{(d)} \in N_d(x^{(k)}) : f_d(x^{(d)}) = 1]} \tag{3}$$

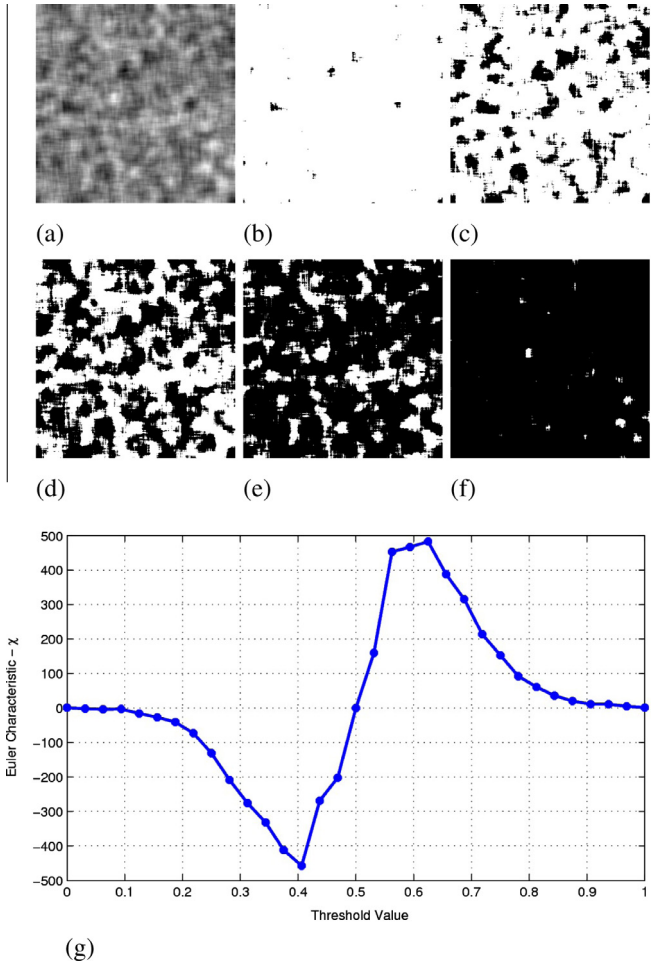

Fig. 1. The EC Graph (a) the input image (b) $t = 0.2$ (c) $t = 0.4$ (d) $t = 0.5$ (e) $t = 0.6$ (f) $t = 0.8$ (g) the EC Graph for 32 threshold values.

where $N_d(x^{(k)})$ are all d-cells in the immediate neighborhood of $x^{(k)}$ (i.e. $x^{(k)}$ is in the closure of $x^{(d)}$). See Fig. 2(a). If we define $M_f(k)$ to be the number of k-cells satisfying $f_k(x^{(k)}) = 1$, the EC of the subset defined by the function f (or f_d) is:

$$\chi_f = \sum_{k=0}^{n} (-1)^k M_f(k) \tag{4}$$

For two-dimensional binary images, the input function f or f_2 is the binary image itself, defined over the 2-cells (pixels). 1-cells (pixel edges) and 0-cells (pixel corners) are assigned the value of 1 if they border a "1" pixel. We calculate the EC of the binary image by counting the "1" corners minus the "1" edges plus the "1" pixels.

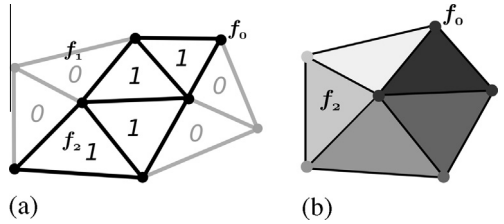

Fig. 2. Function over triangulation cells (a) edges and vertices on the boundary of faces with $f_2 = 1$, have $f_1 = 1$ and $f_0 = 1$ respectively (shown in darker color) (b) f_0 value of the vertices are defined as the max f_2 values of their neighboring faces (darker gray = higher value). f_1 values are not shown.

Now consider the case of a non-binary function: $f_d(x^{(d)}) \in [0 \dots 1]$. We would like to calculate $\chi_f(t)$, the Euler characteristic of the sub-set defined by $f(x) \geqslant t$ for multiple threshold values: $t \in \{t_1, \dots, t_T\}$. One option of course is to define a binary indicator function for each threshold:

$$\hat{f}_d^t(x^{(d)}) = \mathbf{1}_{[f_d(x^{(d)}) \geqslant t]} \tag{5}$$

and calculate the EC for each binary function as described above. The time complexity of such a method is $O(NT)$, where N is the number of cells and T the number of threshold values.

We describe an alternative method for calculating $\chi_f(t)$ in $O(N+T)$. Unlike the methods proposed by Snidaro and Foresti [19] and Conaire [5], our method uses the basic triangulation and is therefore applicable to any domain. Given $f_d(x^{(d)}) \in [0 \dots 1]$, we define $f_k(x^{(k)}), k < d$ to be:

$$f_k\left(x_i^{(k)}\right) = \max_{j \in N_d\left(x_i^{(k)}\right)} f_d\left(x_j^{(d)}\right) \tag{6}$$

where $N_d\left(x_i^{(k)}\right)$ are all d-cells in the immediate neighborhood of $x_i^{(k)}$ (i.e. $x_i^{(k)}$ is on their boundary). See Fig. 2(b). Define:

$$M_f^t(k) = \mid \{x^{(k)}, f_k(x^{(k)}) \geqslant t\} \mid \tag{7}$$

$M_f^t(k)$ counts the number of k-cells with a value f_k greater or equal to the threshold t. The EC for the subset defined by the threshold t of the function f can be calculated using:

$$\chi_f(t) = \sum_{k=0}^{n} (-1)^k M_f^t(k) \tag{8}$$

The above holds because the closed subset S defined by threshold t contains all open d-cells with $f_d(x^{(d)}) \geqslant t$ and all k-cells ($k < d$) on their boundary. A k-cell will therefore be part of the subset if any of its neighboring d-cells are in the subset, which will be true if $f_d(x^{(d)}) \geqslant t$. The assignment in (6) ensures that.

To calculate $M_f^{t_i}(k)$ for multiple threshold values defined by the monotonic series $(t_1, \dots, t_T)$, we first calculate:

$$\hat{M}_f^{t_i}(k) = \left|\left\{x^{(k)}, t_i \leqslant f_k(x^{(k)}) < t_{i+1}\right\}\right| \tag{9}$$

$\hat{M}_f^{t_i}(k)$ counts the number of k-cells in each half-open interval between two consecutive threshold values and can be calculated for all threshold values using bucket-sort in time $O(N)$. Each cell is assigned to a single half-open interval, or histogram bin (denoted by b in Algorithm 1). We then compute $M_f^{t_i}(k)$ by summing over $\hat{M}_f^{t_i}(k)$ values:

$$M_f^{t_i}(k) = \sum_{j=i}^{T} \hat{M}_f^{t_j}(k) \tag{10}$$

Our method is based on the fact that the highest dimension open cells, for which the function f_d is defined, determine which lower-dimension cells will be part of the subset S. Given a function f_d and a threshold t, d-cells for which f_d is above t will be part of the subset S and all lower-dimension cells on their closure. A k-cell will therefore be in S according to $max(f_d)$ of the d-cells it borders. Another key idea in our method is the ability, for a monotonic series of threshold values $\{t_1, \dots, t_T\}$, to count cells above a threshold t by first counting cells belonging to each interval $[t_i \dots t_{i+1})$ (histogram bins) and then integrating over the intervals.

See Algorithm 1 for the pseudo-code of the proposed method.

Algorithm 1. Calculate $\chi_f(t)$ for $t \in (t_1, \ldots, t_T)$

Require f_d,$\{t_1, \ldots, t_T\}$
$\chi \leftarrow 0$
//Go through the cell dimensions...
for $k = 0 \rightarrow d$ **do**
  //Calculate the function f_k and its histogram $\hat{M}_f$
  $\hat{M}_f \leftarrow 0$
  for $i = 1 \rightarrow M(k)$ **do**
    $f_k \leftarrow \max_{j \in N_d(x_i^{(k)})} f_d(x_j^{(d)})$
    $b \leftarrow bin(f_k)$
    $\hat{M}_f(b) \leftarrow \hat{M}_f(b) + 1$
  end for
  //Accumulate the histogram and update χ
  $c \leftarrow 0$
  for $i = T \rightarrow 1$ **do**
    $c \leftarrow c + \hat{M}_f(i)$
    $\chi(i) \leftarrow \chi(i) + (-1)^k * c$
  end for
end for
return χ

5. Efficient calculation of the Euler characteristic in a sliding window

For object detection it is sometimes required to calculate a feature in a sliding window over the domain. For example, in face detection, we first learn a model for faces and then test the model in a sequence of overlapping windows. Calculating the EC Graph in each window will result in a total time complexity of $O(MW)$, where M is the number of windows and W is the window size. In this section we propose an efficient method for calculating the EC Graph in a sliding window. The method is based on the concept of Integral Images [6], however, adopting this concept for the EC is not trivial and requires special treatment of the boundaries.

The Integral Image is an efficient method for calculating integrals (sums of pixel values) over rectangular windows in constant time. The integral over the window W in Fig. 3 is calculated using the pre-calculated integral values of its four corners (sums over the rectangles to the bottom-left of each corner):

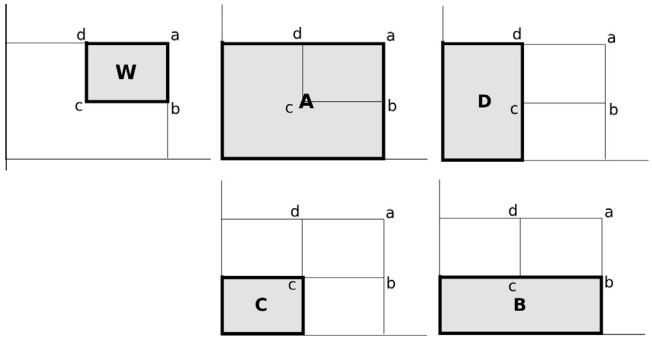

Fig. 3. Integral Image – integral of a function over the window is calculated using the pre-calculated integrals over the four rectangles to the bottom-left of the four window corners. Throughout this section, corners are denoted by small letters and the window to the bottom-left of each corner is denoted by a capital letter.

$$S_W = S_A - S_D - S_B + S_C \tag{11}$$

The above equation is a direct result of the inclusion–exclusion principle. Since the EC is a sum of local properties and the inclusion–exclusion principle is valid for it, we might think that it is possible to calculate the EC value of a window in the same manner as the Integral Image above. This is not correct, as demonstrated in Fig. 4(a): We expect the EC value in the window (W) to be 1, since it contains one component and no holes, however:

$$\chi(A) - \chi(D) - \chi(B) + \chi(C) = 1 - 1 - 0 + 0 = 0 \neq 1$$

The reason for this error is that, as we saw in the previous section, the EC calculation includes one and zero-dimension open cells (edges, vertices). These cells, unlike the 2D cells (pixels) that are counted in the regular Integral Image, exist on the boundary of the window. This breaks the assumption that $W \cap B = W \cap D = \emptyset$, which is made when applying in the inclusion–exclusion principle. To correctly adopt the Integral Image concept to the EC, we need to handle the window boundaries.

Considering the closed window in Fig. 4(b), we expect to get $\chi(W) = 4$. Note that the EC value includes the shape on the right, which is tangent to the window and contains some vertices and edges on the closed border of the window. This calculation is correct even though the shape on the right does not contain any pixels (faces) inside the window.

Our proposed solution for calculating the EC in a window is based on pre-calculating the EC for different closed and half-open sets. Consider the four sets A, B, C, D defined around the window W in Fig. 5. We can state the following:

$$W \cap B = W \cap D = \emptyset, \quad D \cap B = C, \quad W \cup D \cup B = A$$

And applying the inclusion–exclusion principle, we now get:

$$\chi(W \cup D \cup B) = \chi(W) + \chi(D) + \chi(B) - \chi(W \cap D) - \chi(W \cap B) - \chi(D \cap B) + \chi(W \cap D \cap B)$$

$$\chi(W) = \chi(A) - \chi(D) - \chi(B) + \chi(C) \tag{12}$$

To calculate the EC in any window, we therefore need to pre-calculate four different EC values for each point (u, v) in the image, for the four rectangles $\chi_1 : \{x \leqslant u, y \leqslant v\}$, $\chi_2 : \{x < u, y \leqslant v\}$, $\chi_3 : \{x \leqslant u, y < v\}$, $\chi_4 : \{x < u, y < v\}$. If the four corners of the window (clockwise from top-right) are a, b, c, d, then the EC of the window will be:

$$\chi(W) = \chi_1(a) - \chi_2(d) - \chi_3(b) + \chi_4(c) \tag{13}$$

An example for detection using the EC Graph in a sliding window is provided in Section 6.2.2.

6. Experimental results

6.1. Euler characteristic computation time

In Section 4 we presented a method for efficient calculation of the EC Graph – Euler characteristic of subsets of an object defined by thresholding a function over the object at different levels. The method is generic, applicable to any domain and enables calculating the EC for multiple threshold values in time $O(N + T)$ instead of $O(NT)$. Table 1 shows the EC calculation time for 32 and 256 threshold values. The left column shows the time of a naive method that repeats the EC calculation for each threshold value. The calculation time for our method is shown in the right column. In both cases, the time shown is for calculating the multiple EC values only, not including the calculation time of the underlying local property. We measured the calculation time for a 1 MP grayscale image and for a 3D triangular mesh with 250 K faces. As can be seen in Table 1, our proposed method scales well with the number

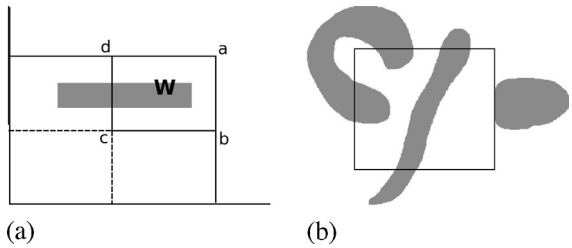

Fig. 4. (a) Wrong result when applying the Integral Image formula directly to EC (b) an example of configuration of shapes relative to a window.

of threshold values T, while a naive calculation that repeats the EC calculation for every threshold level is linear in T. The measurements were taken on a desktop PC with an Intel E6600 CPU @3.06 GHz, using our Matlab implementation.

6.2. Evaluating the EC Graph feature for classification

This section discusses how the EC Graph can be used as a descriptor for object classification in different domains – mesh surfaces and images. In both cases, the Support Vector Machine (SVM) was used as the supervised classification algorithm. We used the default Matlab SVM with a linear kernel and Sequential Minimal Optimization [18]. The pairwise classification accuracy was measured using random k-fold validation (with $k = 4$ or $k = 8$, depending on the size of the dataset). Each test was repeated 100 times.

6.2.1. EC Graph feature for 3D objects classification

A 3D triangle-mesh object X is defined by a set of vertices $V \subset \mathbb{R}^3$ and a set of faces $F \subset \mathbb{Z}^3$. Each face is defined by three vertices. The mesh also implicitly defines edges $E \subset \mathbb{Z}^2$, which are the borders between adjacent faces. The 3D object is therefore constructed of 0-cells (vertices), 1-cells (edges) and 2-cells (faces). Following the definition of Section 4, we calculate the EC Graph for a function f_2 defined over the faces at multiple threshold values. Note that if $t_{min} = \min_{x^{(2)} \in X} f_2(x^{(2)})$, then $\chi(t_{min})$ will be the EC of the entire surface.

We tested the classification performance of different functions using two datasets of 3D objects: The TOSCA high-resolution dataset [4] contains 80 objects divided to 9 categories (e.g. cat, dog). Objects within each category differ by non-rigid transformations. See Fig. 6(a) for a sample from the 'cat' category. The second dataset we experimented with contains 248 3D-scanned prehistoric stone tools from two excavation sites – Qesem and Nahal Zihor [9]. See Fig. 7(a) for a sample. In both cases, we performed pairwise classification between all category pairs and calculated the classification accuracy.

In both datasets, a function that yielded high classification results was the maximum curvature. The two principal curvature values (k_1, k_2) of a point on a surface $S \subset \mathbb{R}^3$ are defined as the two eigenvalues of the Hessian matrix at that point. Geometrically, the maximum curvature k_1 is the amount by which the surface bends at each point. See Figs. 6(b) and 7(b) for a blue-red visualization of the maximum curvature values on the surface of the objects. Figs. 6(c) and 7(c) demonstrate how thresholding the curvature segments the original surfaces to several connected components and holes.

Another function that yielded high classification results is the Shape Index [11], which is a scale-invariant interpretation of the two principal curvature values defined as $s = \frac{2}{\pi} arctan \frac{k_2+k_1}{k_2-k_1}$.

Fig. 8 shows the EC Graphs of the Shape Index feature for instances from different classes of the TOSCA dataset. We calculated the EC Graph of the Shape Index function for 20 uniform threshold values in $[-1 \ldots 1]$.

The pairwise classification accuracy values are given in Table 2. As can be seen, the curvature-based EC Graph features provided the best results for the Tosca dataset and the second best for the prehistoric stone tools. The run time for the EC Graph features

Table 1
Computation time of the EC Graph feature for images and for 3D mesh objects.

Experiment	Naive calculation (sec)	Our method (sec)
1 MP image/32 threshold levels	4.6	0.19
1 MP image/256 threshold levels	37.1	0.29
250 K faces mesh/32 threshold levels	39.4	1.25
250 K faces mesh/256 threshold levels	314.6	1.25

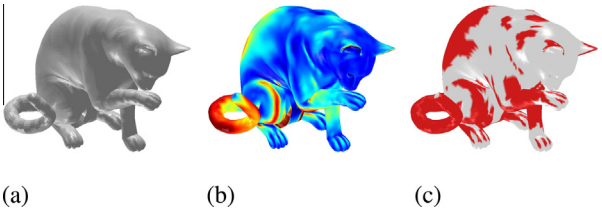

Fig. 6. A sample object from the TOSCA dataset (a) the original object (b) the maximum curvature function (c) curvature above a threshold of 0.05 (shown in red). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

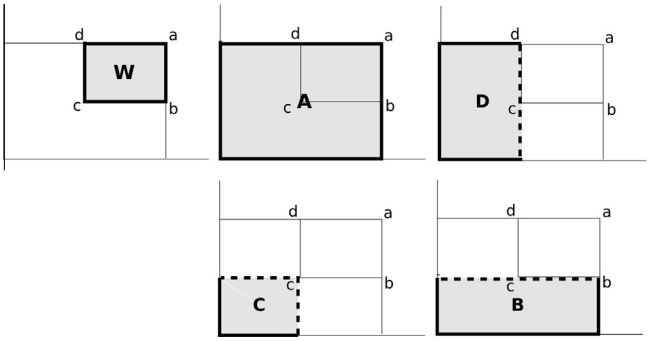

Fig. 5. Constructing a window using closed and half-open rectangular sets.

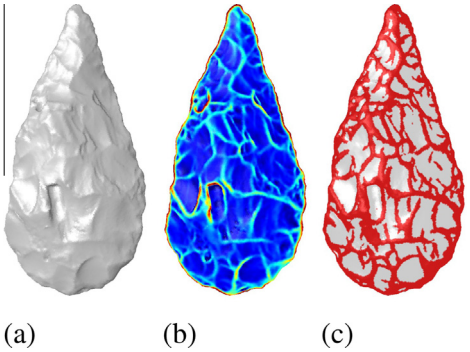

Fig. 7. A 3D-scanned stone tool from Nahal Zihor (a) the original object (b) the maximum curvature function (c) curvature above a threshold of 0.007.

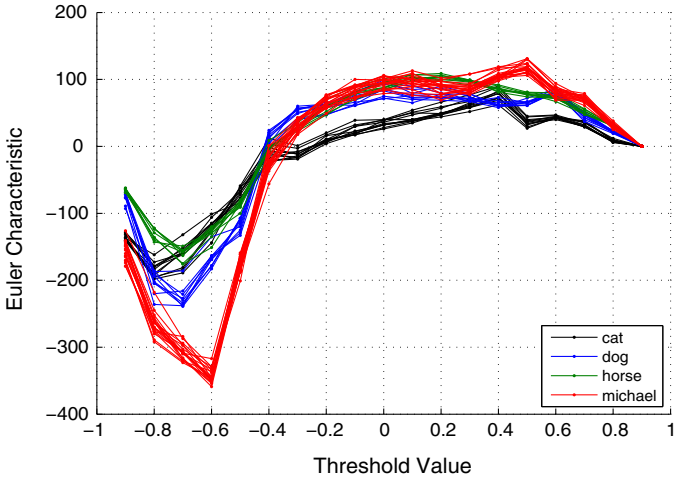

Fig. 8. The EC Graph feature of the Shape Index property for different objects in the TOSCA dataset.

Table 2
Classification accuracy for the TOSCA and Lithic datasets using different features. The best result in each category is marked in bold.

Feature	TOSCA (%)	Lithics (%)
EC Graph/curvature	98.6	93.7
EC Graph/Shape Index	**99.6**	96.8
EC Graph/distance from center	89.7	52.2
EC Graph/normal distance	95.2	70
Osada D2 [17]	86.9	65.2
SISI [7]	96.9	**97.4**
LD-SIFT [7]	96.2	91.4

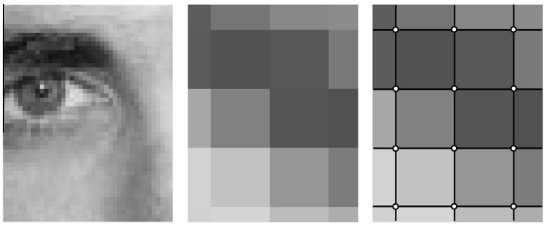

Fig. 9. 2D image (a) part of the original gray-scale image (b) zoom in on individual pixels (c) 0,1 and 2 dimension cells.

(including calculation of the underlying curvature function) is only a fraction (less than one tenth) of the run time for the state of the art BoW features – LD-SIFT and Scale-Invariant Spin Image (SISI) [7].

6.2.2. EC Graph feature for image classification

As discussed in Section 2, there are several ways to calculate the Euler characteristic of an image. We choose to keep using the cell-decomposition method, which constructs the image object from 0, 1 and 2-degree cells, that is, pixel-corners, pixel-edges and pixels. See Fig. 9 for an example.

For testing the performance of the EC Graph, we used UIUCTex [13], a dataset of natural textures containing 25 classes of textures with 40 samples in each class. See Fig. 10 for some examples. We compared two basic functions defined over the pixels – the gray-scale pixel intensity value and the gradient magnitude. Table 3 provides the pairwise classification accuracy of all texture classes.

Comparing the global distribution to the EC Graph: the first two rows in Table 3 list the success rates of the global distribution of the grayscale values and of the EC Graph of these values. As can be seen, the EC Graph provides a slightly better result (98.1% vs 97.2%).

To test the robustness of the features to image distortions, we applied a random illumination distortion to the texture images. The distortion consisted of an illumination offset: $\acute{I} = I + \alpha$, scale: $\acute{I} = (1+\beta)I$ and gamma change: $\acute{I} = I^{(1+\gamma)}$. The three parameters where chosen randomly for each image. An example of the illumination distortion can be seen in Fig. 11. As can be seen, the distortion is minor and typical to photographs taken at different lighting conditions. The classification accuracy for the distorted textures is given in the right column of Table 3. Comparing the performance of the EC Graph to that of the global distribution, the EC Graph results are significantly better (84.5% vs 67.7%), indicating that it is more robust to transformations that affect the underlying local property.

When using gradient magnitude as the local feature, the EC Graph classification accuracy for the original textures was slightly

Table 3
Pairwise classification accuracy for the original and distorted textures dataset using different features. The best result in each category is marked in bold.

Feature	Original (%)	Distorted (%)
Distribution/gray-level	97.2	67.7
EC Graph/gray-level	98.1	84.5
EC Graph/gradients	94.8	90.2
EC Graph/gray-level, Gradients	**98.9**	**91.8**

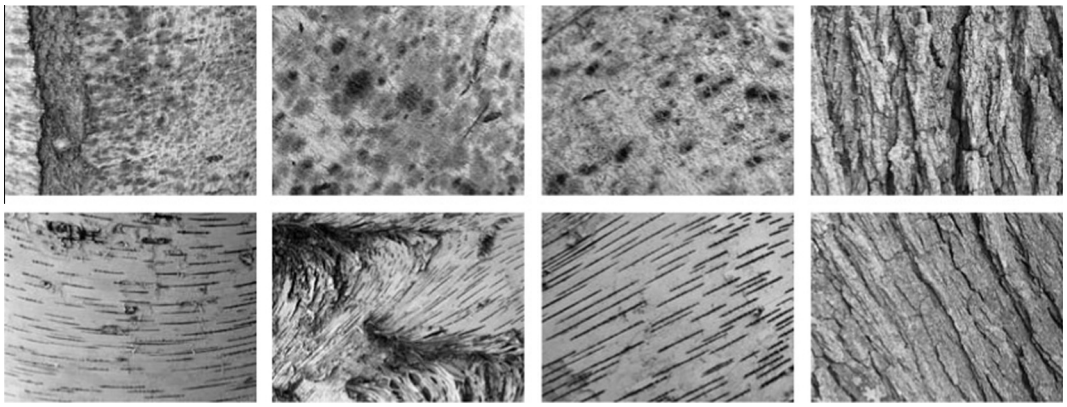

Fig. 10. Some examples from the natural textures dataset.

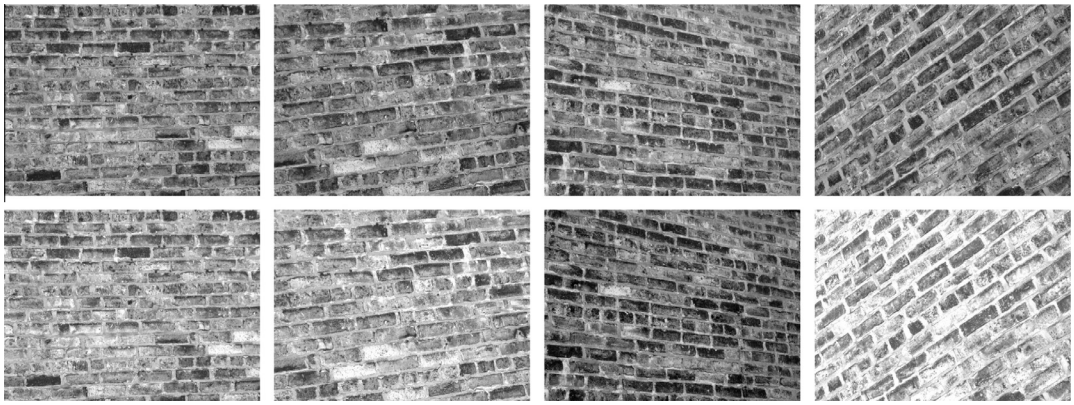

Fig. 11. Applying illumination distortion to the textures dataset. First four samples from the 'brick1' class (top row), same samples after applying random illumination distortions (bottom row).

Fig. 12. Detection of the 'bricks' texture in a sliding window. Intensity of red color indicates the positive detection probability. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

lower than that of the original gray values, however, the accuracy for the distorted textures was higher when using the gradients (90.2%) compared to the gray value (84.5%). The best results where achieved when concatenating the EC Graphs of the gray-level and the gradients to form a single descriptor – 98.9% and 91.8% for the original and distorted textures respectively. For all EC Graph features, 16 threshold values were used.

Fig. 12 shows an example for detection in a sliding window using the EC Graph descriptor. A classifier for brick texture was trained using all 'brick' texture examples from [13]. 60 random background images were used as negative examples. Two EC Graph descriptors were calculated for each 80×80 pixels window, one for the gray-level values and one for the gradient magnitudes. Each descriptor was calculated using 8 threshold values, and the two descriptors were concatenated to form one vector. The probability of each pixel to be part of a brick texture region was defined as the number of positive detection windows it was part of. The EC Graph descriptors in the detection phase were calculated using the efficient method described in Section 5 for EC Graph calculation in a sliding window.

7. Conclusions

We presented the EC Graph – a new descriptor for supervised object classification in various domains, which is based on the Euler characteristic. Using the fast computation method we presented, the EC Graph in any domain can be computed in time complexity of $O(N+T)$, which is the same as the time for computing the global distribution of the underlying local feature. The EC Graph yields surprisingly good results in the two domains we evaluated – images and 3D mesh objects, even with simple underlying local properties, possibly because of the information it encodes about their spatial distribution.

The choice of local feature for which the EC Graph descriptor is calculated affects the classification performance. We evaluated different local features for the different domains, for example, the image spatial gradients and the curvature for 3D surface objects. Evaluation of additional local features in conjunction with the EC Graph descriptor is encouraged. For the task of object detection, we have presented an efficient method for calculating the EC Graph feature in multiple windows (sliding window) over the domain.

Matlab code for the EC Graph descriptor calculation for images, and 3D mesh objects and for sliding window calculation is available at the author's web site.

Acknowledgments

The authors would like to thank Dr. Leore Grosman and Prof. Uzy Smilansky from the Computerized Archaeology lab at the Hebrew University for providing the 3D-scanned prehistoric artifacts.

References

[1] R. Adler, The Geometry of Random Fields, SIAM, 2009.
[2] C. Anagnostopoulos, I. Anagnostopoulos, V. Loumos, E. Kayafas, A license plate-recognition algorithm for intelligent transportation system applications, IEEE Trans. Intell. Transp. Syst. 7 (2006) 377–392.
[3] E. Bribiesca, Computation of the Euler number using the contact perimeter, Comput. Math. Appl. 60 (2010) 1364–1373.
[4] A. Bronstein, M. Bronstein, R. Kimmel, Numerical geometry of non-rigid shapes, Monographs in Computer Science, 2008.
[5] C. Conaire, Efficient Euler-Number Thresholding. Technical Report, 2003. URL <http://elm.eeng.dcu.ie/oconaire/papers/QuickEuler.pdf>.
[6] F. Crow, Summed-area tables for texture mapping, Comput. Graphics 18 (1984) 207–212.
[7] T. Darom, Y. Keller, Scale-invariant features for 3-d mesh models, IEEE Trans. Image Process. 21 (2012) 2758–2769.
[8] S. Gray, Local properties of binary images in two dimensions, IEEE Trans. Comput. 100 (1971) 551–561.
[9] L. Grosman, O. Smikt, U. Smilansky, On the application of 3-d scanning technology for the documentation and typology of lithic artifacts, J. Archaeol. Sci. 35 (2008) 3101–3110.
[10] M. Huber, M. Nagarajan, G. Leinsinger, L. Ray, A. Wismüller, Classification of interstitial lung disease patterns with topological texture features. <arXiv:1005.5086>, 2010, preprint.
[11] J. Koenderink, A. van Doorn, Surface shape and curvature scales, Image Vis. Comput. 10 (1992) 557–564.

[12] A. Kovashka, K. Grauman, Learning a hierarchy of discriminative space-time neighborhood features for human action recognition, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2010, pp. 2046–2053.
[13] S. Lazebnik, C. Schmid, J. Ponce, A sparse texture representation using local affine regions, IEEE Trans. Pattern Anal. Mach. Intell. 27 (2005) 1265–1278.
[14] S. Lazebnik, C. Schmid, J. Ponce, Beyond bags of features: spatial pyramid matching for recognizing natural scene categories, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, IEEE, 2006, pp. 2169–2178.
[15] X. Li, P.R. Mendonça, R. Bhotika, Texture analysis using minkowski functionals, SPIE Med. Imaging Int. Soc. Opt. Photonics (2012) 83144Y. 83144Y.
[16] D. Mery, J. Chanona-Pérez, A. Soto, J. Aguilera, A. Cipriano, N. Veléz-Rivera, I. Arzate-Vázquez, G. Gutiérrez-López, Quality classification of corn tortillas using computer vision, J. Food Eng. 101 (2010) 357–364.
[17] R. Osada, T. Funkhouser, B. Chazelle, D. Dobkin, Shape distributions, ACM Trans. Graphics 21 (2002) 807–832.
[18] J. Platt, et al., Sequential minimal optimization: a fast algorithm for training support vector machines, 1998.
[19] L. Snidaro, G. Foresti, Real-time thresholding with Euler numbers, Pattern Recogn. Lett. 24 (2003) 1533–1544.
[20] K. Worsley, The geometry of random images, Chance 9 (1996) 27–40.
[21] J. Zhang, M. Marszałek, S. Lazebnik, C. Schmid, Local features and kernels for classification of texture and object categories: a comprehensive study, Int. J. Comput. Vision 73 (2007) 213–238.
[22] F. Zhao, P. Mendonça, R. Kaucic, Image-based automated defect recognition via statistical learning of minkowski functionals, Eur. Conf. NonDestr. Test. (2010) 1–10.