Hindawi Publishing Corporation
EURASIP Journal on Image and Video Processing
Volume 2007, Article ID 17358, 15 pages
doi:10.1155/2007/17358

Research Article

An Ordinal Co-occurrence Matrix Framework for Texture Retrieval

Mari Partio,[1] Bogdan Cramariuc,[2] and Moncef Gabbouj[1]

[1] *Institute of Signal Processing, Tampere University of Technology, P.O. Box 553, 33101 Tampere, Finland*
[2] *Tampere eScience Applications Center, P.O. Box 105, 33721 Tampere, Finland*

Received 5 May 2006; Revised 9 October 2006; Accepted 30 October 2006

Recommended by Jian Zhang

We present a novel ordinal co-occurrence matrix framework for the purpose of content-based texture retrieval. Several particularizations of the framework will be derived and tested for retrieval purposes. Features obtained using the framework represent the occurrence frequency of certain ordinal relationships at different distances and orientations. In the ordinal co-occurrence matrix framework, the actual pixel values do not affect the features, instead, the ordinal relationships between the pixels are taken into account. Therefore, the derived features are invariant to monotonic gray-level changes in the pixel values and can thus be applied to textures which may be obtained, for example, under different illumination conditions. Described ordinal co-occurrence matrix approaches are tested and compared against other well-known ordinal and nonordinal methods.

1. INTRODUCTION

In many application domains, such as remote sensing and industrial applications, the image acquisition process is affected by changes in illumination conditions. In several cases only the structure of gray-level variations is of interest. Therefore, invariance to monotonic gray-level changes is an important property for texture features. In the reality the illumination changes are not necessarily monotonic, but for the case of simplicity in this paper we deal with only monotonic illumination changes.

Several existing methods have considered texture features which are invariant with respect to monotonic changes in gray level using an ordinal approach [1–10]. Monotonic changes in gray levels in two textures refer to the case where the relative order of corresponding gray-level pixel values (in the same positions) in both texture images remains the same. In such situations ordinal descriptors, which are calculated only based on the ranks of the pixels, remain unchanged. N-tuple methods [3] consider a set of N neighbors of the current pixel and can be divided into three different approaches: binary texture co-occurrence spectrum (BTCS), gray-level texture co-occurrence spectrum (GLTCS), and zero-crossings texture co-occurrence spectrum (ZCTCS). All these methods describe texture in two scales, micro-texture and macro-texture, meaning that micro-texture information is extracted from N-tuples and the occurrence of states is used to describe the texture on a macro texture scale.

Binary texture co-occurrence spectrum (BTCS) [8] operates on binarized textures and each N-tuple represents a binary state. Texture information is described by the relative occurrences of these states over a textural region resulting in a 2^n-dimensional state vector. The advantage of this method is its low computational complexity; however, its accuracy is rather low when compared to related methods [3]. In addition, natural textures are seldom binary in nature and therefore the method is limited by the thresholding techniques employed.

Later BTCS method was extended to gray-scale images resulting in gray-level texture co-occurrence spectrum (GLTCS) [9] where rank coding is used in order to reduce feature dimensionality. Intensity values within N-tuples extracted separately for 4 different orientations (horizontal, vertical, and the two diagonals) are ordered and the appropriate rank statistics is incremented. Assuming that $N = 4$, the dimensionality of the spectrum which is used as a feature vector is 96. However, low-order profiles dominate the spectrum resulting in all spectrums appearing similar for all textures.

To obtain better separation of different textures in the feature space, zero-crossings texture co-occurrence spectrum (ZCTCS) was introduced [3]. It first filters the image with the Laplacian of Gaussian aiming to find edges on a specific scale of operation. An important property of this filter is the balance between positive and negative values. Binarization of the filter output reduces the computational complexity while preserving the signs of zero crossings in an image. Finally, the co-occurrence of zero crossings, and thus intensity changes, can be presented by BTCS. Relatively good results are obtained, but the disadvantage is that the method is tuned to a particular scale of operation. Therefore, problems occur when trying to classify textures having some other dominant scale. Assuming that $N = 4$, the dimension of the feature vector is 64, or it could be reduced to 16 if only the cross-operator is used.

In [1] texture co-occurrence spectrum (TUTS) is introduced. Three possible values (0,1,2) can be assigned for the neighbors of the center pixel, depending on whether their value is smaller than, equal to, or greater than the value of the center pixel. In case of 3×3 neighborhood, 6561 texture units are obtained. The resulting texture units are collected into a feature distribution, called texture spectrum, which is used to describe the texture. Therefore, the feature vector dimension is 6561 and the majority of these are not very relevant when describing the texture.

In local binary pattern (LBP) approach [4, 10], a local neighborhood is thresholded into a binary pattern, which makes the distribution more compact and reduces the effect of quantization artifacts. The radius of the neighborhood is specified by R and the number of neighbors within that radius by P. The pixel values in the thresholded neighborhood are multiplied by the binomial weights and these weighted values are summed to obtain the LBP number. The histogram of the operator's outputs accumulated over the texture sample is used as the final texture feature. Rotation invariance can also be achieved by rotating the binary pattern until the maximal number of most significant bits is 0. This reduces the number of possible patterns and, to reduce it even further, the concept of uniform patterns is introduced. Only patterns consisting of 2 or less 0/1 or 1/0 transitions are considered as important and the rest of the patterns are all grouped into miscellaneous bin in the histogram. Since the best classification results for LBP were reported using multiresolution approach with (P, R) values of (8,1), (16,2), and (24,3) [4], those parameter values are used also in the comparative studies of this paper resulting in feature vector of length $10 + 18 + 26 = 54$.

Recently, we have proposed a novel concept based on combining traditional gray-level co-occurrence matrices and ordinal descriptors. We have introduced several practical approaches for building ordinal co-occurrence matrices in [5–7]. In [7], Ordcooc, only the center pixel of a moving window was compared to its anticausal neighbors. However, in that approach problems occurred especially when considering textures with large areas of slightly varying gray levels. In order to improve the robustness, we considered also the other pixels as seed points (Ordcoocmult) [6]. The main drawback of that method was the increase in computational complexity. To overcome this limitation we proposed a method which is a further development and combination of the two approaches (Blockordcooc) [5]. In that method multiple seed points are used for feature construction, as in Ordcoocmult. However, to avoid the increase in computational complexity moving region is first divided into blocks consisting of several pixels, representative value is determined for each block and feature construction is done based on these representative values.

The aim of this paper is to propose a novel common framework for different ordinal co-occurrence matrix approaches and to represent recently proposed approaches as particularizations of the framework. The framework can accommodate other possible variations, and therefore it can be used as a basis for developing also other texture feature extraction methods that are invariant to monotonic gray-scale changes. These feature extraction methods could then be applied to image retrieval and classification applications, for instance.

The paper is organized as follows. In Section 2 a novel ordinal co-occurrence framework is presented. Section 3 represents different ordinal co-occurrence matrix approaches as particularizations of the framework. Complexity evaluation of different ordinal co-occurrence approaches is provided in Section 4. Test databases and experimental results are presented in Section 5. Retrieval performance of the different ordinal co-occurrence methods is compared against some of the existing methods using two sets of well-known Brodatz textures [11]. Finally, conclusions are included in Section 6.

2. FRAMEWORK FOR ORDINAL CO-OCCURRENCE MATRICES

2.1. Description of the ordinal co-occurrence matrix framework

We will here introduce a new ordinal co-occurrence matrix framework based on which various algorithms may be defined to extract relevant texture features for the purpose of content-based indexing and retrieval. The framework is intended to be flexible and versatile. Particularly, it provides local as well as global information at different scales and orientations in the texture. The framework consists of five main blocks. The first block is region selection whose purpose is to divide the texture into local regions, where local ordinal co-occurrence features can be calculated. In the second block ordinal information within these local regions is coded. The aim of the third block is to reduce the dimensions of the local region by combining several pixels into one subregion and specifying a single label for each subregion. Each subregion may span one or more pixels. The goal is to reduce the amount of comparisons needed when extracting local ordinal co-occurrence features specified in the fourth block. The purpose of the fifth block is to use local ordinal co-occurrence matrices for building global ordinal co-occurrence matrices and to normalize the obtained features. The framework structure is presented in Figure 1 and it is further detailed in the following sections.

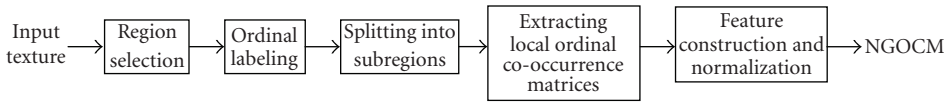


Figure 1: Ordinal co-occurrence matrix framework.

2.2. Image partitioning and region selection

A given arbitrary texture T is split into a set of possibly overlapping regions to allow texture features to be computed locally, that is, $T = \{R_i \mid i = 1, \ldots, L\}$, where L is the number of regions in T. No restrictions are imposed on the region shape or its position. Arbitrary regions could be obtained for example from some previous segmentation step. Although arbitrary shape-based partitioning may be used depending on the application at hand, the general case of arbitrary shaped regions is outside the scope of this paper and thus here we consider only square regions. Local ordinal co-occurrence matrices are then computed over these regions. Global ordinal co-occurrence features for texture T are then calculated based on the local features obtained for each region.

2.3. Ordinal labeling

The purpose of ordinal labeling is to retain the ordinal information of the local region, and to represent it in a compact manner to allow efficient feature construction. Ordinal labeling is done based on a region representative value P_i and its relations with the other pixels within region R_i. The region representative value can for instance be determined from the pixel values and their locations within that specific region. As a simple case, P_i could be the pixel value at the center of the region. We denote by the pixel value a scalar value associated with that pixel. Throughout this paper we use the pixel gray level as pixel value.

Ordinal labeling can be accomplished by any suitable technique. One possibility for ordinal labeling is to divide the values within a region into two or three categories based on the ranking with respect to the value P_i. In this paper, we propose the following labeling:

$$ol = \begin{cases} a, & p - P_i \geq \delta_i, \\ e, & -\delta_i \leq p - P_i < \delta_i, \\ b, & p - P_i < -\delta_i, \end{cases} \tag{1}$$

where ol is the ordinal label of pixel p; a, b, and e are the labels and δ_i is a threshold for region R_i. As a result, the pixels within the region are labeled with two (a and b) or three (a, b, and e) values. When $\delta_i = 0$, the above expression reduces to a binary labeling, where, for example, $a = 1$ and $b = 0$. In binary labeling, equality relation carries label a. Figure 2 illustrates the region labeling into three values. In the examples of this paper it is assumed that label a is set to value 1 and label b to 0.

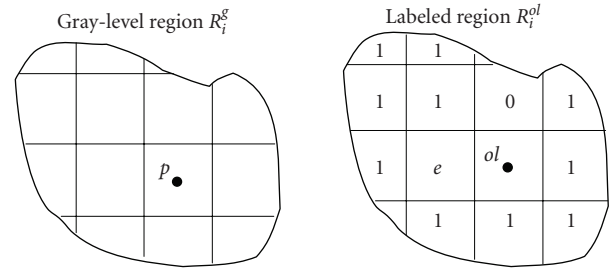


Figure 2: Labeling of the current region when $a = 1$ and $b = 0$.

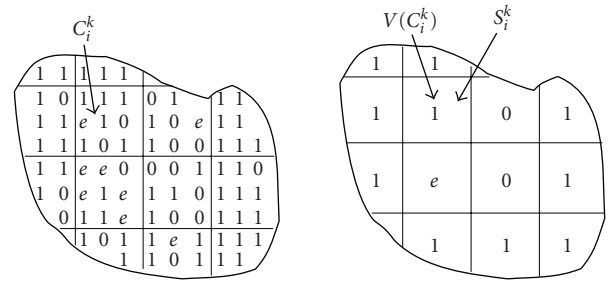


Figure 3: Block-based splitting of a region into subregions of size 3×3 when $a = 1$ and $b = 0$.

2.4. Splitting R_i into subregions

In order to speed up the feature extraction process, each region R_i is further split into M subregions S_i^k, $R_i = \{S_i^k \mid k = 1, \ldots, M\}$. Subregions can in turn assume any shape, but in this paper, we consider subregions to be square blocks of size $bs \times bs$. The resulting subregions may thus contain a single pixel each ($bs = 1$). There might be also cases when the region R_i is not of regular shape or cases where the region size is not multiple of the subregion size. In those cases the representative value for each subregion may be determined based on the available pixel values. Figure 3 shows an example of arbitrary region splitting into blocks of size 3×3 pixels. C_i^k denotes the location of the center of mass of the kth subregion S_i^k; whereas, $V(C_i^k)$ is the representative subregion label value, which could be obtained from the subregion labels for example by majority decision. If the subregion S_i^k contains only one pixel, then $V(C_i^k)$ corresponds to the label of that pixel.

2.5. Local ordinal co-occurrence matrices

Local ordinal co-occurrence matrices capture the co-occurrence of certain ordinal relations between representative

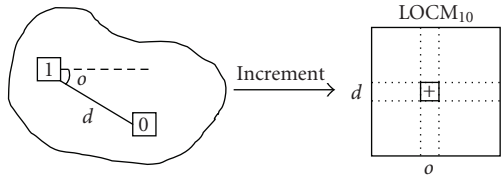


Figure 4: Example of incrementing ordinal co-occurrence matrix.

subregion values $V(C_i^k)$ at different distances d and orientations o. Columns of the matrices represent the occurrences at different orientations o; whereas, rows represent occurrences at different distances d. If r and s represent the labeled values within a set $[a, b, e]$, then the obtained local ordinal co-occurrence matrices can be represented using the notation $\mathrm{LOCM}_{rs}(d, o)$. Therefore, the number of obtained matrices depends on the number of labels used.

Ordinal co-occurrence matrices aim to capture local features relating to possible patterns at different distances and orientations. For example, $\mathrm{LOCM}_{ab}(1, 2)$ represents the number of occurrences where label a occurs at the first specified distance apart from label b at the second specified orientation. Figure 4 shows an incrementing example when labeled values are 1 and 0 occur at distance d and orientation o apart from each other. Therefore $\mathrm{LOCM}_{10}(d, o)$ is incremented.

When binary ordinal labeling is used, the occurrence of horizontal patterns is indicated if the first column of matrix $\mathrm{LOCM}_{aa}(d, o)$ contains greater values than the rest of the columns. Similarly, the largest values in the middle column, corresponding to -90° orientation, would suggest occurrence of vertical patterns. On the other hand, $\mathrm{LOCM}_{ab}(d, o)$ suggests the frequency in which the differently labeled values occur at each orientation.

2.6. Feature construction and normalization

Global ordinal co-occurrence matrices $\mathrm{GOCM}_{rs}(d, o)$ represent the features of the whole texture area T and they are incremented based on $\mathrm{LOCM}_{rs}(d, o)$ from each region R_i. Before feature evaluation or comparison, matrices $\mathrm{GOCM}_{rs}(d, o)$ are normalized. Normalization can be done for example by counting the number of used pairs for different distances and orientations and dividing each position in the global matrices with the corresponding value. The purpose of the normalization is to make the features independent on the size of the texture T. The resulting normalized ordinal co-occurrence matrices $\mathrm{NGOCM}_{rs}(d, o)$ are used as features for the texture T.

3. VARIANTS WITHIN THE PROPOSED FRAMEWORK

Several variants can be defined within the proposed ordinal co-occurrence framework depending on the application at hand. First of all, ordinal labeling may be performed in different ways as mentioned previously. Different methods for incrementing global matrices based on local matrices may be utilized, for example, by selecting only some of the columns. In addition, alternative approaches for normalizing the ordinal co-occurrence matrices may be derived. After ordinal labeling different methods may be used for selecting how labeled values are used for incrementing the local ordinal co-occurrence matrices.

In this section we describe three variants within the proposed framework. In variant 1 only the center pixel of region R_i is compared to its neighbors. The advantage of this approach is that it captures salient features within the texture; furthermore, the computational complexity is low. However, in that approach problems occur especially when considering textures with large areas of slightly varying gray levels. With the aim of improving the robustness of variant 1, variant 2 compares all pixels within a region to their neighbors. Since in variant 2 the pixel pair is not always fixed to the center pixel, the method is capable to detect more details from texture than variant 1. However, the main drawback of variant 2 is the increase in computational complexity. In order to avoid the increase in computational complexity but still to obtain robust features, multiple seed points are used in variant 3, as in variant 2, whereas a block-based approach for building the ordinal co-occurrence matrices is utilized in order to keep the computational complexity low. The rest of this section details these variants.

Common for all the variants is that they construct features representing the occurrence frequency of certain ordinal relationships ("greater than," "equal to," "smaller than") at different distances d and orientations o. Each of the matrices is of size $N_d \times N_o$, where N_d stands for the number of distances and N_o for the number of orientations. These numbers may be varied. To enable comparison of ordinal co-occurrence matrices obtained from varying texture sizes, the resulting ordinal co-occurrence matrices are normalized by the total number of pairs with the corresponding distance and orientation when moving over the region T. This information is saved in matrix ALL_COOC. The normalization is performed after building the global ordinal co-occurrence matrices. In pseudocodes shown in Algorithms 1, 2, and 3, implementing variant 1, variant 2, and variant 3, respectively, the normalization is presented at the end. The resulting ordinal co-occurrence matrices are used to characterize the texture. The total difference between two textural regions T_1 and T_2 can be obtained by summing up the differences from matrix comparisons using, for example, the Euclidean distance. In the comparison we assume that the same number of distances and orientations are used for both textural regions. In the following, three different methods for building the ordinal co-occurrence matrices are introduced and their advantages and disadvantages are evaluated.

In the variants presented in this section we make the following assumptions. For the case of simplicity we assume that both texture T and region R_i are square shaped. It is also assumed that label $a = 1$ and $b = 0$. In the described variants $\mathrm{LOCM}_{rs}(d, o)$ are used as such for incrementing $\mathrm{GOCM}_{rs}(d, o)$, and therefore $\mathrm{GOCM}_{rs}(d, o)$ are directly incremented.

```
(1)  FOR all regions in T
(2)    Label pixels within region R_i
(3)    FOR all anticausal neighbors X_j of C_i^m
(4)      Increment ALL_COOC(d, o)
(5)      IF (V(C_i^m) = e & V(X_j) = e)          Increment GOCM_ee(d, o)
(6)      ELSEIF (V(C_i^m) = e & V(X_j) = 0)      Increment GOCM_e0(d, o)
(7)      END
(8)    ENDFOR
(9)  ENDFOR
(10) Normalize GOCM_ee and GOCM_e0 with ALL_COOC
```

Algorithm 1: Pseudocode for the Ordcooc method.

```
(1)  FOR all possible regions in T
(2)    Label pixels within region R_i
(3)    FOR all C_i^k in R_i
(4)      FOR all anticausal neighbors X_j of C_i^k
(5)        Increment ALL_COOC(d, o)
(6)        IF (V(C_i^k) = 0 & V(X_j) = 0)          Increment GOCM_00(d, o)
(7)        ELSEIF (V(C_i^k) = 1 & V(X_j) = 0)      Increment GOCM_10(d, o)
(8)        ELSEIF (V(C_i^k) = 0 & V(X_j) = 1)      Increment GOCM_01(d, o)
(9)        END
(10)     ENDFOR
(11)   ENDFOR
(12) ENDFOR
(13) Normalize GOCM_00, GOCM_10 and GOCM_01 with ALL_COOC
```

Algorithm 2: Pseudocode for the Ordcoocmult method.

3.1. Basic ordinal co-occurrence (Ordcooc)

In the basic ordinal co-occurrence approach, Ordcooc, only the center pixel of each region R_i is compared to its neighbors [7]. Therefore, the most important ordinal relations within each region may be saved to the global ordinal co-occurrence matrices GOCM_{rs}. The advantage of this approach is that salient features of the texture can be captured; furthermore, the computational complexity is low.

The implementation of this variant is based on going through all pixels in the textural area T. The processing is done using a moving region R_i. The size of the region depends on the number of distances N_d used, and in general case subregions are considered as distance units. For this variant bs, that is, dimension of the subregion, is 1 and therefore N_d represents the number of distances used in pixels. C_i^k represents the center of mass of the subregion S_i^k within R_i. Since for this variant each subregion S_i^k contains only one pixel, C_i^k represents the position of that pixel and $V(C_i^k)$ represents its label. In the following descriptions we assume that C_i^m denotes the center of mass of S_i^m, the center most subregion of R_i. In general case region R_i, which consists of M subregions S_i^k, can be defined as follows:

$$R_i = \left\{(x,y) \mid \mathrm{dist}\left((x,y), C_i^m\right) \leq N_d \times bs + \left\lfloor \frac{bs}{2} \right\rfloor \right\} = \bigcup_{k=1}^{M} S_i^k. \tag{2}$$

Since in this variant each subregion consists of only one pixel, that is, $bs = 1$, definition of region R_i can be simplified as follows:

$$R_i = \left\{(x,y) \mid \mathrm{dist}\left((x,y), C_i^m\right) \leq N_d\right\} = \bigcup_{k=1}^{M} S_i^k. \tag{3}$$

When building ordinal co-occurrence matrices in this particular method only the representative value $V(C_i^m)$ of the center most subregion S_i^m is used as a seed point and is compared to its anticausal neighbors X defined by expression (4). In the definition we assume that region R_i is scanned in row-wise order (from top left to bottom right) and the center of mass locations of the subregions S_i^k are saved into a 1-dimensional array $\mathrm{ind}(C_i^k)$, where $k = 1, \ldots, M$ and M is the number of subregions.

$$X \subset R_i,$$
$$X = \left\{C_i^k \mid d = \mathrm{dist}\left(C_i^k, C_i^m\right) \leq N_d, \mathrm{ind}\left(C_i^k\right) > \mathrm{ind}\left(C_i^m\right)\right\}. \tag{4}$$

We denote by X_j the elements of the set X. The pseudocode is shown in Algorithm 1.

Ordinal labeling is performed for every pixel p within region R_i with respect to the region representative value, which is now selected to be $T(C_i^m)$, the pixel value at the center of mass of the center most subregion of S_i^m. Determination of

```
(1)  FOR all possible regions R_i in T
(2)    Label pixels within region R_i
(3)    FOR all subregions S_i^k in R_i
(4)      Determine representative subregion value V(C_i^k) by majority decision
(5)    ENDFOR
(6)    FOR all C_i^k in R_i
(7)      FOR all anticausal neighbors X_j of C_i^k
(8)        Increment ALL_COOC(d, o)
(9)        IF (V(C_i^k) = 0 & V(X_j) = 0)          Increment GOCM_00(d, o)
(10)         ELSEIF (V(C_i^k) = 1 & V(X_j) = 0)    Increment GOCM_10(d, o)
(11)         ELSEIF (V(C_i^k) = 0 & V(X_j) = 1)    Increment GOCM_01(d, o)
(12)       END
(13)     ENDFOR
(14)   ENDFOR
(15) ENDFOR
(16) Normalize GOCM_00, GOCM_10 and GOCM_01 with ALL_COOC
```

Algorithm 3: Pseudocode for the Blockordcooc method.

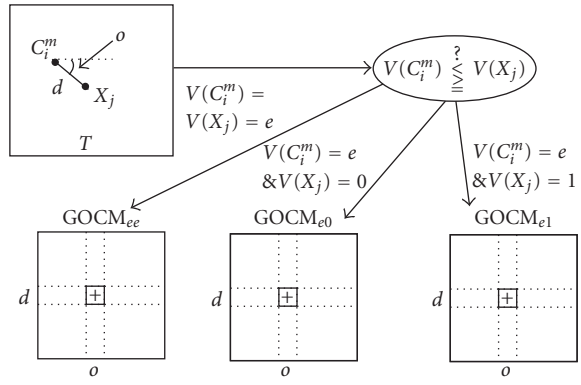


Figure 5: Ordcooc.

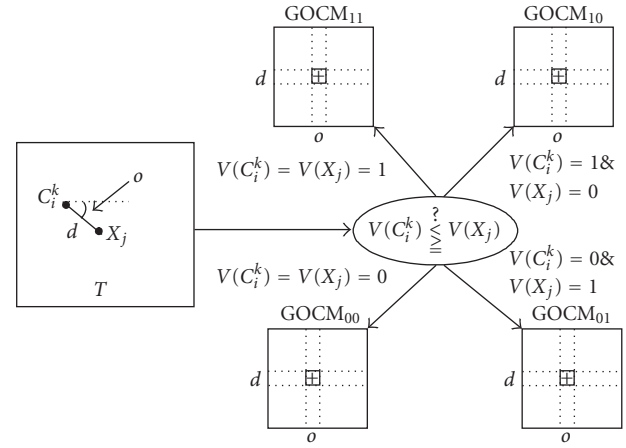


Figure 6: Ordcoocmult.

ordinal labels *ol* can be done using (1). To obtain the label e only in the case where p equals $T(C_i^m)$ and label a only in case where p is greater than $T(C_i^m)$ we assume that for all $\delta_i \in]0, 1[$ and both p and $T(C_i^m)$ are integers. Since δ_i is not equal to 0, ternary labeling is applied to R_i.

The results are saved in the form of ordinal co-occurrence matrices, which are incremented based on the values and spatial relationships of the current pixel and its neighbors. All occurrences of distance and orientation patterns are saved in matrix ALL_COOC for normalization purposes. If $V(C_i^m)$ and $V(X_j)$ both are e, then the matrix $GOCM_{ee}$ is incremented. On the other hand, if $V(C_i^m)$ is e and $V(X_j)$ is 0, the matrix $GOCM_{e0}$ is incremented. We could also consider a third relation where $V(C_i^m)$ is e and $V(X_j)$ is 1. However, this information could also be obtained from $GOCM_{ee}$, $GOCM_{e0}$ and ALL_COOC matrices. Therefore, the resulting normalized ordinal co-occurrence matrices, $NGOCM_{ee}$ and $NGOCM_{e0}$, are used as features for the underlying textural region. Figure 5 illustrates how the different matrices are incremented based on the pixel comparisons.

3.2. *Ordinal co-occurrence using multiple seed points (Ordcoocmult)*

This approach differs from the basic ordinal co-occurrence by considering also other pixels inside each region as seed points [6]. The advantage of this approach is that robustness of the features is greatly improved when compared to variant 1 since now more details within the local region can be captured. The drawback of this variant is the increased computational complexity when compared to the variant 1, as will be detailed later.

As in variant 1, each subregion S_i^k contains only one pixel, C_i^k represents the position of that pixel, and $V(C_i^k)$ represents its value. Ordinal labeling of each region is done with respect to $T(C_i^m)$, pixel value at the center of mass of the center most subregion of R_i. Since δ_i is selected to be 0 in this particular case, binary labeling is applied to R_i. The ordinal labels *ol* for each pixel within R_i can be determined using (1).

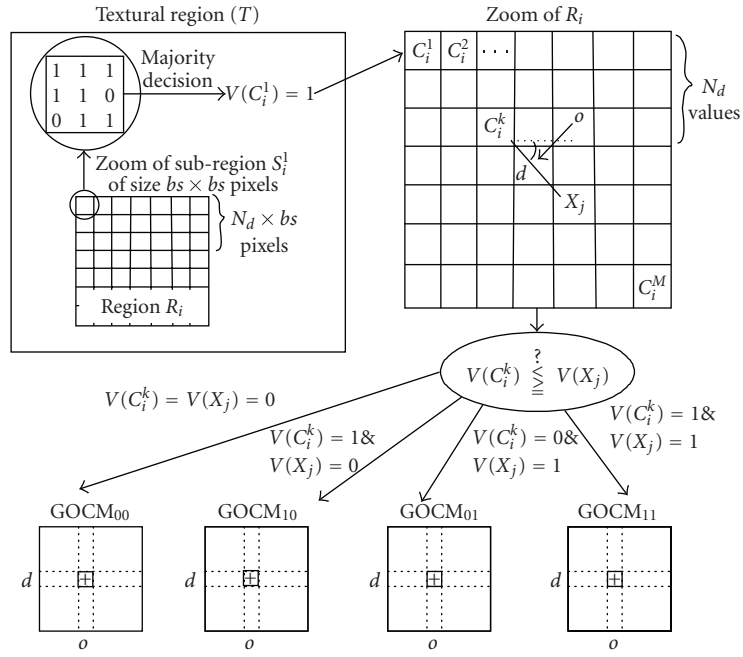


Figure 7: Blockordcooc.

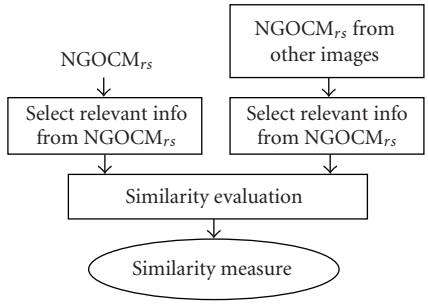


Figure 8: Block diagram of similarity evaluation.

When building ordinal co-occurrence matrices for this particular method, all C_i^k are used as seed points and their representative value $V(C_i^k)$ is compared to their anticausal neighbors defined by expression (4) and occurrences of ordinal relations are updated in ordinal co-occurrence matrices GOCM$_{rs}(d, o)$. The pseudocode of the method is shown in Algorithm 2.

GOCM$_{11}(d, o)$ represents the occurrences of $V(C_i^k)$ and its neighbor both being equal to 1 at distance d and orientation o, while GOCM$_{00}(d, o)$ represents the case when both values are 0. GOCM$_{10}(d, o)$ shows the occurrences where $V(C_i^k)$ is 1 and the label of its neighbor is 0 at (d, o). The opposite case is represented in GOCM$_{01}(d, o)$. All ordinal co-occurrence matrices are normalized using ALL_COOC matrix and the obtained normalized matrices NGOCM$_{rs}(d, o)$ are used as features.

In the Ordcoocmult approach presented in [6] all four matrices are used as features, but since the information of one of the relations could be obtained from the other matrices and ALL_COOC matrix, one of the matrices could be left out of the comparisons. We have selected to leave out matrix NGOCM$_{11}$. This also reduces the dimensionality of the computed features. Based on the comparison between the pixel values, the corresponding cell in the corresponding matrix is incremented, as shown in Figure 6.

3.3. Block-based ordinal co-occurrence using multiple seed points (Blockordcooc)

This approach utilizes multiple seed points just as in variant 2, but the significant difference is that after ordinal labeling, the values in region R_i are divided into subregions consisting of more than one pixel and comparison is performed using the representative subregion values $V(C_i^k)$ [5]. Therefore, the number of comparisons can be greatly reduced. The aim of this variant is to keep the computational complexity on the level of variant 1, that is, Ordcooc, but to obtain robustness of variant 2, that is, Ordcoocmult.

Since in this approach subregion size is greater than 1, N_d represents the number distances used using subregions as distance units. Since bs denotes dimension of the subregion, $w_p = 2 \times bs \times N_d + bs$ represents the width of the region R_i in pixels. After combining the pixels within each of the subregions we obtain a sampled region of width $w = 2 \times N_d + 1$. Therefore, processing of similar size neighborhoods in pixels as in the earlier approaches is possible with a smaller N_d and hence the dimensions of the ordinal co-occurrence matrices become smaller.

Since for this variant each subregion S_i^k contains more than one pixel, C_i^k represents center of mass of that subregion, and $V(C_i^k)$ is the representative value of the

corresponding subregion. In the following descriptions we assume that C_i^m denotes the location of center of mass of the center most subregion of S_i^m. If subregion size is even and no pixel is located at the actual center of mass of that subregion, then center of mass is selected to be location of one of the four pixels closest to the actual center of mass. Similar to the earlier approaches, labeling of each region is performed with respect to $T(C_i^m)$. Since δ_i is set to 0 in this case, binary labeling is applied to R_i. The ordinal label for each pixel is determined using expression (1). Features are now computed in a block based manner, and therefore, in the splitting step, the subregion size can be selected to be greater than 1. The representative value $V(C_i^k)$ for each of the subregions is determined by the majority decision. If the majority of values within the subregion are 1s, then the value of the subregion is set to 1. On the other hand, if the majority of values equal to 0, the value for the subregion is set to 0. If an equal number of ones and zeros occur, the label of the center pixel within the corresponding subregion is selected.

When building ordinal co-occurrence matrices using this method, all C_i^k are used as seed points and their representative values $V(C_i^k)$ are compared to their anticausal neighbors Xdefined by expression (4). Finally, the matrices are incremented in a similar manner as in Section 3.2. The procedure for building the block-based ordinal co-occurrence matrices is shown in Figure 7, and Algorithm 3 describes the pseudocode of the method. X_j denotes the elements of the set X.

3.4. Similarity evaluation

A similarity measure between two different textural regions T_1 and T_2 can be obtained by summing up the differences of the corresponding matrices. We assume that the same number of distances and orientations are used for calculating the features for both textural regions. The block diagram of the similarity evaluation is represented in Figure 8. In some special cases only some orientations or distances might be of importance, therefore relevant information selection block is included before similarity evaluation step. However, in this paper NGOCM$_{rs}$ are used as such in the similarity evaluation step. Different similarity metrics can be applied in the similarity evaluation step; however, in this paper only the Euclidean metric is used.

4. COMPLEXITY ANALYSIS OF PROPOSED ORDINAL CO-OCCURRENCE VARIANTS

We will here evaluate the complexity of the variants described in Section 3. Since the variants differ in terms of block size and number of seed points, evaluation is based on the average number of pixel pairs taken into consideration per each pixel in T. Let us denote by β_i this number, where i represents the variant 1, 2, or 3. This evaluation is an approximation since in the actual calculations only the pairs up to distance N_d are considered. Let us denote by α_i the number of pairs considered per region R_i. In the following descriptions w_p is the width of the region R_i in pixels; whereas, w is the width of the region R_i in blocks.

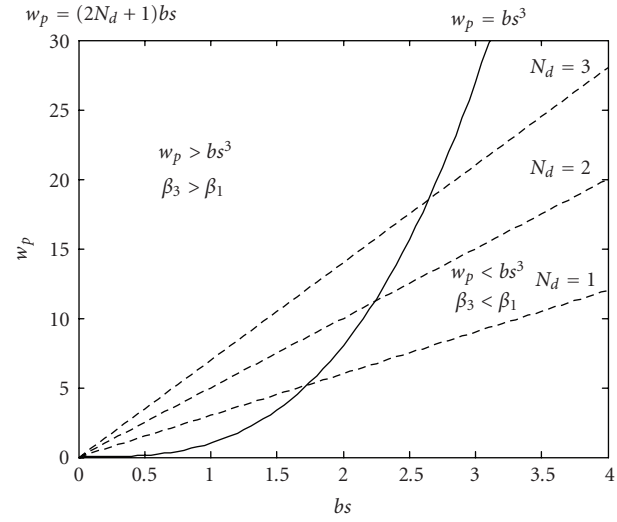


Figure 9: Relation of block size bs and window size w_p in the complexity evaluation of Blockordcooc and Ordcooc.

For variant 1, Ordcooc:

$$\beta_1 = \alpha_1 = \frac{1}{2}w_p^2. \tag{5}$$

For variant 2, Ordcoocmult:

$$\alpha_2 = \frac{w_p^2(w_p^2 - 1)}{2} \approx \frac{1}{2}w_p^4, \qquad \beta_2 = \alpha_2 \approx \frac{1}{2}w_p^4. \tag{6}$$

For variant 3, Blockordcooc, each region R_i is divided in blocks, and therefore $w_p = w \times bs$. In this case,

$$\alpha_3 = \frac{w^2(w^2 - 1)}{2} \approx \frac{1}{2}w^4 = \frac{1}{2}\frac{1}{bs^4}w_p^4. \tag{7}$$

Due to the fact that the region R_i is moved one block size at a time β_3 will be

$$\beta_3 = \frac{\alpha_3}{bs^2} = \frac{1}{2}\frac{1}{bs^6}w_p^4 = \left(\frac{w_p}{bs^3}\right)^2\beta_1. \tag{8}$$

It can be noted that $\beta_3 < \beta_2$ always, since $bs > 1$. It is also clear that $\beta_1 < \beta_2$, since $w_p > 1$. The relation between β_1 and β_3 is not static but it depends on bs and w_p. β_1 is identical to β_3 when $bs^3 = w_p$. Starting from a pair of values (bs, w_p) satisfying the above condition, if w_p is increased, then the complexity of Blockordcooc becomes bigger than that of Ordcooc. However, if bs is increased, then the complexity of Blockordcooc becomes lower than that of Ordcooc. This relation can also be seen in Figure 9. For $bs = 2$ from (6) and (8) we have $\beta_2/\beta_3 = 64$ from where it can be seen that Blockordcooc is significantly less complex than Ordcoocmult.

5. EXPERIMENTAL RESULTS WITH BRODATZ IMAGES

5.1. Test databases

In the retrieval experiments, we used 2 databases. Test database 1 consists of 60 classes of Brodatz textures [11]. The images were obtained from the largest available already digitized

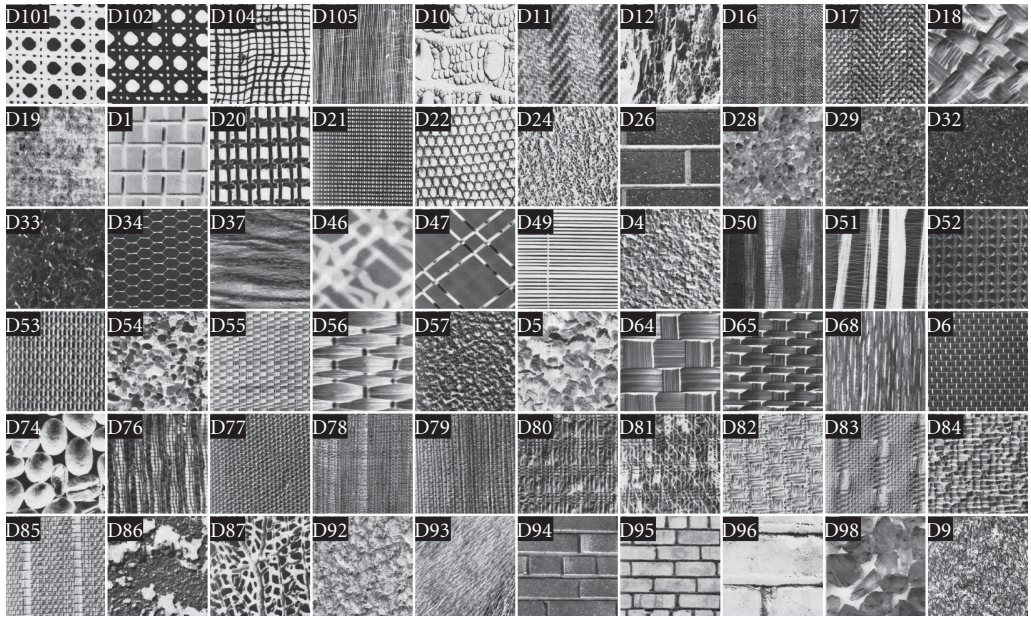


Figure 10: Sample Brodatz textures used in the experiments.

set of Brodatz textures [12], where the size of each texture is 640×640 pixels. To populate the test database, each texture was divided into 16 nonoverlapping texture patches of size 160×160. Originally the set contains 111 different Brodatz textures, however the scale in some of them is so large, that after splitting into 16 patches the patches do not really contain texture (no apparent repeating patterns). Therefore, we decided to use 60 classes of Brodatz textures in retrieval experiments (those textures were included in the experiments which were more likely to have some repeating patterns also after division to 16 subimages). Thus Test database 1 contains altogether 960 textures. Sample images from all of the used texture classes are shown in Figure 10.

In order to test the invariance to monotonic gray-level changes, we created Test database 2, which contains the same images as Test database 1, but now in addition to original image, the database contains also such sample where the overall gray level is decreased and also such sample where the overall gray level is increased. In this simple experiment the gray-level change is monotonic and uniform which is not always true in the nature. Since Test database 2 contains three different versions of each texture sample, it contains altogether 2880 images.

Overall gray level of image I_1 is decreased using monotonic function $f_{\text{dec}}(x) = x - c$, where c is a positive constant. In a similar manner, overall gray level of I_1 can be increased using monotonic function $f_{\text{inc}}(x) = x + c$. Since the pixel values in the images in question are limited within the range $[0 \cdots 255]$, adding and subtracting value c from pixel values close to the limits of the interval may result into saturation, that is, more pixel values 0 or 255 occur in the resulting image than in the original one. In order to avoid too much saturation, we have selected to keep the value for c low. Test database 2 is obtained by setting $c = 10$, since, for example, for class D32, over 10% of original pixel values are below 15. Image I_2, where overall gray level is increased is obtained as follows: $I_2 = f_{\text{inc}}(I_1)$. In a similar manner, image I_3, with the decreased overall gray level can be produced as follows: $I_3 = f_{\text{dec}}(I_1)$. Some example images with the described monotonic gray-level changes are shown in Figure 11, however since the gray-level change is not large, the visible difference is only minor.

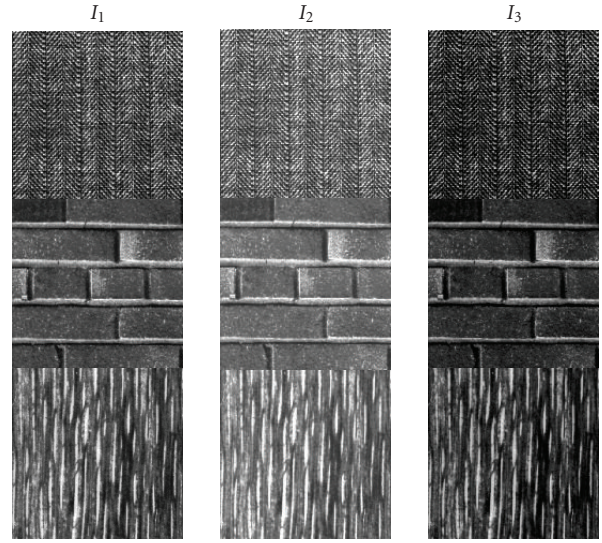


Figure 11: Examples of monotonic gray-level changes: I_1 is the original image, I_2 is obtained by adding a constant c to I_1, and I_3 by subtracting c from I_3.

Table 1: Average retrieval results for different methods using test database 1.

Class	GABOR	LBP	GLTCS	ZCTCS	GLCM	Ordcooc	Ordcoocmult	Blockordcooc
D101	51.2	100.0	79.3	100.0	97.7	61.7	96.5	97.3
D102	58.6	100.0	78.5	100.0	87.9	63.7	100.0	100.0
D104	100.0	100.0	99.6	100.0	100.0	98.8	100.0	98.8
D105	93.4	98.8	100.0	98.8	97.3	69.5	100.0	98.0
D10	75.0	82.4	81.6	51.6	74.2	39.1	88.3	88.3
D11	99.6	77.3	100.0	89.5	60.2	97.3	100.0	100.0
D12	93.8	75.0	54.7	53.9	42.6	35.9	63.3	60.5
D16	100.0	100.0	100.0	100.0	84.8	100.0	100.0	100.0
D17	100.0	100.0	100.0	99.6	50.0	88.7	100.0	100.0
D18	99.2	90.2	80.5	76.2	45.7	87.9	97.3	97.7
D19	88.7	81.3	98.0	59.0	54.7	74.2	95.7	88.7
D1	99.6	83.6	94.9	56.6	74.6	93.0	99.6	96.5
D20	100.0	100.0	87.9	100.0	93.0	79.7	100.0	98.8
D21	100.0	100.0	100.0	100.0	100.0	84.8	100.0	100.0
D22	95.3	100.0	60.9	75.8	49.6	52.0	74.6	71.9
D24	88.3	91.4	74.6	82.8	89.8	55.5	86.3	78.1
D26	96.1	87.9	95.7	80.9	90.2	95.7	94.5	91.0
D28	91.8	90.6	91.0	82.0	42.2	70.3	94.9	91.4
D29	100.0	100.0	100.0	89.5	93.8	87.5	100.0	100.0
D32	100.0	96.5	82.0	60.5	54.3	49.6	82.4	84.4
D33	91.4	99.6	92.6	67.2	46.1	50.8	93.0	94.5
D34	100.0	100.0	78.9	100.0	82.8	68.4	97.3	95.7
D37	97.7	78.9	85.9	97.3	47.3	72.3	97.3	96.5
D46	75.8	92.2	98.8	32.8	81.6	64.5	98.0	96.9
D47	100.0	98.8	93.8	100.0	79.3	46.1	87.9	78.1
D49	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
D4	92.2	56.3	74.2	85.2	58.2	91.8	98.0	98.8
D50	83.6	90.2	87.9	33.6	40.2	43.4	82.8	74.6
D51	85.2	97.3	94.1	66.4	44.9	44.9	91.0	89.1
D52	74.2	99.6	96.1	97.3	46.1	64.5	96.5	94.9
D53	100.0	100.0	100.0	100.0	90.6	100.0	100.0	100.0
D54	62.1	81.3	55.5	37.5	35.9	54.3	72.3	68.4
D55	100.0	100.0	100.0	99.6	53.1	98.0	100.0	99.6
D56	100.0	100.0	100.0	79.7	67.2	100.0	100.0	100.0
D57	100.0	99.2	100.0	99.6	92.6	98.8	100.0	100.0
D5	84.4	55.5	66.4	44.1	48.4	69.9	80.1	77.7
D64	97.7	79.7	77.7	85.5	66.4	85.2	97.3	96.9
D65	99.2	90.2	100.0	58.2	92.6	99.2	100.0	99.6
D68	100.0	94.9	79.3	97.3	72.7	68.8	97.7	98.8
D6	100.0	100.0	75.0	91.4	88.7	51.2	100.0	98.8
D74	92.6	99.2	96.5	72.7	54.7	85.2	96.5	94.5
D76	100.0	100.0	79.3	96.1	53.5	76.2	99.6	91.8
D77	100.0	100.0	100.0	100.0	60.5	100.0	100.0	100.0
D78	84.0	96.9	99.6	62.5	49.2	100.0	97.7	98.0
D79	84.8	81.6	93.4	93.0	44.5	89.5	88.7	98.0
D80	91.8	78.5	91.4	60.9	32.8	93.8	88.3	89.8
D81	94.9	87.9	91.8	87.5	37.9	69.9	93.4	87.5
D82	98.0	100.0	100.0	99.6	91.8	95.3	100.0	100.0
D83	100.0	92.2	100.0	96.5	84.4	100.0	96.9	100.0
D84	100.0	100.0	100.0	87.1	68.0	100.0	98.8	96.5
D85	100.0	99.6	100.0	95.7	73.4	99.2	94.5	98.8
D86	71.1	70.7	77.3	48.4	48.0	47.3	89.8	79.3
D87	95.3	89.8	96.5	50.8	61.3	89.8	87.5	91.0
D92	94.5	72.7	99.6	80.5	69.5	83.2	91.4	91.8
D93	97.7	98.0	84.8	70.7	42.6	45.3	83.2	77.0

Table 1: Continued.

Class	GABOR	LBP	GLTCS	ZCTCS	GLCM	Ordcooc	Ordcoocmult	Blockordcooc
D94	95.3	68.8	73.4	44.9	33.6	46.1	82.0	86.3
D95	99.2	87.9	98.8	67.6	68.0	69.5	96.5	91.0
D96	76.2	93.4	94.5	59.4	76.2	53.9	98.0	96.5
D98	90.2	96.9	82.4	55.1	56.3	71.9	69.1	66.8
D9	93.0	56.6	71.1	73.4	75.8	64.8	87.1	92.2
Ave%	92.2	90.7	89.1	78.9	66.7	75.6	93.4	92.1

Table 2: Average results for Test databases 1 using different parameter values for variant 3, Blockordcooc.

N_d	N_o	bs	Average
2	4	2	89.79
3	4	2	92.12
4	4	2	91.88
5	4	2	91.23
8	4	2	86.20
5	4	3	79.58
10	4	3	61.65

Table 3: Average retrieval accuracies using Test database 1 (TD1) and Test database 2 (TD2).

Method	Ave% for TD1	Ave% for TD2
Blockordcooc	92.1	92.1
Ordcoocmult	93.4	93.4
Ordcooc	75.6	75.6
LBP	90.7	90.7
GLTCS	89.1	89.1
ZCTCS	78.9	78.9
GABOR	92.2	92.2
GLCM	66.7	62.6

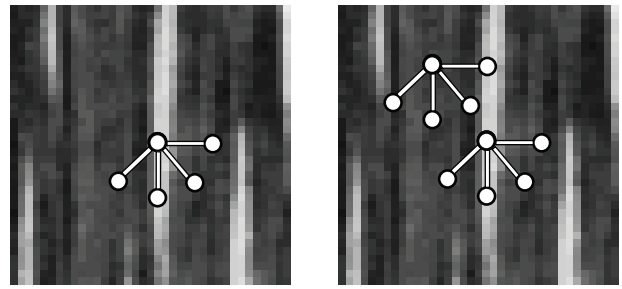
5.2. Experiments

Since Test database 1 consists of 16 images from different classes, the 16 best matches are considered in the retrieval. The average results for different methods in Table 1 indicate the average amount of correct matches for all the classes in Test database 1. There Ordcooc presents the results using variant 1, whereas Ordcoocmult refers to variant 2. Both of the methods are applied with parameters $N_d = 5$ and $N_o = 4$. Results using variant 3, that is, Blockordcooc, with different parameters shown in Table 2 indicate the average amount of correct matches for all the classes in Test database 1. There Ordcooc presents the results using variant 1, whereas Ordcoocmult refers to variant 2. Both of the methods are applied with parameters $N_d = 5$ and $N_o = 4$. Results using variant 3, that is, Blockordcooc, with different parameters are shown in Table 2. In Table 1 results of Blockordcooc are shown with parameters $N_d = 3$, $N_o = 4$, and $bs = 2$, which suggest the best retrieval accuracy according to Table 2. LBP refers to rotation invariant local binary pattern operator $\text{LBPRIU}_{P,R}$ [4], with (P, R) values (8,1), (16,2), and (24,3). GLTCS features are calculated using interpixel spacing $k = 1$ and ZCTCS is applied using interpixel spacing $k = 1$, since with these parameter values the best results are reported in [2].

BTCS and TUTS methods are left out of comparisons, since their performance is already shown to be lower than GLTCS and ZCTCS [2]. The feature dimensionality of the TUTS method is also much higher than that of the other methods considered in this study. In addition to ordinal methods gray-level co-occurrence matrices GLCM [13] (with $d = 1$ and $d = 2$ with orientations 0, 45, 90, and 135) and Gabor wavelet features (6 orientations and 4 scales) [14] are evaluated for comparison purposes. Euclidean distance is applied as similarity metric for all the other methods except for local binary pattern approach where the similarity evaluation is carried out by using G (log-likelihood) statistic as reported in [4]. In order to test the invariance to monotonic gray-level changes, we perform the retrieval experiments using Test database 2. Now, one image from each class is used as the query image, and since each class contains 48 samples, 48 best matches are evaluated in the retrieval. For comparison purposes, retrieval accuracies are shown in percentages in Table 3.

5.3. Evaluation of the results for the different ordinal co-occurrence approaches

As can be seen from Table 1, the average retrieval accuracy of variant 1, Ordcooc, outperforms gray-level co-occurrence matrices (GLCM). However, its performance remains lower than for other evaluated methods. According to Table 1 variant 2, that is, Ordcoocmult, outperforms all the other evaluated methods. However, the drawback of that method is the increased computational complexity. Average retrieval accuracy of variant 3, that is, Blockordcooc, is only a little lower than that of Ordcoocmult and it outperforms Ordcooc significantly, whereas the computational complexity is reduced to the level of Ordcooc by adopting the block-based approach. Therefore we can consider variant 3 to be best among the evaluated ordinal co-occurrence matrix approaches when considering both retrieval accuracy and computational simplicity. As can be seen from Table 2, the best retrieval accuracy for Blockordcooc using the Test database



(a) Ordcooc and only one seed point

(b) Ordcoocmult and multiple seed points

Figure 12: Properties of different ordinal co-occurrence approaches.

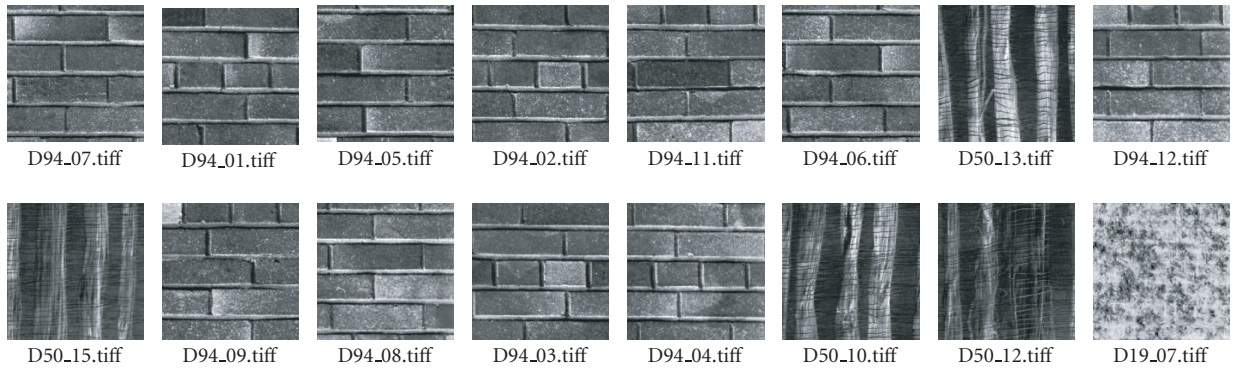


Figure 13: Sample query results (16 best matches) for Ordcooc: D94_07 used as query, accuracy 68.75%.

1 described in Section 5.1 can be obtained by using $N_d = 3$ and $bs = 2$. Degradation of the results by the increasing distance might be explained by the increased amount of noise in matrices with bigger distances. Increasing the block size naturally also degrades the results somewhat since the amount of simplification increases.

According to Section 4 computational complexity of Ordcooc and Blockordcooc are similar when $\beta_1 = \beta_3$, that is, when for example $bs = 2$ and $w_p = 8$. This case is close to first line in Table 2, where $w_p = 10$. From Table 1 one can see that for most of the classes Ordcoocmult and Blockordcooc outperform Ordcooc approach. Ordcooc performs slightly better than Blockordcooc for Brodatz texture classes D26, D78, D84, and D85, although the difference in performance in these cases is only minor. However, when querying with the class D26, Ordcooc falsely returns some samples from class D37, whereas for Blockordcooc the mismatches come from class D94. Although the amount of mismatches is somewhat greater for Blockordcooc, the mismatches are visually more relevant than for Ordcooc. Also when using Blockordcooc for querying with class D78 the mismatches are from the visually relevant class D79. Similarly, when querying with class D85, the mismatching classes are visually quite similar (D80 and D83).

Differences in behavior of variant 1, that is, Ordcooc, and other ordinal co-occurrence matrix approaches can be explained as follows. Ordcooc is computationally much simpler than Ordcoocmult, since it considers only center pixel of a current region as seed point whereas in Ordcoocmult all pixels within region are considered as seed points. However, detection of patterns with Ordcooc is not as robust as the approaches considering multiple seed points. As an example we could consider texture with bright stripes on a dark background, for example, part of D68 from Brodatz textures as shown in Figure 12. There the region representative value $T(C_i^m)$ is located on a brighter stripe, which causes problems in the case (a) of Figure 12. In that case other values located on that stripe will be labeled with value e only in cases where they equal to $T(C_i^m)$, which might be very seldom. Therefore, the stripe might not be clearly detected, although it contains only small variations in gray scale. In addition in Ordcooc the other pixel of the pixel pair has always value $T(C_i^m)$ and therefore the darker vertical areas next to the bright stripe might not be well detected. Variants 2 and 3, which utilize multiple seed points, alleviate this problem since the considered label pair is not always fixed to the center value itself, as shown in the case (b) of Figure 12. Sample retrieval results using Ordcooc and Blockordcooc can be seen in Figures 13–16. Left most texture image on the upper row is used as a query image and the best matches are shown in row-wise order. Also from these examples it can be seen that using multiple seed points improves the retrieval accuracy significantly.

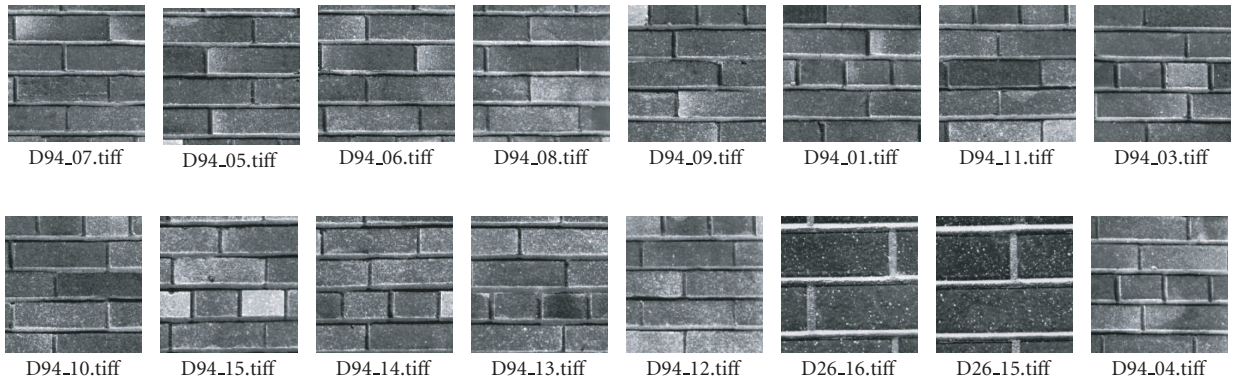


Figure 14: Sample query results (16 best matches) for Blockordcooc: D94_07 used as query, accuracy 87.50%.

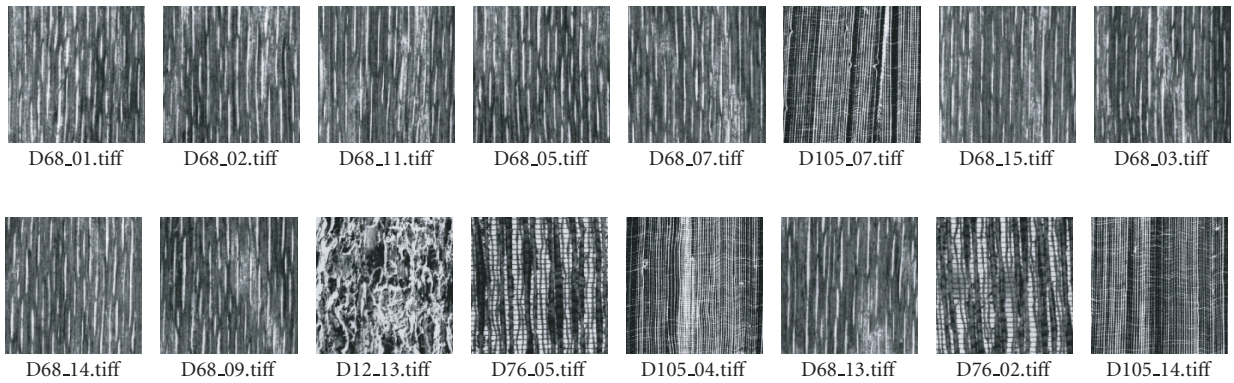


Figure 15: Sample query results (16 best matches) for Ordcooc: D68_01 used as query, accuracy 62.50%.

According to Table 1 average retrieval accuracy of Blockordcooc remains lower than 75% for Brodatz classes D12, D22, D50, D54, and D98. However, other ordinal co-occurrence methods have problems with these classes, too. For example, 4 subimages of class D12 seem to differ somewhat from the other images of that class causing problems in retrieval evaluation. Apart from these problematic classes, Blockordcooc seems to perform pretty well for both structural and stochastic textures, since according to Table 1 for none of the classes retrieval accuracy of Blockordcooc is lower than 60%, although all the competing methods, except Ordcoocmult, have average retrieval accuracies for some classes lower than that.

5.4. Comparison with other methods

For comparison purposes, retrieval results for the Test database 1 using different methods are provided in Table 1. Gabor wavelet features obtain slightly better average retrieval accuracy than Blockordcooc, but their feature extraction time is longer. For example, with PC (Intel Pentium 4, 3 GHz and 512 M RAM) feature extraction time for whole Test database 1 using Blockordcooc is 216 seconds, whereas the feature extraction time for Gabor wavelet feature and the same dataset is 5120 seconds. Both of these methods are implemented in C. Comparisons of feature extraction times for all of the methods are not provided, since some of the methods are implemented in Matlab and the comparison would not be fair. However, the computational complexity of local binary patterns with the defined parameter values is expected to be close to that of Blockordcooc. Generally Gabor features seem to perform well in retrieval; however, some problems occur, for example, with texture classes D101, D102, D10, D54, and D86. In majority of these classes texture has no clear dominant directionality.

Local binary pattern approach, LBP, performs the best among the other ordinal methods. The average retrieval accuracy of LBP is only 1.4% lower than that of Blockordcooc. However, the retrieval accuracy of LBP for couple of classes remains lower than 60%, whereas the average retrieval accuracy of Blockordcooc is better than that for each of the classes. Retrieval performance of gray level co-occurrence matrices seems to be the lowest among the evaluated methods. From Table 3 one can see that for ordinal and Gabor approaches, the average retrieval results remain the same for Test database 1 and Test database 2. This was expected since the tested ordinal methods are invariant to monotonic gray-level changes and Gabor filters are made insensitive to

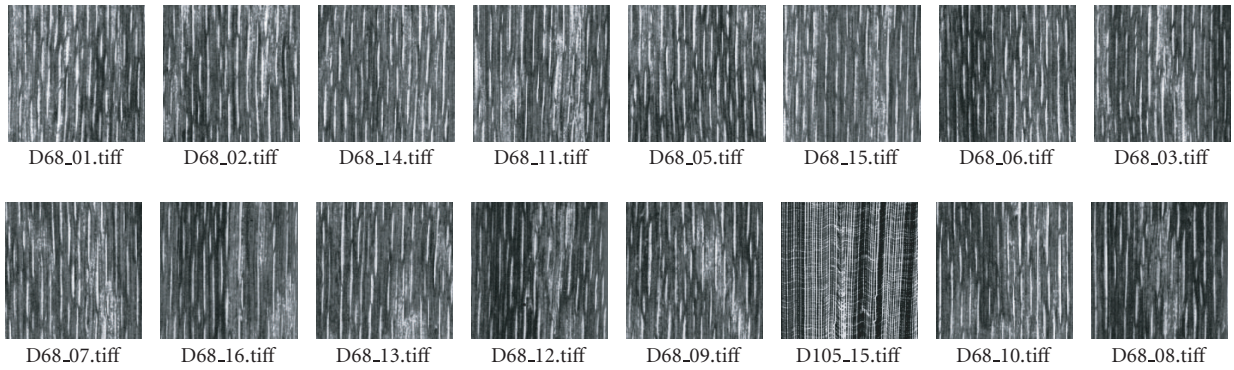


Figure 16: Sample query results (16 best matches) for Blockordcooc: D68_01 used as query, accuracy 93.75%.

absolute intensity changes by adding a constant to the real components of the filters [14]. However, retrieval results using GLCM features decrease somewhat since they are not invariant to gray-level changes.

6. CONCLUSION

In this paper we present a novel ordinal co-occurrence framework, into which different ordinal co-occurrence matrix approaches can be fitted and which can be used as a basis for new particularizations of the general framework. We also proposed and further analyzed and compared several ordinal co-occurrence methods as particularizations of this new framework. The proposed framework is intended to be flexible and therefore new variants can be derived. We also provided an extensive comparison of the ordinal methods derived from the framework with other ordinal methods. To demonstrate the performance of the ordinal methods we also compared their retrieval performance with a couple of other well-known methods for texture feature extraction which are not ordinal in nature. Due to its good average retrieval accuracy and low computational complexity, variant 3 of the proposed framework, Blockordcooc, was considered to be the best among the ordinal co-occurrence matrix approaches. It also outperformed the other evaluated ordinal methods and gray-level co-occurrence matrices. Average retrieval accuracy of Blockordcooc was almost as good as that of Gabor wavelet features, furthermore the computational cost of Blockordcooc is significantly lower. In addition Blockordcooc performed relatively well for all the classes, whereas other methods had relatively low accuracies for some of the evaluated texture classes.

Future work will focus on improving the proposed ordinal-based methods. Particularly, texture scale, if detected automatically, may restrict the maximum block size used and thus avoid the loss of important details.

ACKNOWLEDGMENTS

This work was supported by Graduate School in Electronics, Telecommunications and Automation (GETA), and the Academy of Finland, Project no. 213462 (Finnish Centre of Excellence Program 2006–2011).

REFERENCES

[1] D.-C. He and L. Wang, "Texture unit, texture spectrum, and texture analysis," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 28, no. 4, pp. 509–512, 1990.

[2] L. Hepplewhite and T. J. Stonham, "N-tuple texture recognition and the zero crossing sketch," *Electronics Letters*, vol. 33, no. 1, pp. 45–46, 1997.

[3] L. Hepplewhite and T. J. Stonham, "Texture classification using N-tuple pattern recognition," in *Proceedings of the 13th International Conference on Pattern Recognition (ICPR '96)*, vol. 4, pp. 159–163, Vienna, Austria, August 1996.

[4] T. Ojala, M. Pietikäinen, and T. Mäenpää, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 971–987, 2002.

[5] M. Partio, B. Cramariuc, and M. Gabbouj, "Block-based ordinal co-occurrence matrices for texture similarity evaluation," in *Proceedings of IEEE International Conference on Image Processing (ICIP '05)*, vol. 1, pp. 517–520, Genova, Italy, September 2005.

[6] M. Partio, B. Cramariuc, and M. Gabbouj, "Texture retrieval using ordinal co-occurrence features," in *Proceedings of the 6th Nordic Signal Processing Symposium (NORSIG '04)*, pp. 308–311, Espoo, Finland, June 2004.

[7] M. Partio, B. Cramariuc, and M. Gabbouj, "Texture similarity evaluation using ordinal co-occurrence," in *Proceedings of International Conference on Image Processing (ICIP '04)*, vol. 3, pp. 1537–1540, Singapore, October 2004.

[8] D. Patel and T. J. Stonham, "A single layer neural network for texture discrimination," in *Proceedings of IEEE International Symposium on Circuits and Systems*, vol. 5, pp. 2656–2660, Singapore, June 1991.

[9] D. Patel and T. J. Stonham, "Texture image classification and segmentation using RANK-order clustering," in *Proceedings of 11th International Conference on Pattern Recognition (ICPR '92)*, vol. 3, pp. 92–95, Hague, The Netherlands, August-September 1992.

[10] M. Pietikäinen, T. Ojala, and Z. Xu, "Rotation-invariant texture classification using feature distributions," *Pattern Recognition*, vol. 33, no. 1, pp. 43–52, 2000.

[11] P. Brodatz, *Textures: A Photographic Album for Artists and Designers*, Dover, New York, NY, USA, 1966.

[12] http://www.ux.uis.no/~tranden/brodatz.html.

[13] J. S. Weszka, C. R. Dyer, and A. Rosenfeld, "Comparative study of texture measures for terrain classification," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 6, no. 4, pp. 269–285, 1976.

[14] B. S. Manjunath and W. Y. Ma, "Texture features for browsing and retrieval of image data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 8, pp. 837–842, 1996.