A Novel and Robust way of Content Based Image watermarking: A case study

Mr. Aashish Joshi, Mr Virendra Banale
Department of E&TC Engineering,
Karmayogi Engineering College, shelve Pandharpur Dist –Solapur Maharashtra-INDIA-413304

Abstract: In order to endow the proposed scheme with content based Image watermarking (CBIW), information must be hidden region wise in digital image. The success of many content based approaches depends largely on the segmentation of the image based on its content. Here a region based approach to segmentation is presented. The segmentation of the image into regions is followed by the estimation of a set of region descriptors for each region; these serve as indexing information.

Keywords: image segmentation, watermarking, data hiding, image moment normalization, detection, image clustering, attacks.

1. INTRODUCTION

In watermark embedding process generally we observe the image frequency wise and then embed the watermark pixel in it. The watermark embedding is depends on the original image, we call this type of scheme as content-based image watermarking (CBIW).

In order to endow the proposed scheme with content based functionalities, information must be hidden region wise in digital images. Feature selection is one of the key problems to generate content based watermark so we use simple scheme to extract the watermark. Here we are using segmentation, indexing, filtering steps to embedding the watermark. Success of content based watermarking scheme is depending on segmentation of the image based on its content. Embedding segmentation and indexing information in image regions has the following advantages:

1. Each region in the image carries its own description and no additional information must be kept for its description;

2. The image can be moved from a database to another without the need to move any associated description;

2. SYSTEM OVERVIEW

The block diagram of the proposed system is shown in Figure 1. The system first segments an image into objects using a segmentation algorithm that forms only connected regions. The segmentation algorithm is applied to a reduced image consisting of the mean values of the pixel intensities in 8 × 8 blocks of the original image. Following Segmentation, watermarking can proceed immediately. Unlike segmentation, the watermarking process is applied to the full resolution image. Specifically, the segmentation information is embedded first. The indexing information is obtained from the reduced image and the indexing bits are channel coded and then embedded in the full resolution image.

2.1. Image Segmentation

In computer vision, segmentation refers to the process of partitioning a digital image...
Image segmentation is typically used to locate objects and boundaries (lines, curves, etc.) in images. The result of image segmentation is a set of segments that collectively cover the entire image, or a set of contours extracted from the image. Each of the pixels in a region is similar with respect to some characteristic or computed property, such as color, intensity, or texture. The overview of the segmentation algorithm is shown below in figure 2. Adjacent regions are significantly different with respect to the same characteristic(s).

The segmentation system described in this section is based on a variant of the popular K-means algorithm: the K-means-with-connectivity-constraint algorithm (KMCC).
The combination of intensity and texture information enables the algorithm to handle textured objects effectively, by forming large, chromatically non-uniform regions instead of breaking down the objects to a large number of chromatically uniform regions.

**KMMC (K-Mean Connectivity Constraint)** is commonly used algorithm which is explained as follows:

1) Classifies the pixels into regions taking into account
2) Intensity or texture information associated with each pixel
3) Position of the pixel, thus producing connected regions rather than sets of chromatically similar pixels.

The segmentation algorithm (KMMC) consists of the following stages:

**Stage 1.** Extraction of the intensity and texture feature vectors corresponding to each pixel. These will be used along with the spatial features in the following stages.

**Stage 2.** Estimation of the initial number of regions and their spatial, intensity, and texture centers of the KMCC algorithm.

**Stage 3.** Conditional filtering using a moving average filter.

**Stage 4.** Final pixel classification using the KMCC algorithm.

Image 1: Result of Few Segmented Images

2.2. **Image Reduction**

To reduce the size of an image, we split the image reduction algorithm in several primitive operations to ease explanation. The easiest way to reduce the image is by just sub-sampling its pixels. So if we want to reduce its size by 4 in each dimension, we just select one pixel at every four. To fully understand how this processing can be seen in the frequency domain. We first multiply the image by an impulse image, which the impulses located in the positions of the sampling which is equivalent of convolving their DFT's. We can now apply the image expansion DFT property.

2.3. **Image Indexing**

Indexing is an act of classifying and providing an index in order to make item (Pixel/Area) easier to retrieve. Objective of indexing as follows:

1. To stores data in a suitably abstracted and compressed form
2. In order to facilitate rapid processing by an application

2.4. **Initial Clustering**

Similarly to any other variant of the K-means algorithm, the KMCC algorithm requires initial values; in our case, an initial estimation is needed of the number of regions in the image and their spatial, intensity, and texture centers. A set of values chosen randomly could be used as initial values, since all these values can and are expected to be altered during the execution of the algorithm. Nevertheless, a well-chosen starting point can lead to a more
accurate representation of the objects of the image. In order to compute the appropriate initial values, the image is broken down to square, no overlapping blocks of dimension $f \times f$. In this way, a total of $L$ blocks, $bl$, $l = 1, \ldots, L$ are created. In our experiments, the value of $f$ was chosen so that the number $L$ of blocks created would be approximately 75; this was found to be a good compromise between the need for accuracy of the initial clustering, which improves as the number $L$ of blocks increases, and the need for its fast completion. The center of block $bl$ is pixel $pl$ cntr. A color feature vector $I(bl)$ and a texture feature vector $T(bl)$ are assigned to each block, as follows:

$$I(bl) = \frac{1}{f^2} \sum_{m=1}^{f^2} I(p^l_m)$$

$$T(bl) = T(p^l'_cntr)$$

where $pl$, $m, m = 1, \ldots, f \times f$ is the pixels belonging to block $bl$. The distance between two blocks is defined as follows:

$$D(b_{l_1}, b_{l_2}) = \| I(b_{l_1}) - I(b_{l_2}) \| + \| T(b_{l_1}) - T(b_{l_2}) \|,$$

Where

$$\| I(b_{l}) - I(b_{k}) \| = \sqrt{(I_{1}(b_{l})-I_{1}(b_{k}))^2 + (I_{2}(b_{l})-I_{2}(b_{k}))^2 + (I_{3}(b_{l})-I_{3}(b_{k}))^2},$$

$$\| T(b_{l}) - T(b_{k}) \| = \sqrt{\sum_{q=1}^{18} (T_q(b_{l})^h - T_q(b_{k})^h)^2}.$$}

The number of regions of the image is initially estimated by applying a variant of the maximin algorithm to this set of blocks. This algorithm consists of the following steps.

**Step 1.** The block in the upper left corner of the image is chosen to be the first intensity and texture center.

**Step 2.** For each block $bl$, $l = 1, \ldots, L$, the distance between $bl$ and the first center is calculated; the block for which the distance is maximized is chosen to be the second intensity and texture center. The distance $Db_{max}$ between the first two centers is indicative of the intensity and texture contrast of the particular image.

**Step 3.** For each block $bl$, the distances between $bl$ and all centers are calculated and the minimum of those distances is assigned to block $bl$. The block that was assigned the maximum of the distances assigned to blocks is a new candidate center.

**Step 4.** If the distance that was assigned to the candidate center is greater than $\gamma \cdot Db_{max}$, where $\gamma$ is a predefined parameter, the candidate center is accepted as a new center and Step 3 is repeated; otherwise, the candidate center is rejected and the maximin algorithm is terminated. In our experiments the value for $\gamma = 0.4$ was used.

The number of centers estimated by the maximin algorithm constitutes an estimate of the number of regions in the image. Nevertheless, it is not possible to determine whether these regions are connected or not. Furthermore, there is no information regarding their spatial centers. In order to solve these problems, a simple K-means algorithm is applied to the set of blocks, using the information produced by the maximin algorithm as initial values. The simple K-means algorithm consists of the following steps.

**Step 1.** The output of the maximin algorithm is used as a starting point, regarding the number of regions $sk, k = 1, \ldots, K$, and their
Step 2. For every block $b_l$, $l = 1, \ldots, L$, the distance is evaluated between $b_l$ and all region centers. The block $b_l$ is assigned to the region for which the distance is minimized.

Step 3. Region centers are recalculated, as the mean values of the intensity and texture vectors over the blocks belonging to the corresponding region:

$$I(s_k) = \frac{1}{M_k} \sum_{m=1}^{M_k} I(p^k_{m}),$$

$$T(s_k) = \frac{1}{M_k} \sum_{m=1}^{M_k} T(p^k_{m}),$$

where $b_{km}$, $m = 1, \ldots, M_k$ are the blocks currently assigned to region $s_k$.

Step 4. If the new centers are equal to those calculated in the previous iteration of the algorithm, then stop, else go to Step 2. When the K-means algorithm converges, the connectivity of the regions that were formed is evaluated; those that are not connected are easily broken down to the minimum number of connected regions using a recursive four-connectivity component labeling algorithm [17], so that a total of $K_\text{c}$ connected regions are identified.

Their centers, including their spatial centers $S(s_k) = [S_x(s_k), S_y(s_k)]$, $k = 1, \ldots, K_\text{c}$, will now be calculated. In order to obtain other useful information as well, such as the current size $M_k$ of each region in pixels, we choose to perform the center calculation process not in the block domain but in the pixel domain, as we will do during the execution of the KMCC algorithm. These centers will be used as initial values by the KMCC algorithm:

$$I(s_k) = \frac{1}{M_k} \sum_{m=1}^{M_k} I(p^k_{m}),$$

$$T(s_k) = \frac{1}{M_k} \sum_{m=1}^{M_k} T(p^k_{m}),$$

$$S_x(s_k) = \frac{1}{M_k} \sum_{m=1}^{M_k} p^k_{m,x},$$

$$S_y(s_k) = \frac{1}{M_k} \sum_{m=1}^{M_k} p^k_{m,y},$$

The output of the conditional filtering module can thus be expressed as:

2.5. Conditional Filtering

In some images, there are parts of the image where intensity fluctuations are particularly pronounced, even when all pixels in that part of the image belong to a single object (Image2). In order to facilitate the grouping of all these pixels in a single region based on their texture similarity, which is our objective, it would be of great importance to somehow reduce their intensity differences. This can be achieved by applying a moving average filter to the appropriate parts of the image, thus altering the intensity information of the corresponding pixels. The decision of whether the filter should be applied to a particular pixel $p$ or not is made by evaluating the norm of the texture feature vector $T(p)$ (see Section 3.2); the filter is not applied if that norm is below a threshold $T_\text{th}$. The output of the conditional filtering module can thus be expressed as:
Image 2: Original Image of Zebra and Filtered Image of Zebra

\[ J(p) = I(p) \quad \text{if} \quad \|T(p)\| < T_{th} \]
\[ = \frac{1}{f^2} \sum_{m=1}^{f^2} I(p_m) \quad \text{if} \quad \|T(p)\| > T_{th} \]

An appropriate value of the threshold \( T_{th} \) was experimentally found to be \( T_{th} = \max(0.65 \cdot T_{max}, 14) \) where \( T_{max} \) is the maximum value of the norm \( T(p) \) in the image. For computational efficiency purposes, the maximum of \( T(p) \) can be sought only among the pixels that served as block centers during the initial clustering. The term \( 0.65 \cdot T_{max} \) in the threshold definition is used to make sure that the filter will not be applied outside the borders of the textured objects. In this way, the boundaries of the textured objects will not be corrupted, thus enabling the KMCC algorithm to accurately detect those boundaries. The output of the conditional filtering stage will be now used by the KMCC algorithm.

2.6. Content Based Information Embedding

The information obtained for each image using the techniques of the preceding sections is embedded in the image itself. Two kinds of watermarks are embedded; one containing segmentation information and another that carries indexing information. Both are embedded in the spatial domain.

2.6.1. Segmentation Information Embedding:

The segmentation watermark consists of a number of symbols. A different symbol is embedded in each image region in order to make possible the identification of the regions during the watermark detection process. In a sense, the segmentation watermark symbols can be seen as labels that are tagged to each image object (see fig. 4).

We chose to embed the segmentation watermark in the Blue component of the RGB images because the Human Visual System is less sensitive to blue color [10]. Even though the methodology that will be developed is entirely general and an arbitrary number of regions may be labeled, we shall describe it here for the sake of brevity and simplicity for the case where only four regions of each segmented image need be labeled i.e. three main objects and the background. It will be assumed that any other objects determined using the segmentation algorithm are small and insignificant objects which are labeled using the same label as the background and therefore cannot be indexed and retrieved independently. The pixels of block \((l1; l2)\) in the Blue component \(IB\) of the image will be modified due to the watermarking process as follows:

\[ I'(l1, l2)_{B} = IB[l1+8l1+1, l2+j]+al1, l2*w(l1, j) \]

where \(l1; l2\) are the block indices, \(i, j\) are the indices specifying the position of the pixel inside the block and \(w[i, j]\) is the watermark matrix (see fig. 5), which is the same for all blocks in the image and is given by

\[ W(l1, j) = \begin{cases} 1 & \text{if } i+j=\text{even} \\ -1 & \text{if } i+j=\text{odd} \end{cases} \]
Fig No3: Example of Embedding labels to each image block
a $l_1,l_2$ is modulating factor valued as follows
- $a_{l_1,l_2} = -3$ if block $(l_1,l_2)$ belongs to the background
- $a_{l_1,l_2} = -1$ if block $(l_1,l_2)$ belongs to the 1st region
- $a_{l_1,l_2} = +1$ if block $(l_1,l_2)$ belongs to the 2nd region
- $a_{l_1,l_2} = +3$ if block $(l_1,l_2)$ belongs to the 3rd region

To detect the watermark, we calculate the correlation between the intensities $I_0(l_1,l_2)B[i,j]$ of the watermarked pixels in a block and $w[i,j]$. The detector output for block $(l_1;l_2)$ is calculated by

$$q_{l_1,l_2} = \frac{1}{N} \sum_i \sum_j I'(l_1,l_2)B[i,j] \cdot w[i,j],$$

where $N$ is the number of pixels in a block. In our case $N = 64$ since 64 pixels are included in a 8 × 8 block. The symbol that is extracted from each block depends on the detector output. The probability density function of the detector output can be approximated by a Gaussian distribution with mean equal to -3, -1, 1, or 3, depending on the symbol that was embedded. In this case, the optimal rule for extracting the label of block $(l_1;l_2)$ is

$$S=0 \quad \text{if} \quad q_{l_1,l_2} < -2$$
$$=1 \quad \text{if} \quad -2 < q_{l_1,l_2} < 0$$
$$=2 \quad \text{if} \quad 0 < q_{l_1,l_2} < 2$$
$$=3 \quad \text{if} \quad q_{l_1,l_2} > 2$$

since the above choice minimizes the probability of erroneous symbol detection. Although this is a small probability, there are some cases in which even such a small error could affect the synchronization capability of the system and the subsequent indexing information extraction. Such a case may occur if a block on region boundaries is misinterpreted. For this reason, immediately before embedding indexing information, a dummy detection of segmentation information takes place in order to identify blocks which yield ambiguous segmentation labels. In such blocks, no indexing information is embedded.

2.6.2. Indexing Information Embedding:
Indexing information is embedded in the Red component of each image using binary
symbols. For each region, eight feature values described by 8 bits each are ordered in a binary vector of 64 bits. Each bit of this vector is embedded in a block of the corresponding region. After the embedding of the watermark, the Red component of the block \((l_1; l_2)\) of the image is as follows:

\[ I'(l_1, l_2)(i, j) = IR[l_1+1, 8l_2+j] + a_{l_1,l_2} \cdot w(i, j) \]

Where \(w\) is the watermark matrix given in eq. (8) and \(a_{l_1,l_2}\) is a modulating factor valued as follows:

\[ a_{l_1,l_2} = 1, \text{ if the embedded bit is 1} \]
\[ = -1, \text{ if the embedded bit is 0} \]

The Green component is not altered. The detection is correlation-based i.e.

\[ q_{l_1,l_2} = \frac{1}{N} \sum_i \sum_j I^R_{(l_1, l_2)}[i, j] \cdot w[i, j]. \]

If the output \(q\) of the detector is less than zero then the extracted bit is 0, otherwise 1. Using this rule, the resulting probability of erroneous detection is very small. However, in order to achieve lossless extraction, error correcting codes can be used. Error correcting codes can detect and correct errors that may occur during the extraction of the embedded bit stream. In this paper, a simple Hamming code is used that adds three error control bits \(BC1, BC2, BC3\) for every four information bits \(BI1, BI2, BI3, BI4\). Thus, the embedding bit stream takes the form \(BC1, BI2, BC3, BI4\) for every four indexing bits. If only a single error occurs while detecting the four indexing bits, the error can be corrected. The protection achieved using this approach is so strong (for the given application) that practically guarantees the correct extraction of all indexing bits.

2.7. Watermark Detection
Conversely, the first step in the watermark detection process is to detect the segmentation watermark and subsequently, based on this segmentation, to extract the information bits associated with each object (see Fig. No 4). If, due to unsuccessful watermark detection, the segmentation mask detected at the decoder is different than the one used at the encoder, then the detection process will not be synchronized with the embedding process and the embedded indexing information will not be retrieved correctly. To alleviate this problem, a dummy detection of the embedded segmentation information takes place at the encoder prior to the embedding of any indexing information.

3. EXPERIMENTAL RESULTS

To check the robustness of the watermarking algorithm, we applied some of the attacks as summarized in the table below.

**TABLE NO1**

RESPONSE OF FEW IMAGES FOR APPLIED ATTACKS

<table>
<thead>
<tr>
<th>Parameter Name of Image</th>
<th>PSNR</th>
<th>IF</th>
<th>BER Original</th>
<th>BER LD</th>
<th>MESE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hilly Region</td>
<td>34.26</td>
<td>0.999</td>
<td>4.957</td>
<td>9.52</td>
<td>0.2098</td>
</tr>
<tr>
<td>Zebra</td>
<td>34.87</td>
<td>0.999</td>
<td>6.02</td>
<td>17.27</td>
<td>0.2478</td>
</tr>
<tr>
<td>Lena</td>
<td>32.37</td>
<td>0.999</td>
<td>5.74</td>
<td>13.72</td>
<td>0.255</td>
</tr>
</tbody>
</table>

**Acronyms:**
- BER= Bit Error Rate
- PSNR=Peak Signal to Noise Ratio
- MSE=Mean Square Error
- IF=Image Fidelity
- LD=Line Deletion

**Watermark Evaluation**

- **PSNR:** (Peak Signal to Noise Ratio) It is the ratio of maximum possible power of corrupting noise that affects fidelity of its representation.
  \[ \text{PSNR} = 20 \log_{10}(\text{MAX}^2/\text{MSE}) \]
- **NCC** (Normalized Correlation Coefficient): It shows the comparison between two images. Following MATLAB functions are used to calculate correlation.
  - \( R = \text{corr2}(a,b) \) ------- 2D correlation coefficient.
  - \( R = \text{corrcoef}(x,y) \) ------- correlation coefficient.
4.0 IMAGE MOMENT NORMALIZATION

The source image is normalized based on its central moments. Moment normalization is much a useful technique as the moments of an image is used to describe its contents with respect to its axes. Moments can be used to characterize images and to express properties that have analogy in statistics. Moment Normalization is done mainly to resist geometrical attacks.

Steps of Image moment Normalization:
1) Compute the centroid of image I
Xbar = M10/M00
Ybar = M01/M00

Where Mi,j defined as
Mi,j = ΣΣx^i*y^j*I(x,y)

2) Compute the central moments
μ00 = ΣΣI(x,y)
μ20 = ΣΣx^2*I(x,y)
μ11 = ΣΣx*y*I(x,y)

3) Compute the co-variance matrix for moments
μ20  μ11
μ11  μ02

the matrix is represented as CoV

(4) Compute the Eigen vectors of CoV
elx  ely
-ely  elx

And the Eigen value of CoV
λi= ½ *(μ20+μ20)±√(4μ11²+(μ20-μ02)²)

5) Compute the orientation angle
θ=1/2 * tan^-1 (2μ11/(μ20-μ02))

6) Compute the rotation matrix as
Cosθ  sin θ
-sinθ  cosθ

7) Compute the scaling matrix S
S = (λ1λ2)^.25/√λ1 0
0 (λ1λ2)^.25/√λ2

8) The translation matrix T is eigenvector of
CoV

9) Construct the moment normalized image
Im=R S T *I(X,Y)

Thus the source image is moment normalized, so that it can withstand affine transformation attacks. The result of image moment normalization is explained by fig as shown below.

Image4: Original Image & Normalized Image

As we explained above after the proper application of image moment normalization we need to do the inverse process of it called Inverse image moment normalization. The result of which is explained as below.
5.0. APPLICATIONS
The watermarking applications depend on type of techniques used for the implementation. Here we are concentrating on Content based approach. This approach has numerous applications, some of which are explained below.

1. Validating identity cards, debit and credit cards, voter ID cards, driving licenses and employee identity cards.
2. Authenticating financial instruments such as fixed deposit receipts and financial stocks.

6.0 CONCLUSION
Methodology presented here for the segmentation and content-based embedding of indexing information in digital images. The segmentation algorithm combines pixel position, intensity, and texture information in order to segment the image into a number of regions. Two types of watermarks are subsequently embedded in each region: a segmentation watermark and an indexing watermark.

The proposed system is appropriate for building flexible databases in which no side information is needed to be kept for each image. Moreover, the semantic regions comprising each image can be easily extracted using the segmentation watermark detection procedure.

7.0. FUTURE SCOPE
- Currently very few algorithms exist with content Based Watermarking and Image Moment Normalization.
- Work on collaboration
- Developing robust technique for Content Based Image Watermarking.

8.0. REFERENCES
[3] ZHOU Haitao, QI Chun, GAO Xiaochuan” Low Luminance Smooth Blocks Based Watermarking Scheme in DCT Domain” 07803-9584-0/06/$20.00©2006 IEEE.
[6] Segmentation and Content-Based Watermarking for Color Image and Image
[7] ROBUST CONTENT
BASED IMAGE
WATERMARKING Selena
Kay and Ebroul Izquierdo
Multimedia&Vision
Research Lab, Department of
Electronic Engineering, Queen Mary,
University of London, London, U.K.

[8] A Content Based Image
Watermarking Scheme
Resilient to Geometric
Attacks - Latha
Parameswaran, and K.
Anbumani

[9] A Robust Image
Watermarking Scheme using
Image Moment
Normalization - Latha
Parameswaran, and K.
Anbumani

[10] Image Segmentation:
EE4H, M.Sc 0407191
Computer Vision - Dr. Mike
Spann

for Multimedia Security - R.
Chandramouli, MSyNC: Multi
media Systems, Networking,
and Communications Lab,
Stevens Institute of
Technology.

[12] Primary Image
Segmentation Ullrich Köthe
Fraunhofer-Institut für
Graphische