Development and Implementation of a Video Watermarking Method Based on DCT Transform

Ali Benziane¹, Suryanti Awang², and Mohamed Lebcir²
¹Faculty of Science and Technology, University of Djelfa, Algeria
²Faculty of Computing, Universiti Malaysia Pahang, Malaysia

Abstract: This paper presents a new color video watermarking technique based on the one-dimensional Discrete Cosine Transform (DCT). This approach uses a differential embedding technique to insert the bits of the watermark into the video frames so that the extraction process is blind and straightforward. To further ensure the security of the method, the binary image watermark is scrambled using Arnold transform before embedded into the video segment. Also, a color space transformation from Red, Green and Blue (RGB) to YUV is performed in order to deal with the color nature of the video segments. The proposed approach exhibits good robustness against a wide range of attacks such as video compression, cropping, Gaussian filtering, and noise adding. Finally, we propose an implementation of the video watermarking technique using the Raspberry Pi 3 platform. Nearly the same remarks may be made as in the simulation results concerning the robustness against video compression attacks.

Keywords: Blind video watermarking, DCT, differential embedding, Raspberry Pi.

Received May 1, 2019; accepted April 8, 2020
https://doi.org/10.34028/iajit/18/2/2

1. Introduction

The recent advances in mobile communication technologies, video sharing websites, and social networks, have made it easier to share and distribute any content [1]. To some purposes, users able to edit and re-distribute digital multimedia contents such as images, audios and videos [3]. Digital watermarking has become a promising technology for protecting digital contents from unauthorized copying and manipulation by embedding a secret information directly into the data itself [7, 14]. Basic requirements of digital watermarking are transparency, robustness, and capacity [25]. These three requirements of watermarking are conflicting and limited by each other; improvement in any one of them, affects the other two negatively [14].

The watermark can be inserted in either the spatial or the transformed domain [12]. The transform domain schemes, which are usually more robust than the spatial domain ones, have attracted much attention in the literature [1, 7]. Examples of the transformed domains are the Discrete Fourier Transform (DFT) [15] the Discrete Cosine Transform (DCT) [9], the Discrete Wavelet Transforms (DWT) [13] and the combined transform domain [6, 11].

Digital watermarking can also be blind or non-blind based on extracting process [24]. In contrast to non-blind detection, blind detection schemes do not need the original media content to recover the inserted watermarks, and hence more practical in real application [14].

The process of video watermarking exhibits more difficulties than image watermarking and is more time consuming, because video segments contains large amount of data compared to digital images [1].

Recently, transform-based video watermarking schemes have attracted a lot of attention especially those based on DCT and DWT [14]. Li et al. [16], the embed the watermarking information into the last DC coefficient of the last macro block in every slice of the luminance component of the Moving Pictures Expert Group v2 (MPEG2) transport stream. Their results showed good performance especially in terms of time complexity, but the overall method is strictly limited to MPEG2 coding.

Abdi et al. [1], have developed a watermarking scheme for H.264 video. The watermark is embedded into the video sequence by modifying the number of non-zero-quantized AC coefficients in a 4×4 block of I frames. Their scheme showed acceptable performance in terms of robustness and payload for this video compression standard.

Preda and Vizireanu [18] proposed a video watermarking method based on multi-resolution wavelet Transform. In their method, the wavelet coefficients of the second level LH, HL and HH sub-bands, are used to embed the binary image watermark by means of a quantization process.

Wang et al. [23] developed a robust and real-time video watermarking algorithm for MPEG-2 compressed video. A set of histogram bins deduced from the DWT low frequency sub-band, are used to embed the binary watermark. Their scheme is
especially robust against geometric distortions such as
cropping, rotation, scaling, and frame dropping.

Farfourea et al. [10] the authors proposed a semi-
fragile watermarking method for the authentication of
the integrity of H.264 compressed videos. The Watermark
insertion is performed by flipping the signs of
nonzero DCT coefficients of candidate pairs of
certain DCT-transformed blocks. The embedded video
frames are selected using a spatial analysis to ensure
invisibility and robustness. Their technique exhibits
high robustness against content-preserving attacks but
it has high sensitivity against content-changing ones.

In this paper, a new blind color video watermarking
technique using 1D-DCT transform is proposed and
evaluated. The watermark bits are embedded into two
1D-DCT transformed sub-vectors issued from the sub-
sampling the Y channel of the YUV space of the video
frames. The simple difference between the corresponding sub-vectors of the watermarked frames,
leads to the complete extraction of the watermark
image.

Also, we examine the possibility of implementing
the proposed video watermarking technique using a
real hardware device which is the Raspberry Pi 3. In
order to further increase the security of the implemented scheme [4], a biometric feature
(fingerprint image) is used as a digital watermark [24].

The paper is organized as follows: section 2 gives a
background of the used color space which is YUV. In
section 3, wedescribe the general watermarking
process of the video segment. The embedding and the
extracting processes of the binary watermark at the frame level are detailed in section 4. Section 5 presents
the simulation results and analysis. Section 6 discusses
a possible implementation of the proposed method
using Raspberry Pi, while section 7 concludes the
paper.

2. YUV Color Model

The first color system used for video transmission was
the YIQ system developed for National Television
System Committee (NTSC) video TV broadcasting
and the closely linked YUV standard developed for
Phase Alternating Line (PAL) [21]. In both cases, it
was intended to have a luma (for luminance) Y
channel that would be similar to the regular black and
white TV signal, together with two lower frequency
color channels (I,Q) or (U,V) [21].

The conversion of the Red, Green and Blue (RGB)
color system to YUV color system could be per-
formed using the following system of equations [22]:

\[
\begin{bmatrix}
  Y \\
  U \\
  V \\
\end{bmatrix} = \begin{bmatrix}
  0.299 & 0.587 & 0.114 \\
  -0.147 & -0.289 & 0.436 \\
  0.615 & -0.515 & -0.100 \\
\end{bmatrix} \begin{bmatrix}
  R \\
  G \\
  B \\
\end{bmatrix}
\]

Likewise, the RGB channels could be restored back
from YUV using the following system of equations
[22]:

\[
\begin{bmatrix}
  R \\
  G \\
  B \\
\end{bmatrix} = \begin{bmatrix}
  1 & 0 & 1.40 \\
  1 & -0.395 & -0.581 \\
  1 & 2.032 & 0 \\
\end{bmatrix} \begin{bmatrix}
  Y \\
  U \\
  V \\
\end{bmatrix}
\]

In this paper, we have used the Y channel of the YUV
color system to embed the watermark in order to
facilitate the implementation and to ensure the
robustness of the extraction procedure [6].

3. The General Embedding and Extracting
Process

The proposed video watermarking method is an
extended version of the image watermarking method
proposed in [5]. It comprises different modules such as
watermark pre-processing (scrambling), video pre-
processing (frame selection and color space
conversion), watermark embedding, and watermark
extraction. The general embedding process is shown in
Figure 1 and it includes the following steps:

- **Step 1:** Get an uncompressed video segment and
  convert it into frames.
- **Step 2:** Select a random set of frames to insert the
  watermark in, using a secret key \(k_i\). The number
  of the selected frames must be equal to the number
  of columns of the binary image watermark \(W\).
- **Step 3:** Scramble the binary watermark image using
  Arnold transform [20] and the secret key \(k_3\). Given
  a \(H\times H\) square image, one of its discretized versions
  [13] is given as follows:

\[
\begin{bmatrix}
  x_{il} \\
  y_{il} \\
\end{bmatrix} = \begin{bmatrix}
  1 & a \\
  b & ab + 1 \\
\end{bmatrix} \begin{bmatrix}
  x_i \\
  y_i \\
\end{bmatrix} \mod(H)
\]

Where \(a\) and \(b\) are the Arnold transform parameters
that can be defined by the system’s user to encrypt the
watermark image. So the secret key \(k_3\) is composed
of three parameters: \(a, b\) and the number of iteration \(i\).
- **Step 4:** Embed each column of the scrambled
  watermark image in the Y channel of the YUV
  space representation of the selected frame (Step 2)
  using the differential embedding method described in
  [5].
• **Step 5**: Concatenate all frames (watermarked and not) into one watermarked video segment.

On the other hand, the process of extracting the binary watermark image from the watermarked (and possibly attacked) video segment is inversely analogous to the embedding process as depicted in Figure 2.

![Figure 2. The general video extracting process.](image2)

- **Step 1**: Using the secret key \( (k_1) \), select the set of frames that have been used in the watermark embedding phase.
- **Step 2**: Extract each column of the scrambled watermark image from the selected frame using the extracting process described in [5]. The concatenation of all extracted columns constitutes the scrambled watermark \( W_s \).
- **Step 3**: De-scramble the scrambled watermark image \( W_s \) using the inverse Arnold transform and the secret key \( (k_3) \) to produce the original binary watermark image \( W \).

### 4. Proposed DCT-based Video Watermarking Method

The basic process of this frame embedding technique is given in Figure 3 and it includes the following sub-steps:

1. Convert the input frame from RGB to YUV (the luminance Y and two chrominance components U and V) color space and get the intensity Y channel matrix of size \( M \times N \) using Equation (1).
2. Perform zigzag scanning on the Y channel matrix to convert it into one vector \( z \).
3. Generate two sub-vectors \( z_1 \) and \( z_2 \) from the vector \( y \) using the following sub-sampling operation:

   \[
   z_i(k) = z(2k) \quad (4) \\
   z_i(k) = z(2k-1) \quad (5)
   \]

   Where \( k=1,...,M \times N \).

4. Perform DCT on \( z_1 \) and \( z_2 \) to produce their DCT-transformed versions \( Z_1 \) and \( Z_2 \).
5. Insert the vector of the watermark bits (which is one column of \( W \)) in random locations (chosen using a secret key \( k_2 \)) to produce the original binary watermark \( W \) as shown in Figure 4.

![Figure 3. The embedding process for video frames.](image3)

Notice that if the input frame is a watermarked one, and by analogy with the embedding process, the extraction process will produce one watermark vector as shown in Figure 4.

![Figure 4. The extraction process of the DCT method.](image4)

### 5. Experimental Results

The proposed algorithm is evaluated for five Audio Video Interleaved (AVI) format uncompressed video sequences\(^1\) of size 300 frames of 352x288: ‘City’, ‘Coastguard’, ‘Crew’, ‘Foreman’, and ‘Soccer’. The chosen watermark is a binary image of size 100x100 shown in Figure 5. Also, we set \( \alpha = 0.3 \) as the default gain factor value as suggested in [5].

\(^1\)https://media.xiph.org/video/derf/
To assess the quality of watermarked video segments, the average PSNR measure is used [19]:

$$PSNR = 10 \times \log_{10} \left( \frac{3 \times 255^2}{MSE + MSE_G + MSE_B} \right)$$

(8)

$$MSE = \frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} (u(i, j) - \hat{u}(i, j))^2$$

(9)

Where $u(i, j)$ and $\hat{u}(i, j)$ are respectively, the original and the watermarked intensities belonging to R, G and B planes of the video frame. Moreover, the Bit Correct Rate (BCR) is chosen to assess the similarity between extracted and the original watermarks [5]:

$$BCR = \frac{1}{M \times N} \sum_{j=0}^{M-1} \sum_{k=0}^{N-1} W(j, k) \oplus \hat{W}(j, k) \times 100\%$$

(10)

Where $\hat{W}$ and $W$ are respectively the extracted and the original watermarks of sizes $M \times N$ and $@$ is the binary XOR operator.

Figure 6 shows original video frames as well as the watermarked ones for the five test video segments. It is obvious that the quality of the watermarked frames is preserved qualitatively and quantitatively.

### 5.1. Robustness Test

#### 5.1.1. Robustness Against Common Attacks

The robustness of the proposed video watermarking method against noise addition, Gaussian filtering, Gamma correction, sharpening and cropping is evaluated in Tables 1 and 2. Table 3 shows the extracted watermarks after different types of high-strength attacks for the video segment of ‘soccer’. It’s obvious that the suggested DCT-based video watermarking method is robust against all the above attacks even for high intensity ones like the case of Gaussian noise of high variance.

Table 1. Robustness of the proposed video watermarking technique against noise adding attacks.

<table>
<thead>
<tr>
<th>Attack</th>
<th>Video segment</th>
<th>city</th>
<th>coastguard</th>
<th>crew</th>
<th>foreman</th>
<th>soccer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gaussian noise (var=0.01)</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Gaussian noise (var=0.02)</td>
<td>99.8</td>
<td>99.8</td>
<td>99.8</td>
<td>99.7</td>
<td>99.8</td>
<td></td>
</tr>
<tr>
<td>Gaussian noise (var=0.03)</td>
<td>99.3</td>
<td>99.0</td>
<td>99.3</td>
<td>99.0</td>
<td>99.1</td>
<td></td>
</tr>
<tr>
<td>Salt &amp; pepper noise (var=0.01)</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Salt &amp; pepper noise (var=0.02)</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Salt &amp; pepper noise (var=0.03)</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 2. Robustness of the proposed video watermarking technique against different types of attacks.

<table>
<thead>
<tr>
<th>Attack</th>
<th>Video segment</th>
<th>city</th>
<th>coastguard</th>
<th>crew</th>
<th>foreman</th>
<th>soccer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gaussian filter (5x5) var=1</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Gaussian filter (3x3) var=1</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Gamma correction (2)</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Laplacian sharpening</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Surrounding crop (10%)</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Surrounding crop (25%)</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>
Table 3. The extracted watermark under different types of attacks.

<table>
<thead>
<tr>
<th>Attack Type</th>
<th>BCR (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gaussian noise (var =0.1)</td>
<td>BCR=90.25 %</td>
</tr>
<tr>
<td>Motion JPEG (Q=40)</td>
<td>BCR=90.01 %</td>
</tr>
<tr>
<td>Cropping (30%)</td>
<td>BCR=99.1 %</td>
</tr>
<tr>
<td>MPEG4-H.264 (Q=70)</td>
<td>BCR=84.1 %</td>
</tr>
<tr>
<td>Motion JPEG2000 (R=50)</td>
<td>BCR=89.0 %</td>
</tr>
</tbody>
</table>

5.1.2. Robustness Against Compression Attacks

To assess the robustness against video compression, the ‘Motion JPEG’, ‘Motion JPEG2000’, and MPEG-4 ‘H.264’ compression standards are used. The plot of the results of robustness against these attacks is shown in Figures 7, 8, and 9 respectively.

One can see clearly from these results that the presented watermarking algorithm exhibits very good performance against motion JPEG and JPEG2000 compression standards. Differently, it shows a lower performance against MPEG-4 attacks due to the very high compression rate of this standard.

6. Implementation Using Raspberry Pi

In this section, we deal with the implementation of the proposed video watermarking technique using Raspberry Pi (RP) platform [2].

Instead of using an arbitrary binary watermark one can effectively use a fingerprint image (of size 200×200) issued from a fingerprint scanner to watermark the video segments. Therefore, the watermarking process will be much more secure because the fingerprint patterns are highly person-dependent [2].

Figure 10. The proposed implementation of the video watermarking system.

- The zig-zag scanning in Figures 3 and 4 is replaced by a simple column-wise concatenation of each
frame in order to transform the channel Y (resulted from RGB to YUV conversion) into one vector. This modification is needed in order to enhance the speed of the video watermarking process on RP. This will affect slightly the transparency of the watermarked video segment, because of the low correlation between the sub-vectors Z₁ and Z₂ and (section 3) in this case, but this drawback can be compensated by embedding fewer number of bits in each frame.

- The watermark which is a fingerprint image has to be binarized (Figure 11) before embedding into the video segment. A hard thresholding binarization with a threshold of 127 is used in the experiments. After that, the general embedding process as in Figure 1 is performed by the RP platform using the new binary watermark.

### 6.2. Implementation Performance

In the experiments on the implemented system, video segments of 300 frames are captured (by the camera module) with 288×252 resolution at 30 frames per second in order to be compatible with those used in the general DCT-based algorithm.

To verify the robustness of the implemented algorithm, we concentrate only on following video compression attacks: Motion JPEG (MJPEG), MPEG-4, and H.264 standards because of their availability on the RP operating system. Notice that the PSNR of the watermarked video are preserved through the embedding factor $\alpha$ to be slightly more than 35dB.

#### 6.2.1. Execution Speed

The processing time per function for the watermark embedding and extracting (given by the Python Profilers module) are shown by Tables 4 and 5 respectively.

<table>
<thead>
<tr>
<th>Function</th>
<th>frm_extract</th>
<th>vid_extract</th>
<th>dct</th>
<th>idct</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total time (s)</td>
<td>6.34</td>
<td>27.32</td>
<td>6.03</td>
<td>5.75</td>
</tr>
<tr>
<td>N of calls</td>
<td>200</td>
<td>1</td>
<td>400</td>
<td>400</td>
</tr>
<tr>
<td>Time per call (s)</td>
<td>0.03</td>
<td>27.32</td>
<td>0.01</td>
<td>0.01</td>
</tr>
</tbody>
</table>

The watermark embedding at the frame level (embedding one column of the watermark image) takes only 0.03s which is acceptable. The same remark can be made regarding the watermark extracting at the frame level. Also, the most time-consuming functions at the watermark embedding and extracting phases are the DCT and the inverse DCT which is consistent with the features of transform domain watermarking techniques.

Note that the total watermark embedding and extracting times (27.32s and 11.73s respectively) can be reduced significantly by using FPGA-based system instead of Raspberry Pi but at the expense of the implementation complexity [12].

#### 6.2.2. Robustness Against Compression

Tables 6 and 7 shows respectively the BCR of the extracted watermark for MJPEG and MPEG4 compression standards of three video segments and for different quality scale. Table 8 shows the performance of the implemented method against the more recent and size-efficient type of video compression, i.e, the H.264 standard. In this type of compression the Constant Rate Factor (CRF) is used as quality assessment measure [8].

<table>
<thead>
<tr>
<th>Quality (Qscal)</th>
<th>Segment 1</th>
<th>Segment 2</th>
<th>Segment 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Quality (Qscal)</th>
<th>Segment 1</th>
<th>Segment 2</th>
<th>Segment 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CRF</th>
<th>Segment 1</th>
<th>Segment 2</th>
<th>Segment 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>10</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>15</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>20</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

These results confirm the observations made in section 4 about the robustness of the proposed method against video compression even with the usage of column wise concatenation instead of the zigzag operation. The robustness against Motion JPEG and MPEG4-4 standards is high for the acceptable range of the quality factor (Qscal) used in the implementation. For the H264 standard, the behavior of the implemented method is also similar to the simulation.
results given in section 4; it has very good performance under low compression ratios but it gets
limited under high to severe ones as shown in Table 9.

Table 9. The extracted watermark under H.264 compression attacks.

<table>
<thead>
<tr>
<th>CRF</th>
<th>BCR</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>96%</td>
</tr>
<tr>
<td>22</td>
<td>86%</td>
</tr>
<tr>
<td>24</td>
<td>77%</td>
</tr>
<tr>
<td>26</td>
<td>75%</td>
</tr>
</tbody>
</table>

7. Conclusions

In this paper we have designed and implemented a blind and robust watermarking technique for uncompr essed video using the DCT transform. The technique is based on the differential embedding of the bits of the watermark in DCT domain. The perceptual quality of the watermarked video segments and the average PSNR values have been very good for all tested video segments. Different types of attacks have been performed on the watermarked video segments and the corresponding BCR values have shown that the proposed technique fairly fulfills the robustness requirements.

Furthermore, the Raspberry Pi 3 platform is employed to implement a reduced version of the proposed video watermarking method. In this implementation, the user fingerprint image is used as the binary watermark and the zig-zag operation is replaced with column-wise concatenation in order to accelerate the embedding and extracting processes. The experimental results have validated those found in the simulation and have proved the feasibility of the implementation on the Raspberry Pi platform.

Acknowledgments

We would like to show our gratitude to Universiti Malaysia Pahang (RDU vote number RDU190315) for supporting this study.

References


Ali Benziane received a Ph. D degree in Electronics in 2017 from the University of Setif, Algeria. He is presently working as an Assistant Professor at the Faculty of Science and Technology, University of Djelfa, Algeria. His field of interest includes Image Processing, Embedded Systems and IoT.

Suryanti Awang received Ph. D degree in Electrical Engineering in 2014 from UniversitiTeknologi Malaysia, Johor Bahru, Malaysia. Dr. Suryanti is currently a senior lecturer at Universiti Malaysia Pahang, Malaysia from 2005 until now. Her research interests include Pattern Recognition, Machine Learning, and Soft Computing.

Mohamed Lebcir received a master's degree in Computer Science (Option: Image and Speech processing) in 2015 from University of Djalfa, Algeria. He is currently a PhD Student at Universiti Malaysia Pahang, Malaysia. His research interests include Pattern Recognition and Image and Audio Processing.