# Splitting the watermark up, a steganographic watermarking algorithm for H.264 with reduced image distortion

JAN VAN ZWOL, University of Twente, The Netherlands

In the current digital era it is very easy to download, alter and share media. That is why it has become more important to be able to assert ownership over ones intellectual property rights. A solution for indicating ownership over digital media could be steganographic watermarking. This is embedding an invisible watermark in a digital object. Research has been done on watermarking methods, mainly on images. That is why in the field of video watermarking most methods watermark every frame of a video separately as if it were a collection of unrelated images. In this research an algorithm was implemented and tested, that uses the temporal dimension of the video and distributes parts of the watermark over the frames in a video. Watermarking a video in this way is novel and had some advantages in terms of image distortion. The new watermarking algorithm was tested against an algorithm which does not distribute the watermark. Both algorithms were on based on the algorithm described in Liu (et al.,2015), a DCT based watermarking algorithm that has yielded good results in terms of extraction and robustness. Both of these algorithms were implemented in a Python environment. The algorithms embedded an 80x80 pixel black and white watermark in the video files. The watermark was embedded in video files with a 1280x720 resolution made by the researcher. The image distortion caused by both algorithms was measured and compared and the algorithms were tested on robustness against attacks like: frame deletion, cropping, brightness change, compression. The results of the tests were used to determine what the advantages and disadvantages of the distributed watermark algorithm are.

Additional Key Words and Phrases: Video Watermarking, Video Steganography, Distributed Watermark, Image Distortion, Robustness

## 1 INTRODUCTION

The modern internet has made it so that it is easier than ever to acquire, alter and share media. This has brought forth a huge improvement in the flow of ideas and information[5]. However, it has made it harder to protect the intellectual property rights of the owners of these rights [9].

To make it easier to protect these rights a watermark can be inserted in a piece of media using steganographic techniques. These techniques are used to embed information in a digital object, often an image. The hidden information should not be noticeable when looking at the *stego image* (this refers to the image in which the information is hidden) but can be extracted afterwards with an algorithm. This hidden information can be used to indicates ones ownership over the IP rights (*intellectual property rights*) of the *cover image* (this refers to image which is used to embed information in through steganographic techniques) [17].

Even though the field of steganography mainly focuses on images [22] it can be used for a variety of cover objects amongst which are video [15]. Video just like any form of expression is protected by IP rights [24] and could thus benefit form having a steganographic watermark. Since videos consist of frames which are images shown rapidly after one another it should be possible to make use of the techniques developed for image steganography in the application of watermarks for video files.

In the current literature when watermarks are applied to video files these watermarks are embedded entirely in each frame [17]. So in essence the video is seen as a collection of frames and each frame has it's own watermark. The upside of doing this is that you can indicate ownership over every single frame. However it does seem to be computationally inefficient as each frame of the video has to be separately watermarked. This will also have a negative effect on the quality of the stego image and the resistance to *steganalysis* (an algorithm that predicts whether an object has a steganographic message hidden in it) as the more data is hidden in the stego object the more it will differ from the cover object [7, 8].

Instead of watermarking every frame separately, a watermarking algorithm can also make use of the fact that a video is a series of frames. The algorithm can make use of the temporal dimension of the video to embed a watermark in the video by splitting a watermark up into smaller pieces and distributing these parts over the frames of a video. This means that each frame will have less bits embedded in it and this might cause less distortion to the image. However, this will make the distributed watermark algorithm vulnerable to attacks like frame deletion and insertion while a non-distributing algorithm should be robust against it as in embeds the entirety of the watermark in every frame

There is not a lot of research on video watermarking technique that use the temporal dimension of a video [17] even though it could improve the watermark's reduce the distortion in the video file and be computationally more efficient than frame-by-frame watermarking. Due to these effects watermarking videos using the temporal dimensions could have significant advantages and is worth researching.

In this research the advantages and disadvantages of using a *distributed watermark algorithm* (A steganographic watermarking algorithm that splits the watermark up and embeds only parts of it in each frame) are measured against a *non-distributing algorithm* (A steganographic watermarking algorithm that embeds the entirety of the watermark in each frame). A distributed watermark algorithm was proposed and implemented and tested against a non-distributing algorithm in terms of image distortion and robustness.

## 2 RELATED WORK

Like already said until now the main focus of research on steganography and steganographic watermarking has been using images as cover objects [22]. These techniques can also be useful for video watermarking. However, some alteration have to be made in order

Author's address: Jan van Zwol, j.vanzwol@student.utwente.nl, University of Twente, P.O. Box 217, Enschede, The Netherlands, 7500AE.

to make it work for videos [17]. Especially if the temporal dimension of the video file is going to be utilized.

There are two main categories of watermarks: fragile and robust [17]. A fragile watermark is not resistant to attacks, meaning alterations to the stego object, while a robust watermark is (to some extent) resistant to attacks. A fragile watermark can be used to analyse whether a digital object has been altered however if the purpose of a watermark is to indicate ownership over the object a robust watermark should be used since indication of ownership should still be present even after alterations to the cover object.

There are two domains in which the watermark-information can be embedded. The spatial domain and the frequency domain [17]. Making use of the frequency domain yields better results in terms of robustness and data hiding capacity [12] so that was the domain used in this research. Embedding information in the frequency domain can be done in multiple ways, two of the main ones being DCT and DWT [17] in this research DCT was used. DCT is used in the compression of video files. In steganography it is used to hide information. For this mainly the middle frequency band is used because this will make the presence of the watermark the least noticeable to a viewer [3].

Video steganography is not the main focus of the steganographic research field but still some research on it has been done [7, 10, 15, 27]. A big difference with image steganography is the way a video file is compressed. For example sections of the frame were a lot of motion is happening will be compressed differently [10]. Because of this there are algorithms used to find the best places in a frame to embed the information [2].

## 3 PROPOSED ALGORITHM

The algorithm embeds a watermark in the *I frames* of a video. These are frames in a video that do not reference any other frames in the video to create the image [4].

Within the I frames the algorithm makes use of *4x4 luminance blocks* to embed bits. Within the luminance blocks are the pixel values which determine the brightness of a pixel. These values are in the Y-channel of the image when converting the image to a YUV color encoding [19].

The bits of the message are embedded in the 4x4 luminance blocks in the frequency domain. To transform a block to the frequency domain a *Discrete Cosine Transformation* (DCT) is done on the block. This transforms the block containing pixel values to a block containing the amplitude of waves which added together describe the pixel values again [19].

These are all aspects of the *H.264 codec* which is a video compression standard used for (among others) mp4 files. The codec describes how video files are compressed, written to storage (coding) and how they should be showed when viewed (decoding) [1, 19].

This algorithm can embed watermarks of arbitrary size and datatype, as long as the watermark is converted to a binary encoding. In this research the watermarks that were embedded are 80x80 pixel black and white images.

The distributed watermark algorithm first splits the watermark into multiple parts (section 3.5). Then select an appropriate 4x4 luminance block to embed a part of the message in (section 3.1, 3.2).

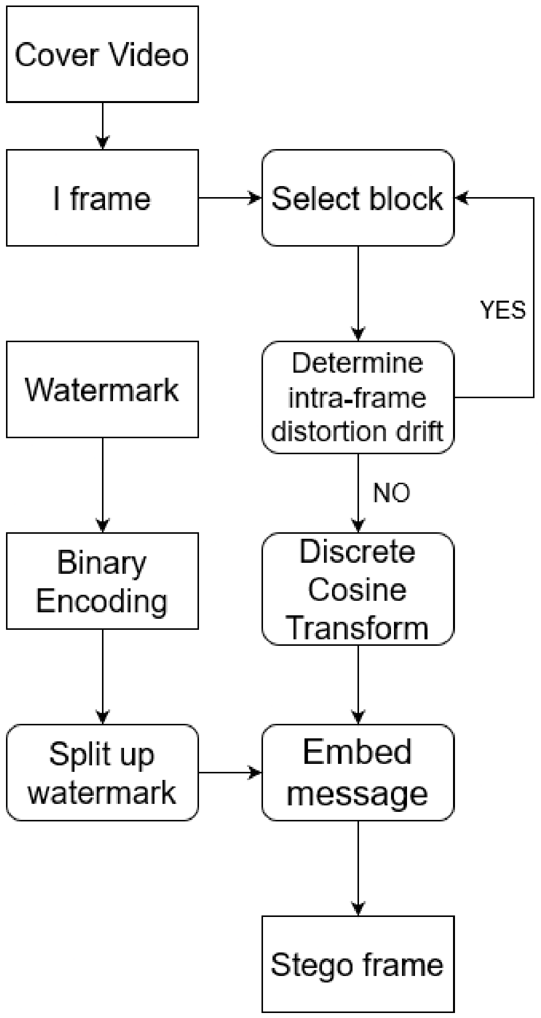


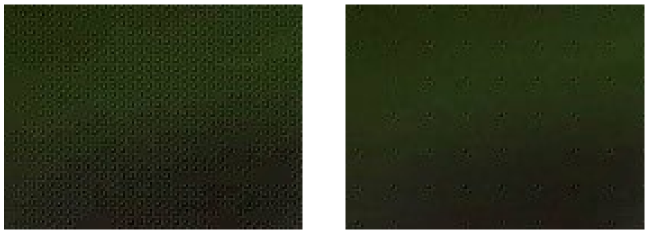
Fig. 1. Flowchart describing the proposed algorithm

And after that transforms that block to the frequency domain and embeds data in it (section 3.3, 3.4). The non-distributing algorithm follows the same steps except it does not split up the watermark. In Fig. 1 the proposed algorithm is visualized using a flowchart.

### 3.1 Block selection

During implementation it was found that if the blocks that were used to embed the message in where chosen naively, say from left to right, top to bottom, the distortion caused by the embedding would be quite easily noticeable with the human eye. Spreading the blocks out would reduce the visibility of the distortions in the stego image. The difference is shown in Fig. 2. Where Fig. 2a was created using the naive way of block selection and figure 2b was created by spreading the blocks out.

Ideally the algorithm would use the entire size of the image so it can spread the blocks used for embedding out as much as possible. However, in order for the extracting algorithm to know which blocks



(a) An image with the blocks used for embedding concentrated

(b) An image with the blocks used for embedding spread out

Fig. 2. An illustration of the visual difference between concentrating block used for embedding versus spreading them out

the message is stored in the locations of these blocks have to be predictable.

To get as much distance between the used blocks as possible while also keeping the locations of the used blocks predictable for the extracting party an algorithm was designed and implemented that calculates the block with furthest distance to the previously used blocks. Doing this over the entire image can become very computationally intensive. In order to keep this technique viable the image is divided into regions of 32 by 32 blocks (128 by 128 pixels). When selecting a block to embed a part of the message in, the algorithm will take the current region and calculate what block in that region is the furthest from all the other blocks that were already used in that region. The pseudo-code for this algorithm can been seen in Algorithm 1. If there are multiple blocks that have a furthest distance from all other used blocks a pseudo-random number will be generated to pick from among them. This is to avoid having a lot of used blocks next to each other on the borders of the regions.

## 3.2 Intra-frame distortion drift prevention

In the H.264 codec compression is done using intra-frame prediction (or spatial prediction). This is a type of compression that works by looking at the value of neighbouring pixels and using those to extrapolate the values of the pixels in the block. For the 4x4 luminance blocks used in this algorithm There are nine different prediction modes as seen in Fig. 3 [19].

For example prediction mode 0 will make use of the pixels above the block to extrapolate the pixel values in the block. If prediction mode 0 applies to a block all pixels in all columns in a block will have the same value as the pixel above that column. It works the same for prediction mode 1 but now instead of columns it uses rows. If prediction mode 1 applies all pixels in all rows in the block will have the same pixel value as the pixel to the left of the row. Intra-frame prediction uses neighbouring pixels to extrapolate or "predict" the values of the pixels within the block.

16x16 luminance blocks can also be predicted using intra-frame prediction with the methods seen in Fig. 4 [19]. Here the same concept applies but with bigger blocks and less prediction modes.

These intra-frame prediction modes may have the unintended consequence that changing a pixel in one part of the frame might

**Algorithm 1** Block selection algorithm within 32 by 32 block region

```
$current\_blocks \leftarrow input$
$rsize \leftarrow 32$
$region \leftarrow rsize$ by $rsize$ 2d array with every entry as 0
$tentative \leftarrow empty\ list$
for $block\ in\ current\_blocks$ do
    $region[block] \leftarrow 1$
end for
while $not\ all\ elements\ in\ region = 1$ do
    for $current\_block\ in\ current\_blocks$ do
        for $adjecent\_block\ to\ current\_block$ do
            if $region[adjecent\_block] = 0$ then
                $tentative \leftarrow adjecent\_block$
                $region[adjecent\_block] \leftarrow 1$
            end if
        end for
    end for
    $current\_blocks \leftarrow tentative$
    $tentative \leftarrow empty\ list$
end while
return $current\_blocks$
```

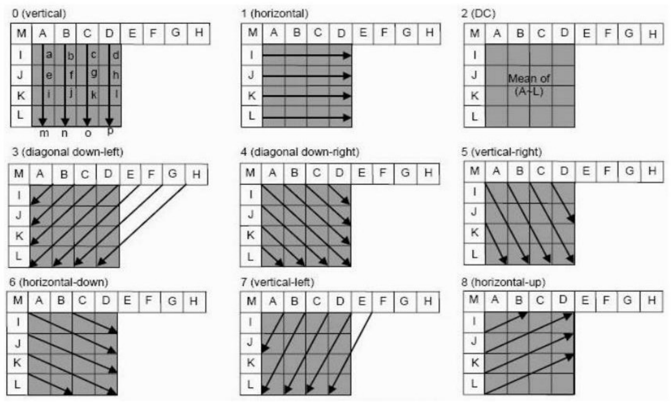


Fig. 3. Prediction modes for 4x4 luminance blocks

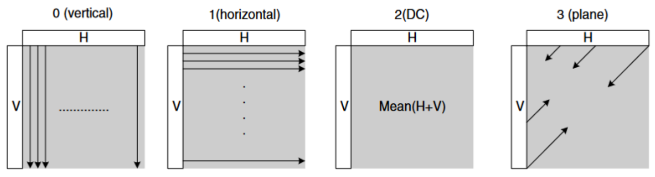


Fig. 4. Prediction modes for 16x16 luminance blocks

create a distortion that, through these prediction blocks, travels through the image. This is called intra-frame distortion drift. In order to mitigate this the algorithm only selects blocks for embedding that are not used by other blocks for intra-frame prediction [14].

To achieve this, whenever a potential 4x4 luminance block is evaluated as a candidate for embedding, it, the three blocks below it and the block to it's right are evaluated to see which possible prediction modes were used for these blocks. This is done per block by looking at the neighbouring pixels and creating a prediction block for each prediction mode. If the prediction block is the same as the actual block the algorithm assumes that prediction mode was used. Note that sometimes multiple prediction modes could have created the actual block in that case the algorithm will take all possible prediction modes into account.

In order for a block to not cause intra-frame distortion drift and thus be eligible for data embedding it needs to meet three conditions:

Condition 1: The block to the evaluated block's right does not use prediction mode 1, 2, 4, 5, 6 or 8 for 4x4 luminance block intra-frame prediction and not mode 1, 2 or 3 for 16x16 luminance block intra-frame prediction.

Condition 2: The block to the evaluated block's under-left (the block one down and one to the left) does not use prediction mode 3 or 7 for 4x4 luminance block intra-frame prediction and not mode 1 for 16x16 luminance block intra-frame prediction

and the block under the evaluated block does not use prediction mode 0, 2, 3, 4, 5, 6 or 7 for 4x4 luminance block intra-frame prediction and not mode 0, 2 or 3 for 16x16 luminance block intra-frame prediction.

Condition 3: The block to the evaluated block's under-right (the block one down and one the right) does not use prediction mode 4, 5 or 6 for 4x4 luminance block intra-frame prediction.

If these conditions are met then there are no blocks that rely on the evaluated block for intra-frame prediction and it is safe to use that block without causing intra-frame distortion drift.

## 3.3 DCT transform

The data will be embedded in 4x4 luminance blocks in the frequency domain. To achieve this the luminance block are transformed using DCT (*discrete cosine transform*). DCT is a compression technique used in the H.264 codec that transforms a matrix representing pixel values spatially to a matrix representing the pixel values as a sum of waves. This compression technique is lossy so the pixel values will not be the exact same as before the transformation [16].

The transformation can be described as follows:

$$\tilde{Y} = C_f Y C_f^T \otimes E \tag{1}$$

Where $W = C_f Y C_f^T$ is the transformation part and $\tilde{Y} = W \otimes E$ is the quantization part.
In the transformation part Y is a 4x4 matrix containing the pixel values of the 4x4 luminance block and $C_f$ is a matrix:

$$C_f = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{pmatrix} \tag{2}$$

And $C_f^T$ is it's transposed.

In the quantization step the values of the resulting matrix $W$ are multiplied element wise with the matrix $E$. The matrix $E$ can be calculated as:

$$E = \frac{MF}{2^{15+floor(QP/6)}} \tag{3}$$

Where QP is the quantization parameter. This parameter can be between 0 and 100 and describes the coarseness of the compression. A high value for QP will result in more data loss but higher data compression[21].

MF is a 4x4 matrix with values depended on QP [20]. For values of QP higher or equal to 5 these values stay the same. For the purposes of this paper QP will be chosen as 28 as this a common QP-value for low quality compression. In the case of $QP \geq 5$ the matrix MF is:

$$MF = \begin{pmatrix} 7282 & 4559 & 7282 & 4559 \\ 4559 & 2893 & 4559 & 2893 \\ 7282 & 4559 & 7282 & 4559 \\ 4559 & 2893 & 4559 & 2893 \end{pmatrix} \tag{4}$$

The final transformed matrix will be:

$$\tilde{Y} = round(W \otimes E) \tag{5}$$

During the embedding phase the matrix containing the watermark data, $\Delta$, will be added to the transformed matrix $\tilde{Y}$ resulting in matrix $\tilde{Y}'$:

$$\tilde{Y}' = \tilde{Y} + \Delta \tag{6}$$

In order to transform matrix $\tilde{Y}'$ back to a matrix containing the pixel values spatially, matrix $Y'$, the inverse transform is used:

$$Y' = C_i^T (\tilde{Y}' \otimes E') C_i \tag{7}$$

Where $C_i$ is the inverse of matrix $C_f$ and $E'$ is:

$$E' = V \cdot 2^{floor(QP/6)} \tag{8}$$

Where $V$ is a matrix which values are depended on QP. The matrix $V$ in case of $QP \geq 5$ is as follows [20]:

$$V = \begin{pmatrix} 18 & 23 & 18 & 23 \\ 23 & 29 & 23 & 29 \\ 18 & 23 & 18 & 23 \\ 23 & 29 & 23 & 29 \end{pmatrix} \tag{9}$$

## 3.4 Data embedding and extraction

In order to embed the data in the DCT blocks the fact that a lot of values in the matrix have become 0 during the transformation is used. A block that had roughly the same Y-pixel values for all sixteen pixels in the block will look like this when transformed into the frequency domain:

$$\tilde{Y} = \begin{pmatrix} n & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{10}$$

The value of an entry being 0 means that these waves are currently not used to modify the pixel values and are thus suitable to embed steganographic information in. However changing some frequency bands will be more obvious then other. The middle frequency bands are the least noticeable [3]. So frequency band 3 and 4 will be used.

Where the entries of frequency band $f$ are all the entries in $\tilde{Y}_{ij}$ where $i + j = f$.

The entries in these frequency bands which are 0 are used to embed the message where each entry will be used to embed one bit. From top to bottom, from left to right, the entry will be changed to a 1 if the currently embedded bit is a 1 and kept a 0 if that bit is a 0. It can occur that there was an entry in one of these frequency bands that was already a 1. In that case the value will be changed to a 2 in order to not accidentally read that bit as part of the message. This process can be summarized in the following formula:

$$\tilde{Y}'_{ij} = \begin{cases} \tilde{Y}_{ij} & \text{if } i + j \notin f \\ 2 & \text{if } i + j \in f \text{ } and \text{ } \tilde{Y}_{ij} = 1 \\ \tilde{Y}_{ij} & \text{if } i + j \in f \text{ } and \text{ } \tilde{Y}_{ij} \neq 0 \text{ } and \text{ } \tilde{Y}_{ij} \neq 1 \\ bit & \text{otherwise} \end{cases} \tag{11}$$

Where $bit$ is the bit that is currently encoded and $f$ is the set of used frequency bands.

To extract data from a DCT block the entries on the chosen frequency bands are looked at from top to bottom, from left to right. If the entry is a 0 or a 1 that bit will be added to the message if not it will be disregarded.

## 3.5 Distribution of the watermark

In order to distribute parts of the watermark over multiple frames the watermark has to be split up. The watermark will be split into 16 parts. The amount of parts the watermark is split up in has two effects:

1. The more parts the watermark is split up in the less robust the watermark becomes as more of the video needs to be preserved in order to be able to retrieve all parts of the watermark.

2. The more parts the watermark is split up in the less bits are embedded per frame and the less distortion will be caused to the frame.

These two effect cause a trade-of and in this implementation the amount of parts has been chosen as 16 although other numbers of parts are also possible.

In every I frame in the video one of these parts of the watermark will be embedded. After all the parts have been embedded the watermark will be embedded again into the next 16 I frames. This means that in every set of 16 consecutive I frames all the data of the watermark will be stored. The first 4 bits in every message embedded into a single frame will serve as a serial number indicating which part of the watermark is embedded into that frame. The watermark will be split up in 16 parts by dividing it in 4x4 pixel blocks. Every pixel in that block will be assigned a number corresponding to the serial number of the frame it will be embedded in. The numbers assigned to the pixels in the 4x4 block are as followed:

| | | | |
|---|---|---|---|
| 0 | 9 | 2 | 11 |
| 4 | 13 | 6 | 15 |
| 8 | 1 | 10 | 3 |
| 12 | 5 | 14 | 7 |

The goal this technique tries to achieve is that when a frame is missing the recognisability of the watermark preserved as much as possible because the pixels missing are not bundled together. This arrangement was also designed to have pixels which will be in frames that are close to each other in the video file far away from each other in the 4x4 block. It was generated by:

$$n = 9i \mod 16 \tag{12}$$

Where n is the position in the flattened table and i is the serial number of the frame.

## 3.6 The proposed scheme

Combining all the methods from the previous sections, the proposed schemes for embedding and extracting will be as follows:

### 3.6.1 *Embedding.*

The algorithm for embedding takes an H.264 encoded video and an 80x80 pixel black and white watermark as input and outputs a stego video:

(1) The watermark will be split up into 16 parts using the method described in section 3.5.
(2) The I frames are extracted from the cover video.
(3) A message is embedded in all the I frames using the following steps:
  (a) The frame is converted to the YUV colorspace.
  (b) A 4 bit serial number is added in front of the message to indicate which part of the watermark is being embedded into the frame.
  (c) A 4x4 luminance block is selected in the frame to embed the data in using the method described in section 3.1.
  (d) The selected block is checked to see whether it will create intra-frame distortion drift as described in section 3.2. If not:
  (e) The block is transformed to the frequency domain using the method described in section 3.3 and then the data is embedded in the block using the method described in section 3.4.
  (f) The block is transformed back to the spatial domain using the method described in section 3.3 and the resulting 4x4 luminance block is put back into the Y colorspace of the image.
  (g) Step 3c to 3f are repeated until the entire message is embedded into the frame.
(4) Step 3 is repeated until all I frames have a message embedded in them.

### 3.6.2 *Extraction.*

The algorithm for extracting will take a stego-video as input and results in a 80x80 pixel black and white watermark as output:

(1) The I frames are extracted from the stego video.
(2) The watermark is extracted using the following steps:
  (a) The frame is converted to the YUV colorspace.
  (b) A 4x4 luminance block is selected in the frame to extract the data in using the method described in section 3.1.
  (c) The selected block is checked to see whether it will create intra-frame distortion drift as described in section 3.2. If not:
  (d) The block is transformed to the frequency domain using the method described in section 3.3 and then the data is

extracted from the block using the method described in section 3.4.

(e) Step 2b to 2d are repeated until the entire message is extracted from the frame. In the case of a 80x80 watermark divided into 16 parts with a 4 bit serial number per frame this is 404 bits.

(3) Steps 1 and 2 are repeated until all 16 parts of the watermark are extracted from the video.
(4) The watermark is reconstructed using the method described in section 3.5.

## 4 IMAGE DISTORTION

During the process of steganographic embedding one of the goals is to minimize the distortion done to the cover object. Changes made to a video frame during the embedding will result in distortion to the image [11]. These distortion can be seen with the human eye when zoomed in on the image. However, under normal circumstances they are not noticeable.

It is still useful to quantify this distortion added to the image by the steganographic embedding to measure the subtlety of the algorithm. There are a couple of metrics for this purpose. The ones used in this research are MSE (*mean square error*), two types of PSNR (*peak-signal-to-noise ratio*) and SSIM (*structural similarity index measure*) [23].

MSE measures the distortion of an image by comparing the pixel values of the two images and measuring the difference. The formula for the MSE value of an image is [23]:

$$MSE = \frac{1}{K} * \sum_{(i,j)} (f_{i,j} - g_{i,j})^2 \tag{13}$$

Where K is the total number of bits in the images and f is the Y-channel of the original image and g is the Y-channel of the distorted image. Both Y-channels are normalized to be in the range [0,1].

PSNR also measure the distortion in an image but relates it to maximum possible strength of the signal. It is measured in dB so it is on a logarithmic scale. There are two ways to measure PSNR over the entire video. Both cases make use of the MSE of the individual distorted frames and calculate an average of the MSE values [13]. The first way will be called PSNR1 and this will be calculated by using the geometric mean:

$$PSNR1 = -\frac{10}{N} * log_{10}(\prod_{k=1}^{N} MSE_k) \tag{14}$$

Where N is the number of frames and $MSE_k$ is the MSE value of the k-th frame.

The second way of calculating the mean PSNR value uses the arithmetic mean. This will be called PSNR2. The formula for PSNR2 is:

$$PSNR2 = -10 * log_{10}(\frac{1}{N} \sum_{k=1}^{N} MSE_k) \tag{15}$$

In some cases PSNR1 and PSNR2 will be very similar however when the variance in the MSE-values of the frames increases the values for the PSNR1 and PSNR2 values will deviate[13].

The last metric to calculate the distortion is SSIM. This metric measures the similarity of two images. The formula for SSIM consist of three parts and is as follows [11]:

$$SSIM(f,g) = l(f,g)c(f,g)s(f,g) \tag{16}$$

The three parts of the of the formula are described as follows:

$$\begin{cases} l(f,g) = \frac{2\mu_f \mu_g + C}{\mu_f^2 + \mu_g^2 + C} \\ c(f,g) = \frac{2\sigma_f \sigma_g + C}{\sigma_f^2 + \sigma_g^2 + C} \\ s(f,g) = \frac{\sigma_{fg} + C}{\sigma_f \sigma_g + C} \end{cases} \tag{17}$$

l(f,g) describes the difference in mean luminance between the two images where $\mu_f$ and $\mu_g$ are the mean values of the pixels in the Y-channel of the images f and g respectively and normalized to be in the range [0,1]. c(f,g) describes the difference in contrast between the images f and g and is measured using the variance of the pixel values. Here $\sigma_g$ and $\sigma_g$ are the variance of the pixel values in the Y-channels of image f and g respectively. s(f,g) compares the structure of the two images by measuring the correlation coefficient between f and g. Here $\sigma_{fg}$ denotes the covariance between the pixel values in the Y-channel of image f and g. C is a very small positive constant used to ensue the denominator is never zero.

These four metrics, MSE, PSNR1, PSNR2 and SSIM, were used to measure the rate of distortion between the original frames and the stego frames. For each video PSNR1 and PSNR2 were calculated as well as the arithmetic mean of the MSE and SSIM over all the frames in each video.

## 5 ROBUSTNESS

Robustness is the property of a stenographic embedding to withstand attacks on the stego object. These attacks are mutation done on the stego object after the embedding. In the case of a video these attacks can entail changes within the frames or changes to the order of the frames. In this research the attacks will be picked as to reflect attacks that can be easily done to videos [17]. The attacks will be:

(1) cropping
(2) frame deletion
(3) brightness change
(4) compression

Resistance to cropping was measured by cropping the frames in the video four different times. Each time a quarter of the frame was cropped from the video. The left, right, bottom and top quarter were deleted.

Resistance to frame deletion was measured by selecting a portion of the I frames in the video and then letting the algorithm purposely ignore those frames when extracting the watermark. The amount of I frames ignored was 20%, 40%, 60% and 80% of the total amount of I frames in the video.

Resistance to brightness change was measured by after reading the image and extracting the Y-channel, multiplying every pixel value in the Y-channel by a certain factor. After that, the algorithm extracted the watermark form the frame. The factors used were 0.8, 0.9, 1.1 and 1.2.

Resistance to compression was measured by compressing the jpg files of the embedded frames using Ceasium Image Compressor [18]. The files were compressed on JPEG quality setting: 70, 80 and 90. After that for each level of compression the watermarks were extracted from the compressed images.

To measure how well the video withstood the attacks the Shannon entropy [6] of the resulting bit stream is calculated. In order to calculate the Shannon entropy the amount of correctly extracted bits was counted and then divided by the total amount of bits. The result is the probability of a given bit being correct:

$$P = \frac{correct\ bits}{total\ bits} \tag{18}$$

Where P is the probability. With this probability the Shannon entropy of the resulting message can be calculated using the following formula:

$$H = -P * log_2(P) - (1 - P) * log_2(1 - P) \tag{19}$$

Where H is the Shannon entropy. Entropy is in a range of [0, 1]. The closer to 0 the measured entropy of the resulting bitstream is after an attack to more robust the algorithm is to said attack.

## 6 EXPERIMENTAL RESULTS

To measure image distortion and robustness five videos were recorded and compressed. These videos will be referred to as shoes, globe, kallax, garden and desk as these are the main objects in the video. The videos are all 240 frames long and have a minimum I frame interval of 4 frames. They have a resolution of 1280 by 720 pixels and a quantization parameter of 23. The videos are all of one continuous shot and thus do not contain any cuts. The original videos are uploaded to Gitlab [25].

Four watermarks were created to embed in the videos. The watermarks will be referred to as utwente, face, woman and feyenoord. The watermarks are shown in figure 5. The watermarks are also uploaded on Gitlab [26]. All of these watermark are 80x80 pixel black and white picture and were converted to a 6400 bit binary encoding before embedding. In this encoding a 0 represents a white pixel and a 1 represents a black pixel. All four watermarks were embedded in all five videos.

The watermarks were embedded in the videos using four different methods:

1. The entire watermark was embedded in each frame resulting in 6400 bits per frame (**non-distributing algorithm**)
2. The watermark was split into 16 parts and a part was embedded in each frame resulting in 404 bits per frame (**distributed watermark algorithm**)
3. The entire watermark was duplicated and embedded in each frame resulting in 12800 bits per frame
4. The watermark was split into 16 parts every part was duplicated and embedded in each frame resulting in 808 bits per frame

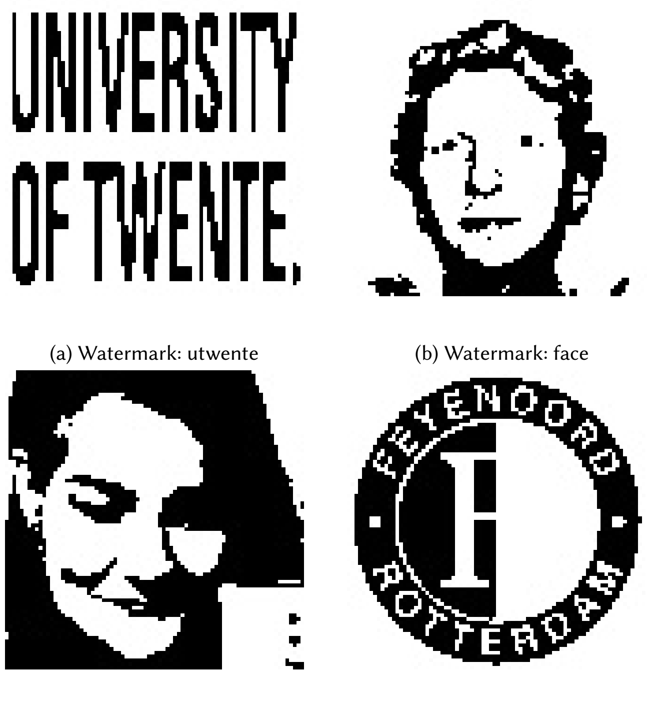
Method 1 is used as a control for method 2. Method 3 and 4 are used to test what happens to image distortion if the algorithms would have made use of error correcting code. In this case the assumption is made that the amount of redundant bits will be the same as the amount of data bits when an error correcting encoding is applied.

This means that in total 80 different combinations of videos, watermarks and embedding methods were used to gather data on the effect of different methods.

The embedded frames were extracted from the videos and saved as jpg-files and tested on as such as the methods used by the researcher to insert the frames back into the video, using the opencv library for python, would cause to much mutation to the embedded frames and the I frame intervals that it was impossible to reliably extract the watermark from the video again. To mitigate this the images were saved as jpg files and then treated as if the frames were extracted from a stego video file. However it is very likely a more sophisticated method might exist to insert the frames back into a video to lessen mutation to the frames.

### 6.1 Extraction

After embedding the watermarks in the frames using the different methods, the watermarks were extracted from the images in the jpg-files again. During this process it was noticed that not every video was as well suited for steganographic embedding and extraction as others. The amount of correct bits extracted per video and a



(a) Watermark: utwente (b) Watermark: face

(c) Watermark: woman (d) Watermark: feyenoord

Fig. 5. The four watermarks used in this research

comment about the legibility of the resulting watermark can be found in the tables below. Table 1 contains the result for method 1. Table 2 contains the result for method 2.

For some videos it was the case that all four watermarks showed the same performance during extraction in that case it is indicated by "All watermarks" in the watermark column.

The embedded watermarks are all 80x80 black and white pixels. This means a watermark is encoded into 6400 bits. So in the column *correct bits* the maximum value is 6400.

| Video | Watermark | correct bits | entropy | legibility |
|---|---|---|---|---|
| Shoes | All watermarks | 6400 | 0.000 | perfect |
| Globe | All watermarks | 6400 | 0.000 | perfect |
| Kallax | All watermarks | 6400 | 0.000 | perfect |
| Garden | utwente | 3933 | 0.962 | Parts shifted |
| | face | 4293 | 0.914 | Parts shifted |
| | woman | 3213 | 1.000 | Parts shifted |
| | feyenoord | 3454 | 0.995 | Parts shifted |
| Desk | utwente | 3881 | 0.967 | illegible |
| | face | 4217 | 0.926 | illegible |
| | woman | 3070 | 0.999 | illegible |
| | feyenoord | 3471 | 0.995 | illegible |

Table 1. Extraction results of non-distributing algorithm

| Video | Watermark | correct bits | entropy | legibility |
|---|---|---|---|---|
| Shoes | utwente | 6399 | 0.002 | near perfect |
| | face | 6400 | 0.000 | perfect |
| | woman | 6341 | 0.076 | little noise |
| | feyenoord | 6318 | 0.099 | little noise |
| Globe | All watermarks | 6400 | 0.000 | perfect |
| Kallax | All watermarks | 6400 | 0.000 | perfect |
| Garden | utwente | 4349 | 0.905 | illegible |
| | face | 4491 | 0.879 | illegible |
| | woman | 3375 | 0.977 | illegible |
| | feyenoord | 3696 | 0.982 | illegible |
| Desk | utwente | 6021 | 0.324 | some noise |
| | face | 6004 | 0.335 | some noise |
| | woman | 5631 | 0.530 | a lot of noise |
| | feyenoord | 5894 | 0.399 | a lot of noise |

Table 2. Extraction results distributed watermark algorithm

The comments on legibility were added because in some cases like the embedding of the face watermark in the garden video using the non-distributing algorithm the entropy suggests that the watermark should be unrecognisable however parts of the watermark are still in tact although shifted, see Fig. 6. So just relying on entropy does not give the full picture.

The tables show that the videos: shoes, globe and kallax are much better suited for extractable embedding than garden and desk. Why this is, is not yet clear but has probably something to do with the way the codec compresses these images when writing them to the disk [19]. Nevertheless, these results do show that under some

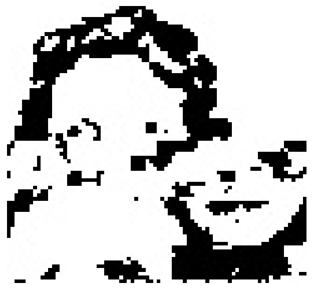


Fig. 6. Results of extraction after embedding the face watermark form the garden video using the non-distributing algorithm

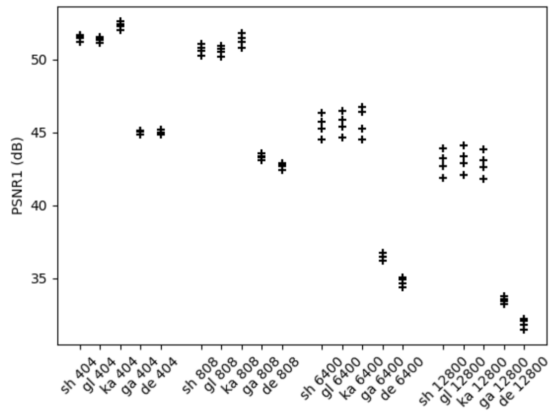


Fig. 7. PSNR1 values per video and embedded bits

circumstances the embedding and extraction methods described in this paper work flawlessly.

## 6.2 Results image distortion

To quantize the amount of image distortion the embedding of a watermark has created in the cover image the metrics described in section 4 are used. This was done for all 80 combinations of video, watermark and method.

The values for PSNR1 and PSNR2 are plotted in Fig. 7 and 8 respectively. Here **sh**, **gl**, **ka**, **bf**, **ga** and **de** refer to the videos shoes, globe, kallax, garden and desk respectively and the number after it refers to the amount of bits embedded per frame. Every + is a different watermark embedded in the video.

It can be seen that the less bits are embedded in the image the higher the PSNR value is. A higher PSNR value means that there is less distortion in the image. So these graphs suggest that the distributed watermark algorithm will cause less distortion in the image, as the average PSNR1 value for a frame with 404 bits embedded in it is 49.06dB while the average PSNR1 value for a frame with 6400 bits embedded in it is 41.61dB which on a logarithmic scale is a considerable difference.

It can also be seen that the videos: garden and desk have a lot lower PSNR1 and PSNR2 value for every amount of embedded bits relative to videos with the same amount of bits per frame. These

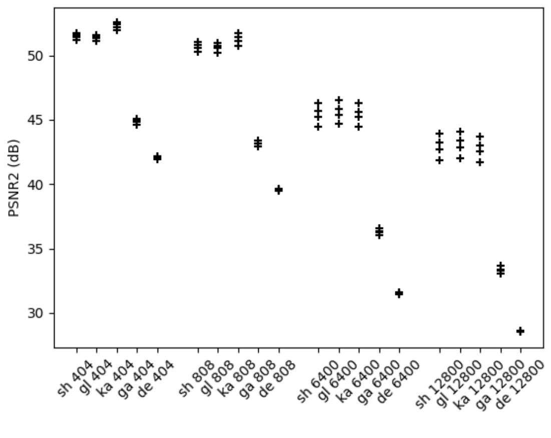


Fig. 8. PSNR2 values per video and embedded bits

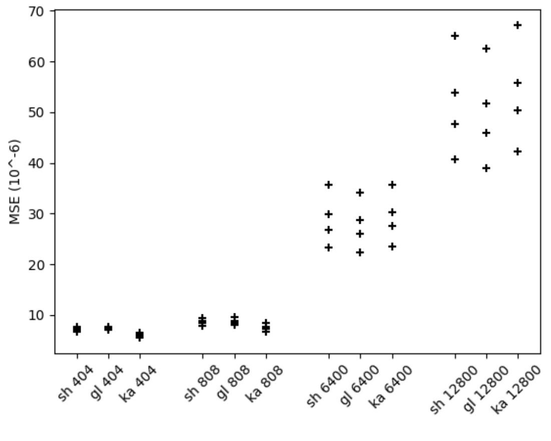


Fig. 9. MSE values per video and embedded bits

are also the videos that had a poor performance on extracting the watermark from the video. The fact that the PSNR values for these two videos is substantially lower than for the other three suggests that between writing and reading the stego images additional mutations have been done to the image, probably by the codec [19]. This would both explain why these videos had a poor extraction performance compared to the other three and why the PSNR values are lower. Why this only happened to these two videos and not the other three is yet unclear.

Because PSNR is on a logarithmic scale, the scale of the graphs are not stretched out too much due to these outliers. However, MSE and SSIM are not logarithmic and plotting the results with the results of garden and desk included will distort the scales of the graphs. Moreover, it could be argued that the result of garden and desk are not as relevant because their extraction performance is quite poor. For these reasons the results of garden and desk will be omitted from the graphs showing the results of MSE and SSIM.

Fig. 9 shows the average MSE values for the frames per video per embedding amount per watermark. The lower the value for MSE is the less distortion is present in the stego image. This graph shows again that having a lower amount of embedded bits will result in less distortion of the image.

This graph also shows something about the effect the watermark has on the distortion of the image. In the graph, especially for the videos with a higher amount of embedded bits, it can be seen that the MSE values for the different watermarks are separated by a noticeable amount. When looking at the data it can be seen that for every video and every amount of embedded pixels the order of watermarks that generate the most distortion is the same. Namely, from high to low: woman, feyenoord, utwente, face. When looked at the watermarks with the most amount of black pixels, which are translated to 1's in the binary encoding of the watermark, the same order arises: woman (3568), feyenoord (2672), utwente (2171), face (1603). Where the number behind the name of the watermark is the amount of black pixels in that watermark out of the total of 6400 pixels in each watermark.

When the woman watermark was embedded entirely in the video shoes: the amount of black pixels embedded in each frame was 3568. This resulted in an average MSE per frame of $35.62 * 10^{-6}$. When looked at the face watermark that was duplicated and then embedded entirely in the video shoes: the amount of black pixels in each frame was 3206. This resulted in an average MSE per frame of $40.75 * 10^{-6}$. While the amount of 1's embedded in these two examples are comparable the amount of 0's embedded in the first example is only 2832 while for the second example this number is 9594 yet the average MSE are fairly close together. This example suggest that the amount of 1's embedded in an image might be correlated stronger to the rate of distortion than the total number of bits is correlated to the rate of distortion.

| Watermark | total bits embedded | 1's embedded | MSE |
|---|---|---|---|
| face | 6400 | 1603 | $23.30 \cdot 10^{-6}$ |
| woman | 6400 | 3568 | $35.62 \cdot 10^{-6}$ |
| face | 12800 | 3206 | $40.75 \cdot 10^{-6}$ |
| woman | 12800 | 7136 | $65.13 \cdot 10^{-6}$ |

Table 3. total bits embedded versus 1's embedded in the video shoes and it's effect on the MSE

In Table 3 the effect of the number 1's embedded versus the total number of embedded bits on the average MSE in the video shoes is highlighted. Here the previously mentioned correlation is visible again.

This could be explained by the fact that when embedding bits in the DCT matrix of a 4x4 luminance block, the algorithm will when having to embed a 1 change a 0 in the matrix to a 1. However, when the algorithm has to embed a 0 in the matrix it will leave the 0 that was already there unchanged. This means that only when embedding 1's the algorithm mutates part of the DCT matrix and thus the image.

In Fig. 10 the SSIM values for the different videos, amount of embedded bits and watermarks can be found. For SSIM applies that the higher the SSIM value is the less distortion is present in the image. For this metric the results again show that a lower amount of

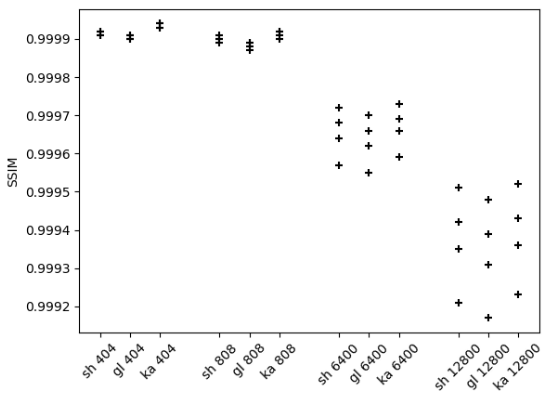


Fig. 10. SSIM values per video and embedded bits

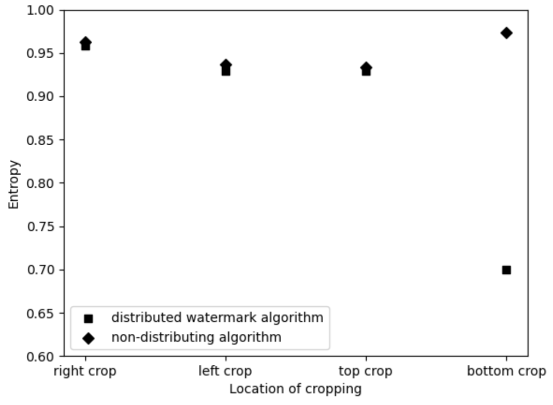


Fig. 11. The location of cropping versus the resulting entropy

bits embedded per frame results in a lower rate of distortion. When measuring distortion using the SSIM metric the same patterns arise, where less embedded bits in the frame cause less distortion and watermarks that contain more black pixels will create more distortion in the stego image.

In every metric used to analyse the rate of distortion caused by embedding the watermark into the frames of a video the same two patterns arise:

(1) The more bits embedded into the cover image the more distortion is measured on the stego image everything else being the same.
(2) The more 1's are present in the binary message embedded in the cover image to more distortion is measured in the stego image everything else being the same.

## 6.3 Results robustness

To measure the robustness of the algorithm, attacks have been done on the jpg files containing the video frames with the watermark embeddings. The attacks were: cropping, frame deletion, brightness change and compression. How these attacks were executed can be read in section 5.

To measure robustness only the videos: shoes, globe and kallax have been tested since garden and desk already had flawed watermarks extracted from them without any attacks done to the stego object. It will not yield any relevant data to measure how well the algorithm extracts the watermark after an attack.

In the three above mentioned videos the four watermarks were embedded using method 1 and 2 from section 6. After the frames have had a (piece of the) watermark embedded in it, they were saved as jpg files. After that these files were read and attacked and then the watermark was extracted using the corresponding algorithm.

The performance in terms of robustness is expressed in the Shannon entropy of the resulting bitstream. In this metric 0 means the watermark was extracted flawlessly and 1 means the extracted bitstream has no correlation to the original watermark.

In Fig. 11 The amount of Shannon entropy in the extracted bitstream is plotted against the location of cropping, where the squares are the results for the distributed watermark algorithm and the diamonds for the non-distributing algorithm. In every attack 25% of the pixels in the frame were cropped from the mentioned side of the frame.

The figure shows that neither of the algorithms are robust against cropping attacks with all the data points lying well above 0.9 entropy, except for cropping the bottom quarter using the distributed watermark algorithm.

If the dimensions of the image are changed in between embedding and extracting, the blocks the extracting algorithm chooses will not corresponds to the blocks the embedding algorithm has chosen. This will mean the algorithm will not extract data from the appropriate blocks. This is what happens after cropping.

The effect of this is slightly smaller for the distributed watermark algorithm with a bottom crop-attack as the algorithm will only start to erroneously select block as soon as it reaches the bottom 25% of image in this case. That means that assuming 8 bits were embedded in every DCT block the first 320 extracted bits were correct which is 79% of the total bits in the watermark and this corresponds with the results of the test where an average of 80% of bits were correctly extracted.

The reason the bottom crop-attack still yield a high entropy for the non-distributing algorithm is because this algorithm embedded 6400 bits in the image but still only the first 320 extracted bits are unaffected after this attack. This means a much smaller percentage (only 5%) of the extracted bits are unaffected.

Something that is curious about the top crop-attack is that even though the entropy of 0.929 suggest that the watermark should be eligible, the watermarks produced by the distributed watermark algorithm are actually quite recognisable. See Fig. 12.

In Fig. 13 the effect of brightness change on the entropy of the extracted bitstream is measured, where the squares are the results for the distributed watermark algorithm and the diamonds for the non-distributing algorithm.

A clear pattern emerges where the closer the factor of brightness change is to 1 (no brightness change) the smaller the entropy is. This is obvious because the algorithm makes use of the Y-channel to embed and extract bits and changing the brightness of an image

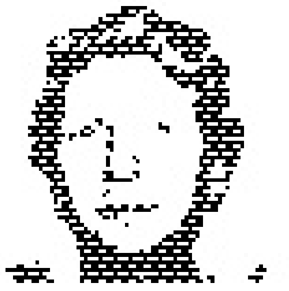


Fig. 12. result of top crop attack on shoes video with face watermark embedded with the distributed watermark algorithm

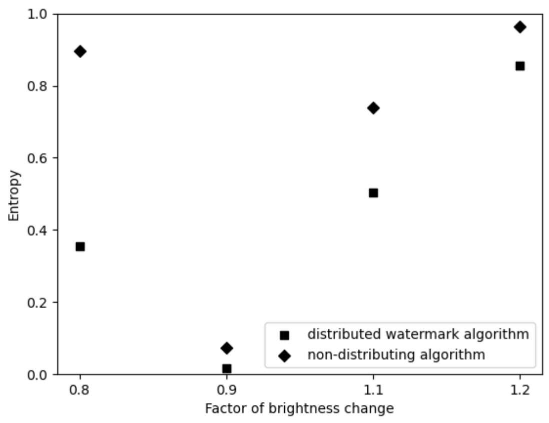


Fig. 13. The factor of brightness change versus the resulting entropy

changes the values of the Y-channel. So naturally more errors will occur the more these values are changed.

Two other patterns emerge from this graph: 1. The distributed watermark algorithm seems less affected by brightness change than the non-distributing algorithm. 2. Lowering the brightness has less effect on the entropy than increasing the brightness.

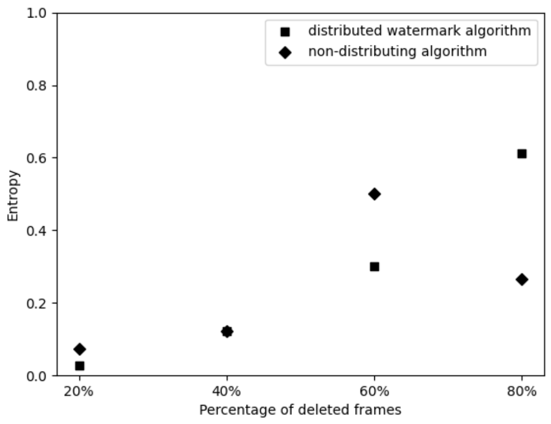


Fig. 14. The percentage of frames deleted versus the resulting entropy

In Fig. 14 the effect of deleting a certain amount of frames from the pool of images the algorithm can use to extract the watermark, on the entropy of the extracted bitstream is measured, where the squares are the results for the distributed watermark algorithm and the diamonds for the non-distributing algorithm.

In Fig. 14 it can be seen that the distributed watermark algorithm is vulnerable to frame deletion and the more frames deleted the higher the measured entropy becomes. However, it must be said that the algorithm is far more resistant to this attacks than to the other attack done in this research, which is to be expected because every deletion of a single frame can either corrupt 1/16 of the watermark OR another frame containing that piece of the watermark is still in the video and no part of the watermark is lost.

Something that is unexpected is that deleting frames also had a negative effect on the non-distributing algorithm. This could be because not every frame will have a perfect representation of the watermark in it after it has been saved to the disk as a jpg file and some compression has been done to it by the codec. Deleting frames might make it so that the video will only have frames left containing distorted embeddings of the watermark.

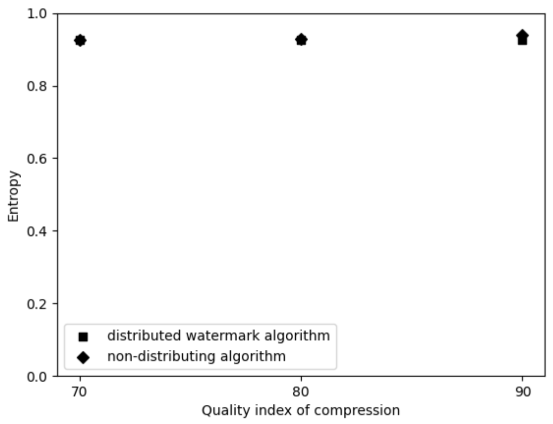


Fig. 15. The jpeg quality of compression versus the resulting entropy

In Fig. 15 the effect of different quality levels of compression on the entropy of the extracted bitstream is measured, where the squares are the results for the distributed watermark algorithm and the diamonds for the non-distributing algorithm.

It can be seen that for both algorithms the watermark is entirely destroyed when the images are compressed on any setting. The reason the entropy is not 1.000 is most likely because if the algorithm cannot extract enough bits from the frame it will supplement the bitstream with 0's. Because three of the four watermarks used in this research contain more white pixels than black pixels, extracting a fully white watermark will have a correct fraction of bits slightly above 50% which will yield a slightly lower entropy than 1.000.

## 7 DISCUSSION

The experiments conducted showed the distributed watermark algorithm had some advantages over the non-distributing algorithm.

Especially in terms of image distortion a clear trend could be seen that a lower amount embedded bits per frame caused less distortion. This is what was expected and one of the core motivation to conduct this research on watermark distribution in video frames.

The experiments showed some difference in robustness between the distributed watermark algorithm and the non-distributing algorithm. In some attacks like the bottom crop-attack and the brightness change attack the results even suggested that embedding less bits in a image increased it's robustness. The distributed watermark algorithm did show to be susceptible to frame deletion, as expected. However, what was unexpected is that the algorithm that embedded the watermark in every frame in it's entirety also experienced increased entropy when frames were deleted which is an unexpected result.

Two shortfalls of this research and algorithm design were:

1. The algorithm did not work for every video. It could happen that either some of or all of the watermark would be destroyed between embedding and extracting. This means that the algorithm is not yet universal in which videos are suitable.

2. The algorithm was not tested by inserting and extracting the stego frames in the video but rather extracting the cover frames from a video and then saving the stego frames as jpg files. This is because inserting and extracting the frames in the video using opencv would mutate the stego frames and videos so much that in no cases the watermark was retrievable afterwards. This is most likely because opencv is a tool for computer vision and not for video editing and thus will compress and alter the video when saved more carelessly.

These two shortfalls did reduce the external validity of the research. The results and conclusions drawn from the experiments should not differ to much from what it should be if these problems were not present. Since both issues probably arise from compression being done on the files after writing the result to the disk either as a jpg file or an mp4 file, a more sophisticated image processing library might mitigate these problems and make the algorithm feasible to actually export a watermarked video instead of a set of jpg files and to be consistent in it's ability to watermark all mp4 videos.

## 8 CONCLUSION

In this paper a steganographic video watermarking algorithm that distributed a 6400 bit watermark over 16 frames was designed and implemented. It's potential advantages in terms of image distortion and robustness were studied by comparing it against a similar algorithm that embedded the entire watermark in each frame.

In terms of image distortion it can be concluded that in the experiments done in this research the distributed watermark algorithm performed better in every metric. This was a consequence of having to embed less bits per frame therefore changing less of the image. Another observation made during these test is that the watermark itself can have a significant effect on the amount of distortion in the stego image. The amount of 1's embedded in the image may have a stronger correlation to the rate of distortion than the total amount of bits.

In terms of robustness multiple attacks have been tested and the algorithms showed some differences from each other.

In theory both algorithms should be quite resistant to frame deletion especially the non-distributing algorithm as having a watermark embedded in every frame should make it immune to this attack. However, this is not what was seen in the results of these test. The non-distributing algorithm did show some vulnerability to frame deletion which is a very unexpected result. Why this happened is yet unclear. The distributed watermark algorithm did show an increase in entropy with an increase in frames deleted which is an expected result.

When measuring the resistance to cropping it was found that when cropping parts of the left or right side of the video this would have a big effect on the entropy of the resulting bitstream for both algorithms. The distributed watermark algorithm was a lot more resilient to the bottom crop-attack because of it's lower amount of embedded bits. Also it showed to be more resilient against top crop-attacks. Even though the entropy of the bitstream was very high, the extracted watermarks after this attack are still recognisable to humans. This could not be said about the non-distributing algorithm.

Brightness change did not effect the entropy significantly for low factors of brightness change however the distributed watermark algorithm did perform better than the non-distributing algorithm.

Neither of the algorithms were resistant to compression attacks. For both algorithms the watermark was entirely destroyed after any amount of compression.

From a standpoint of image distortion as well as robustness the distributed watermark algorithm shows advantages over the non-distributing algorithm proving the merit of this concept.

## 9 FUTURE WORK

In this study the stego frames were saved as jpg files and never actually inserted back into the video. This is because when this was attempted using the opencv library for python the watermark would be destroyed due to compression. So a big improvement on this research would be to insert the stego frames back into the original video without having it compressed so much the watermark is destroyed.

Both algorithms could benefit from more robustness as currently they have a very limited resilience against most attacks. A way this might be improved is to encode the message in an error correcting encoding like hamming code or BCH encoding. The watermarking would also greatly benefit in terms of robustness if a block selection method would be designed that is resilient against cropping.

Since it was found that the amount of black pixels in the watermark, which translated to 1's in the binary encoding, had a strong effect on the distortion in the image, the algorithm could benefit from an encoding scheme that minimizes the amount of 1's in the watermark or at least neutralizes this effect.

The metric used to quantize robustness was the Shannon entropy which was measured by counting the ratio of correct bits. However, this might not be a fair way to asses how much of the watermark was successfully extracted. Let's say the algorithm erroneously adds a couple of bits to the start of the watermark and then flawlessly extracts the watermark. In that case the watermark will be shifted and a very high entropy value will be measured. However, most of the bits were extracted correctly. So robustness would be able to be measured better if there was another way to measure the correctness of the extracted watermark.

## REFERENCES

[1] 2021. H.264 (V14) (08/2021). https://doi.org/11.1002/1000/13903

[2] Bandyapadhay S.K. Biswas, R. 2020. Random selection based GA optimization in 2D-DCT domain color image steganography. *Multimed Tools Appl* 79 (March 2020), 7101–7120. https://doi.org/10.1007/s11042-019-08497-x

[3] Jasdeep Singh Blossom Kaur, Amandeep Kaur. 2011. STEGANOGRAPHIC APPROACH FOR HIDING IMAGE IN DCT DOMAIN. *International Journal of Advances in Engineering Technology* (July 2011). https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=70fbf3b7946777a458b730a6b8791f74040c983b

[4] David R. Bull. 2014. *Chapter 4 - Digital Picture Formats and Representations*. Academic Press, Oxford. 99–132 pages. https://doi.org/10.1016/B978-0-12-405906-1.00004-0

[5] Lloyd A Davidson. 2005. The end of print: digitization and its consequence–revolutionary changes in scholarly and social communication and in scientific research. *International journal of toxicology* 24, 1 (Jan. 2005), 25–34. https://doi.org/10.1080/10915810590921351

[6] Ivan Djordjevic. 2012. *Chapter 3 - Quantum Circuits and Quantum Information Processing Fundamentals*. Academic Press, Oxford. 91–117 pages. https://doi.org/10.1016/B978-0-12-385491-9.00003-4

[7] Gwenaël Doërr and Jean-Luc Dugelay. 2003. A guide tour of video watermarking. *Signal Processing: Image Communication* 18, 4 (2003), 263–282. https://doi.org/10.1016/S0923-5965(02)00144-3 Special Issue on Technologies for Image Security.

[8] Mehdi Fallahpour and David Megias. 2008. Reversible Data Hiding Based On H.264/AVC Intra Prediction, Vol. 5450. 52–60. https://doi.org/10.1007/978-3-642-04438-0_5

[9] Eric Fleischmann. 1987. The Impact of Digital Technology on Copyright Law. *UIC John Marshall Journal of Information Technology PrivacyUIC John Marshall Journal of Information Technology Privacy Law* 8, 1 (Winter 1987). https://repository.law.uic.edu/cgi/viewcontent.cgi?article=1452&context=jitpl

[10] David Griberman and Pavel Rusakov. 2016. Comparison of Video Steganography Methods for Watermark Embedding. *Applied Computer Systems* 19 (05 2016). https://doi.org/10.1515/acss-2016-0007

[11] Alain Horé and Djemel Ziou. 2010. Image Quality Metrics: PSNR vs. SSIM. In *2010 20th International Conference on Pattern Recognition*. 2366–2369. https://doi.org/10.1109/ICPR.2010.579

[12] Gaganjot Kaur Jasvir Singh. 2015. Review of Spatial and Frequency Domain Steganographic Approaches. *International Journal of Engineering Research Technology* 4, 6 (June 2015). https://www.ijert.org/research/review-of-spatial-and-frequency-domain-steganographic-approaches-IJERTV4IS061082.pdf

[13] Onur Keleş, M. Akn Ylmaz, A. Murat Tekalp, Cansu Korkmaz, and Zafer Doğan. 2021. On the Computation of PSNR for a Set of Images or Video. In *2021 Picture Coding Symposium (PCS)*. 1–5. https://doi.org/10.1109/PCS50896.2021.9477470

[14] Yunxia Liu, Leiming Ju, Mingsheng Hu, Xiaojing Ma, and Hongguo Zhao. 2015. A robust reversible data hiding scheme for H.264 without distortion drift. *Neurocomputing* 151 (2015), 1053–1062. https://doi.org/10.1016/j.neucom.2014.03.088

[15] Yunxia Liu, Shuyang Liu, Yonghao Wang, Hongguo Zhao, and Si Liu. 2019. Video steganography: A review. *Neurocomputing* 335 (2019), 238–250. https://doi.org/10.1016/j.neucom.2018.09.091

[16] H.S. Malvar, A. Hallapuro, M. Karczewicz, and L. Kerofsky. 2003. Low-complexity transform and quantization in H.264/AVC. *IEEE Transactions on Circuits and Systems for Video Technology* 13, 7 (2003), 598–603. https://doi.org/10.1109/TCSVT.2003.814964

[17] A. Melman O. Evsutin and R. Meshcheryakov. 2020. Digital Steganography and Watermarking for Digital Images: A Review of Current Research Directions. *IEEE Access* 8 (2020), 166589–166611. https://doi.org/10.1109/ACCESS.2020.3022779

[18] SaeraSoft Matteo Paonessa. 2024. *Caesium Image Compressor*. https://saerasoft.com/caesium

[19] Iain E. G. Richardson. 2003. *H.264 and MPEG-4 Video Compression: Video Coding for Next-Generation Multimedia*. John Wiley Sons, Ltd. https://doi.org/10.1002/0470869615

[20] Iain E. G. Richardson. 2009, 2010. 4x4 Transform and Quantization in H.264/AVC.

[21] Iain E. G. Richardson. 2011. *The H. 264 advanced video compression standard*. John Wiley & Sons.

[22] M.S. Olivier T. Morkel, J.H.P. Eloff. 2005. AN OVERVIEW OF IMAGE STEGANOGRAPHY. (June 2005). https://www.researchgate.net/publication/220803240_An_overview_of_image_steganography

[23] Mohammed Shorif Uddin Umme Sara, Morium Akter. 2019. Image Quality Assessment through FSIM, SSIM, MSE and PSNR—A Comparative Study. *Journal of Computer and Communications* (2019). https://doi.org/10.4236/jcc.2019.73002

[24] Ministerie van Economische Zaken en Klimaat. 2011. *Auteursrecht en naburige rechten*. Retrieved November 22, 2023 from https://www.rvo.nl/onderwerpen/octrooien-ofwel-patenten/vormen-bescherming/auteursrecht#auteursrecht

[25] Jan van Zwol. 2024. *Videos used for testing*. https://gitlab.utwente.nl/s2159732/watermarking-project/-/tree/main/videos?ref_type=heads

[26] Jan van Zwol. 2024. *Watermarks used for testing*. https://gitlab.utwente.nl/s2159732/watermarking-project/-/tree/main/watermarks?ref_type=heads

[27] Jing Zhang, Anthony Ho, Gang Qiu, and Pina Marziliano. 2007. Robust video watermarking of H.264/AVC. *Circuits and Systems II: Express Briefs, IEEE Transactions on* 54 (03 2007), 205 – 209. https://doi.org/10.1109/TCSII.2006.886247