Signal Processing: Image Communication 48 (2016) 1–11

Contents lists available at ScienceDirect

Signal Processing: *Image Communication*

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

Hiding color watermarks in halftone images using maximum-similarity binary patterns

Pedro Garcia Freitas [a,*], Mylène C.Q. Farias [b], Aletéia P.F. Araújo [a]

[a] Department of Computer Science, University of Brasília (UnB), Brasília, Brazil
[b] Department of Electrical Engineering, University of Brasília (UnB), Brasília, Brazil

ARTICLE INFO

Article history:
Received 3 June 2016
Received in revised form
25 August 2016
Accepted 25 August 2016
Available online 26 August 2016

Keywords:
Color embedding
Halftone
Color restoration
Watermarking
Enhancement

ABSTRACT

This paper presents a halftoning-based watermarking method that enables the embedding of a color image into binary black-and-white images. To maintain the quality of halftone images, the method maps watermarks to halftone channels using homogeneous dot patterns. These patterns use a different binary texture arrangement to embed the watermark. To prevent a degradation of the host image, a maximization problem is solved to reduce the associated noise. The objective function of this maximization problem is the binary similarity measure between the original binary halftone and a set of randomly generated patterns. This optimization problem needs to be solved for each dot pattern, resulting in processing overhead and a long running time. To overcome this restriction, parallel computing techniques are used to decrease the processing time. More specifically, the method is tested using a CUDA-based parallel implementation, running on GPUs. The proposed technique produces results with high visual quality and acceptable processing time.

1. Introduction

Printing an image consists of performing a conversion from digital to analog, while scanning an image involves a conversion from analog to digital. These two processes may add several types of distortions to the original content, which include geometric distortions (rotation, scaling, cropping, etc.), color distortions, and noise. These distortions are a consequence of several factors, like, for example, the process of conversion from digital images to halftone representations performed just before printing [1].

The halftone representation is generated using a mathematical model that produces the illusion of colors by using a combination of colored dot patterns [2]. Due to the low-pass property of the Human Visual System (HVS), halftone images are perceived as continuous tone images when viewed from a distance. Many different halftoning methods have been developed over the years, like for example Direct Binary Search (DBS) [3,4], Ordered Dithering (OD) [5,6], Error Diffusion (ED) [7–9], and Dot Diffusion (DD) [10–12]. Although there is a great diversity of image halftoning methods, these methods insert distortions during the quantization process that converts multi-level images (color or grayscale) into binary (halftone) images.

Scanner devices read the printed halftone and restore a multi-level image via an inverse halftoning algorithm [13,14]. Therefore, the scanning process corresponds to the inverse of the printing process. Although the inverse halftoning algorithm is able to recover an approximation of the intensity levels of the original image, the reconstructed image often presents distortions like noise [15] and blurring [16].

Although digital watermarking is a well-established area that mostly targets color and grayscale images (wide range of intensity levels) [17–19], *hardcopy* watermarking is still a challenging area. In particular, distortions introduced by the print-and-scan (PS) process make the task of transmitting data using hardcopy watermarking more difficult [20].

Most works in this area focus on making the embedded information more robust to distortions of the PS channel. Methods differ from each other in terms of efficiency, capacity, and robustness. For example, Brassil et al. [21] have proposed an authentication method that is based on shift coding. To increase robustness, their method requires the use of uniformly spaced centroids, which are often difficult to obtain. Tan et al. [22] extended this method using a directional modulation technique for watermarking Chinese text images. More recently, other methods have been proposed for specific applications [23–26].

Among hardcopy watermarking methods, those that embed information into binary images are particularly interesting because pixel binarization is the last step of the printing process. Also, during the scanning process, the data is first acquired as a halftone. Therefore, restoring the watermark from the halftoned

* Corresponding author.
E-mail address: sawp@sawp.com.br (P.G. Freitas).

http://dx.doi.org/10.1016/j.image.2016.08.007
0923-5965/

image instead of its inverse (grayscale image obtained from the halftone using an inverse halftoning algorithm) increases the robustness of the watermarking the PS process. On the other hand, embedding data in binary images is often more difficult than embedding data in color or grayscale images [27]. As stated by Hou et al. [27], challenges include a limited data hiding capacity and the introduction of noticeable artifacts. Although more challenging, the demand for this kind of technique is high and few techniques have already been developed [28–32].

Several techniques have been proposed with the goal of increasing the embedding capacity for binary images. For example, Pan et al. [33] proposed a low-capacity watermarking scheme for halftone image authentication that uses an image hash as a fragile watermark. Pan et al. [34] also developed a reversible data hiding method for error diffused halftone images. Guo and Liu [35] developed a higher capacity watermarking technique that uses a block truncation code. Son and Choo [36] proposed a watermarking method for clustered halftone dots in which the embedded binary data is recovered using dictionary learning. Guo et al. [37] proposed a halftoning-based approach capable of embedding watermarks using direct binary search to encode the binary data. Guo and Liu [38] proposed a method for embedding a multi-tone watermark that produces a lower quality watermarked halftone image. Although these methods have a reasonable data hiding capacity, they are restricted to a specific type of dithering that limits their performance and application.

The aforementioned Guo's method [38] embeds a grayscale watermark into a halftone image, but it does not covert the embedding of color information. In a recent work, Son et al. proposed techniques [39,40] to insert (and restore) color channels into black-and-white watermarked halftone images. The work is an important contribution to the area of reversible color-to-grayscale mapping [41–43]. Watermarking techniques use color-to-grayscale mapping to recover color channels from watermarked grayscale images submitted to a PS process. This application requires a large data hiding capacity because two chrominance images (color channels) are embedded into a halftone version of the luminance channel [44].

In this paper, we propose a binary image watermarking method that uses maximum-similarity binary patterns. The method embeds color watermarks into dithering patterns of halftone images. Unlike the methods of Son et al. [39,40], which embed two color channels into the corresponding halftoning version of the luminance channel, our method embeds the three RGB channels into it. In other words, the method is capable of embedding any content into the host halftone image. No relation between host and watermark contents is required, which is a significant improvement compared to other methods in the literature.

The rest of this paper is organized as follows. Section 2 describes the proposed method, detailing the parallel algorithms for embedding and recovering watermarks. Experimental results are presented in Section 3. Finally, conclusions are drawn in Section 4.

2. Proposed method

Fig. 1 displays a block-diagram illustrating the proposed halftone-based watermarking method that enables embedding a color image into a binary black-and-white image. The method decomposes the RGB color channels of a watermark image (W) into three binary channels. Each color channel of W is treated as a grayscale image and a halftoning algorithm is used to generate $R = \{r_{ij}\}$, $G = \{g_{ij}\}$, and $B = \{b_{ij}\}$, where $r_{ij}, g_{ij}, b_{ij} \in \{0, 1\}$. Using a combination of these binary pixels, we generate the watermark and embed the host halftone. The method involves three steps: encoding, embedding, and restoring the color watermark.

2.1. Watermark encoding

We encode the color channels in a binary format, creating a common representation that can be used in the watermark embedding and restoration steps. The encoding algorithm creates a set of references, composed of binary vectors representing a subset of the halftone binary information. More specifically, a binary vector Z with n dimensions is defined as:

$$Z = \{z_1, z_2, \ldots, z_n\}, \tag{1}$$

where $z_k \in \{0, 1\}, \quad \forall\, k \in \{1, 2, \ldots, n\}$. These binary vectors are generated by computing the finite n-ary Cartesian product of the set X, which is defined as:

$$X^n = \prod_{k=1}^{n} X = \left\{ (x_1, \ldots, x_n) : x_k \in X, \forall\, k \in \mathbb{N}_n^* \right\}, \tag{2}$$

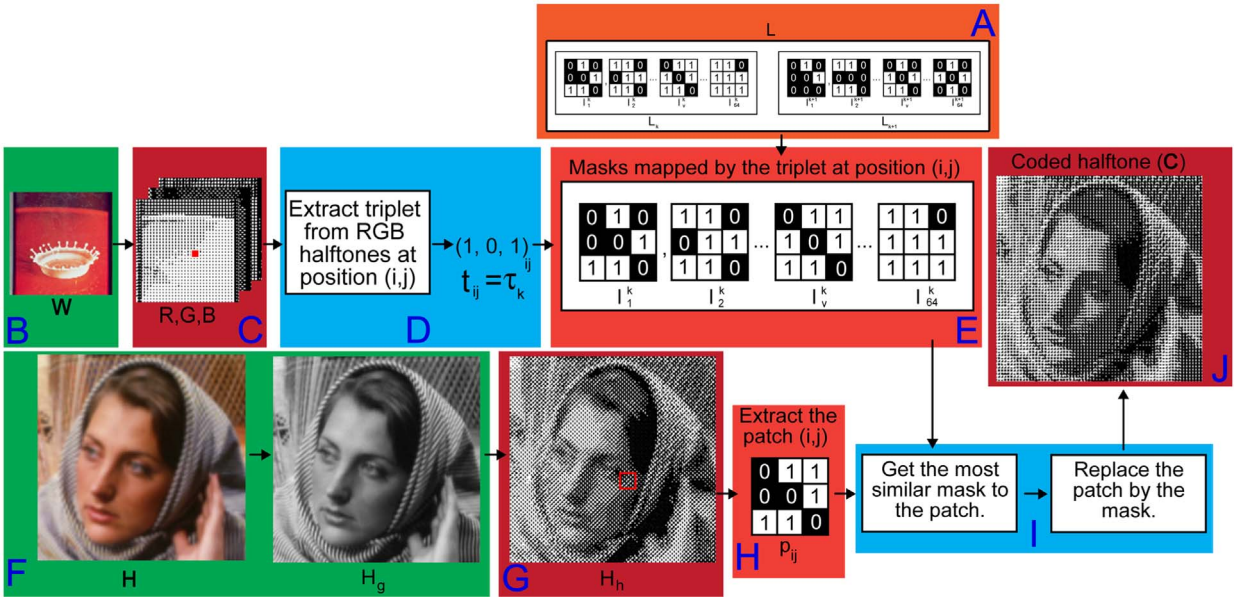


Fig. 1. Steps for embedding the color watermark into the halftone image.

where X^n is a set of all ordered n-tuples $\{x_k\}$ and each element x_k is a basis element of X^n. For $X \in \{0, 1\}$, we have 2^n distinct n-tuples. Each tuple is equivalent to a binary vector (as defined in Eq. (1)) and is used to encode the combinations of each pixel of a halftone RGB channel. To represent each pixel as a binary tuple, we set $n=3$, given the three R, G, and B color channels of a pixel. Therefore, each pixel is coded as 3-tuple or a triplet, so, there are $2^3 = 8$ possible triplet combinations.

We use Eq. (2) again to compute a larger set of tuples with the goal of representing the local distribution of halftones. The higher number of tuples, the larger the distribution of distinct dots, which means that there are more options to represent the distribution of the original pixels. On the other hand, the more distinct tuples there are, the more space is required to represent them. For simplicity, we adopt $n=9$, what gives a total of $2^9 = 512$ distinct nonuples corresponding to 512 distinct and unique distributions of halftoning points. We distribute these 512 sets into eight subsets and associate each of these subsets to a triplet, i.e.:

$$\begin{aligned}&\{0, 0, 0\} \mapsto \{\{0, 0, 0, 0, 0, 0, 0, 0, 0\}, \{0, 0, 0, 0, 0, 0, 0, 0, 1\}, \ldots\}, \{0, 1, 0\}\\ &\quad \mapsto \{\{0, 0, 0, 0, 1, 0, 1, 1, 0\}, \{0, 0, 0, 0, 1, 0, 1, 1, 1\}, \ldots\}, \quad \vdots \{1, 1, 1\}\\ &\quad \mapsto \{\{1, 1, 1, 1, 1, 1, 1, 1, 1\}, \{1, 1, 1, 1, 1, 1, 1, 1, 0\}, \ldots\}.\end{aligned}$$

Therefore, each triplet maps a set of binary sets. More specifically, each triplet t_k maps a set of 64 distinct nonuplets $L_k = \{l_1^k, l_2^k, \ldots, l_{64}^k\}$ using a hash map data structure. The nonuplets L_k are geometrically distributed in 3×3 matrices called 'masks'. This mapping is used to represent the halftoning color channels of the watermark in terms of the binary representation of host (watermarked), as described in the next section.

2.2. Watermark embedding

After generating the set of masks using the nonuplets L_k, we map each pixel of the RGB color planes into a mask. We compute the halftone of each color plane independently, generating R, G, and B. As shown in Fig. 2, we extract the binary value of each halftone channel to generate the triplets. Then, for each pixel of these planes, we extract a triplet:

$$\tau_{i,j} = \{r_{ij}, g_{ij}, b_{ij}\}, \tag{3}$$

where $r_{ij}, g_{ij}, b_{ij} \in \{0, 1\}$. These triplets are compared with the triplets used as a key to map a set of masks. Then, we choose the set of nonuples L_k such that $\tau_{i,j} = t_k$.

After each pixel of W is mapped into a set of binary masks, we choose the masks to use. We compute its halftone H_h from its grayscale H_g, and slice H_h into $M \times N$ patches p_{ij} of 3×3 pixels.

Now, we have two sets: $\{p_{ij}\}$, extracted from data, and $\{\tau_{i,j} \mapsto L_k\}$, obtained by assuming $\tau_{i,j} = t_k$ in the encoding stage. For each p_{ij}, we choose one of the 64 available masks by solving the following optimization problem:

$$\underset{Y}{\text{argmax}} \quad S(X, Y) \text{subject to} \quad X = p_{ij} Y \in L_k, \tag{4}$$

where S is a similarity measure that is calculated between $X \in \Omega$ and $Y \in \Omega$, and Ω is the set of all n-dimensional binary vectors (as defined in Eq. (1)). Since this maximization problem is performed for each patch, the encoded halftone image C is built by placing the computed masks at the corresponding positions of the host halftone image (H_h). There are 76 measures that can be used to evaluate the similarity, $S(X, Y)$, between X and Y [45]. Among these measures, the most popular are Dice, Jaccard–Needham, Sokal–Sneath, Kulzinsky, Rogers–Tanimoto, Sokal-Michener, Russell–Rao, and Yule [46,47].

2.3. Processing similarity measures on GPU

Initially, both images (host and watermark) are read into the main memory. Then, the halftoning version of these images is computed in the CPU or GPU. For this paper, we processed the halftoning algorithm in CPU because the performance bottleneck of the proposed method resides in the optimization step (block I of Fig. 1). After generating the halftone representations, the host image is divided using a grid with $\frac{nB}{mT}$ blocks, where B is the number of CUDA blocks and T is the number of CUDA threads in each block. Each CUDA thread is responsible for processing m patches and, therefore, each block processes $m \cdot T$ patches. It is worth pointing out that a patch extracted from the host halftone is associated with a triplet generated from the watermarking halftone. Each triplet is associated with a set of masks. Therefore, depending on the size of host, a buffer is created to store the patches since they can be processed independently from each

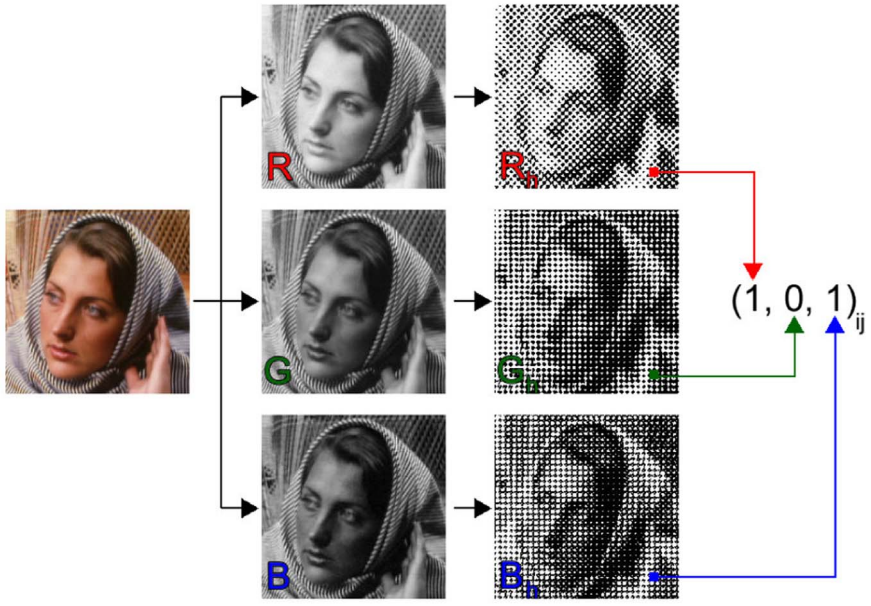
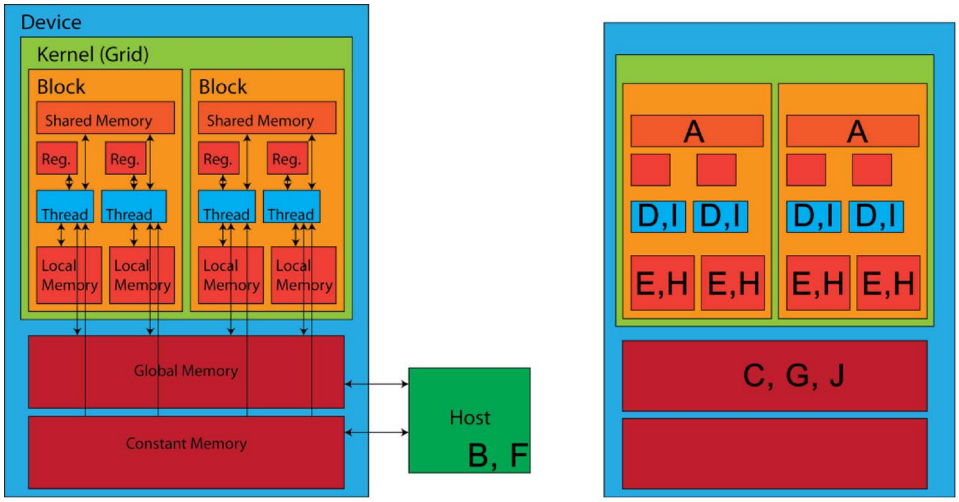


Fig. 2. Extraction of a triplet used to map a halftone mask.



(a) Memory architecture of CUDA device. (b) Location of data structures and threads.

Fig. 3. Memory organization of proposed algorithm's data structures.

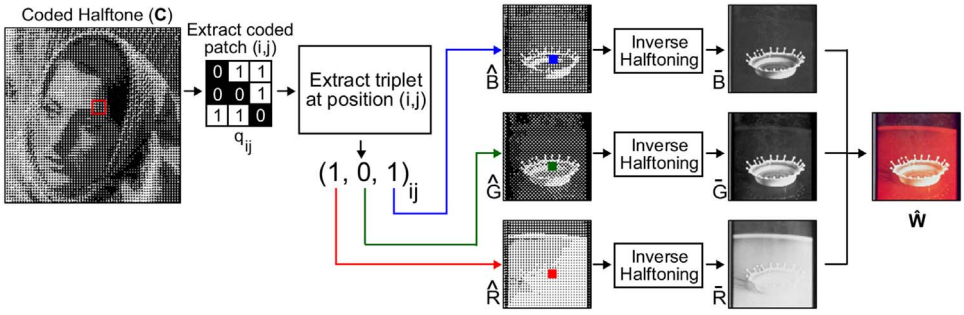
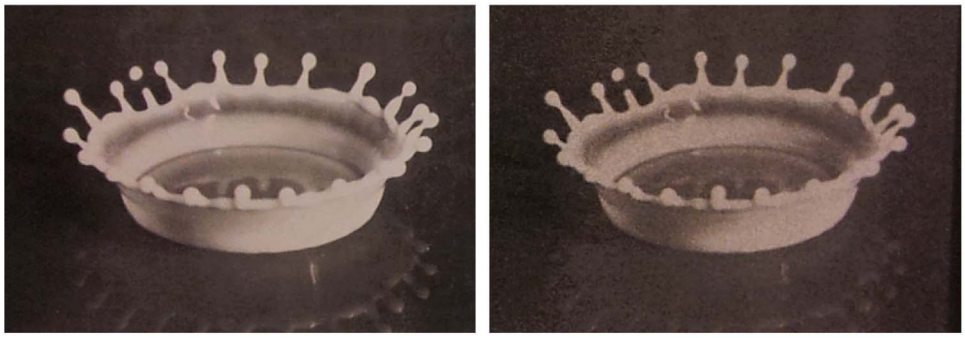


Fig. 4. Extraction of color watermark from encoded halftone.



(a) Original (unmarked) (b) Watermarked

Fig. 5. Printed grayscale halftone versions of 'Splash': no color watermarking (a) and watermarking the color channels (b).

other.

Fig. 3(a) depicts the memory organization of a GPU device (CUDA compatible), which includes registers, shared memory, global memory, and constant memory. The local memory depicted in this figure refers to the local memory which is reserved by a thread in CUDA block. In CUDA, only threads and the host can access memory. This architecture is adequate for parallel problems that lack memory isolation among distinct threads. In general, the required operations performed by each thread are simple. Therefore, the proposed algorithm can be written in a parallel implementation using CUDA.

Fig. 3(a) and (b) depicts the parallel processing scheme used in this work, with Fig. 3(a) showing the memory architecture and Fig. 3(b) the location of the data structures. The characters $\{A, B, \ldots, J\}$ correspond to the intermediate steps of the proposed algorithm, showing the steps of the execution flow (Fig. 1). More specifically, the hash table containing the trained masks (A) is replicated into the shared memory of each block (collection of multiprocessors). We replicate this data structure to prevent the intensive I/O operations from harming the parallel performance. The color channels of the watermarking image (B) are decomposed into three distinct halftones and stored into the GPU global memory. In parallel, each thread gets a subset of pixels and computes their corresponding triplets (D). Using these triplets, each

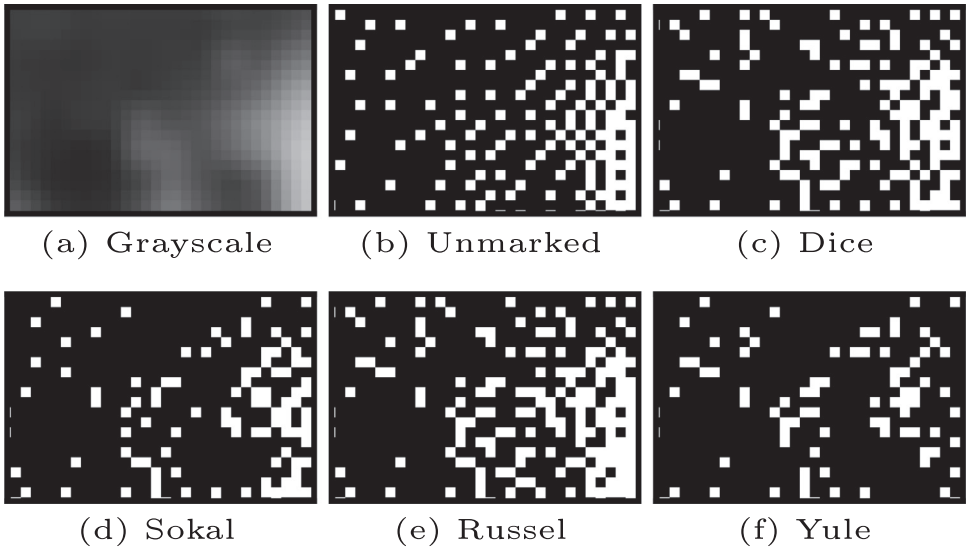


Fig. 6. Visual effect of similarity measure on marked halftones.

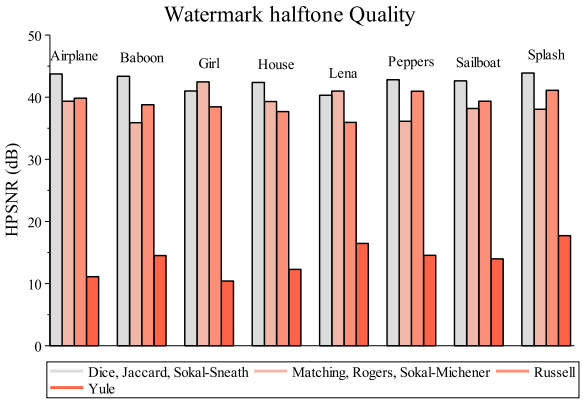


Fig. 7. HPSNR values comparing marked halftone and original grayscale images of D1.

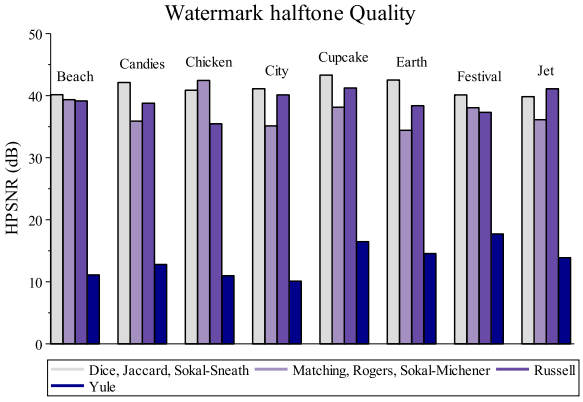


Fig. 8. HPSNR values comparing marked halftone and original grayscale images of D2.

thread copies the subset of mapped masks from the shared memory block to the local memory thread (E). After computing the host halftone on the CPU (F) and loading it into the GPU global memory (G), the threads load the patches into their local memories (H). Since the patches and the corresponding set of masks are loaded into the thread local memory, Eq. (4) is solved in parallel (I). After all threads are finished, the result is stored in the global memory and returned to the host (J).

2.4. Watermark extraction and restoration

The decoding process is depicted in Fig. 4. First, the encoded halftone C is sliced into $M \times N$ patches q_{ij} of 3×3 pixels. For each patch extracted from C, we recover the triplet $\hat{\tau}_{i,j}$ using the inverse mapping of L_k. In other words, we take $\hat{\tau}_{i,j} = t_k$ given that $t_k \mapsto L_k$ and $q_{ij} \in L_k$. In this manner, instead of applying the inverse halftone algorithm to recover grayscale levels, the information is extracted directly from the dot pattern of the printed image. Therefore, each restored triplet contains the bits of restored channels at position i, j:

$$\hat{\tau}_{i,j} = \{\hat{r}_{ij}, \hat{b}_{ij}, \hat{g}_{ij}\}, \tag{5}$$

where $\hat{r}_{ij}, \hat{g}_{ij}, \hat{b}_{ij} \in \{0, 1\}$. Then, we distribute the recovered bits to the respective halftone color channels $\hat{R}$, $\hat{G}$, and $\hat{B}$.

After recovering the halftone versions of the RGB channels, we use an inverse halftoning algorithm to restore the multilevel representation of these color channels: $\bar{R}$, $\bar{G}$, and $\bar{B}$. Finally, the color watermarked image $\hat{W}$ is produced by combining the restored channels.

3. Results

The proposed method was tested using two image datasets. The first dataset (D1) contains eight 512×512 natural color images, taken from the "Miscellaneous" set of USC-SIPI Image Database [48]. The second dataset (D2) consists of eight 4K Ultra High Definition (3840×2160) color images collected from the Internet. The original images of D1 and D2 are depicted in first column of Figs. 10 and 11, respectively. These images are free of copyright, labeled for reuse with modification, and have different visual properties (color ranges, contrast, content, etc.). To process these images, we used a laptop with an Intel i7-4700mq processor, 32 GB of RAM, and a Nvidia GeForce GTX 770m running Windows 7 Professional Edition. The code was written in C++ using CUDA 7 and it was compiled with MS Visual C++ compiler. The Thrust Framework 1.8 [49] was used to implement the interface between the GPUs and a CPU. A multi-functional (printer and scanner) Aficio MP C4501 was used to test the print results. Pictures in Fig. 5

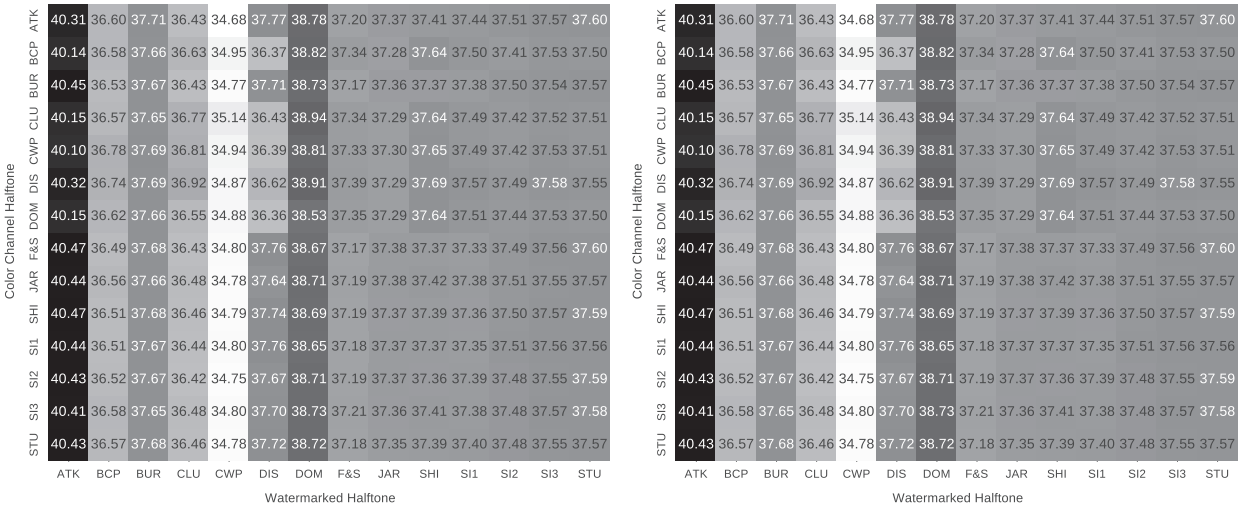


Fig. 9. Pairwise comparison between coded color halftones and watermarked grayscale halftone. In each table, the horizontal axis corresponds to the algorithm used to compute the host halftone image, while the vertical axis corresponds to the algorithm used to compute the color channel halftone. The darker blocks represent the combinations that provide the highest HPSNR values.

were taken using the camera of a Samsung Galaxy Note N7000.

The proposed method is able to embed any content, related or not to the host image. However, in the reported simulations, we used the same image content for both image and host. The tests involved recovering the RGB color channels from a printed halftone that was previously watermarked. In other words, we restricted our comparison to 'color-to-gray-and-back' methods.

3.1. Visual quality of watermarked halftones

We tested the visibility of possible degradations caused by the embedding watermarks generated using the Floyd–Steinberg halftoning algorithm [9]. As a comparison criterion (S) for the optimization problem of Eq. (4), we used Dice, Sokal–Michener, Russel, and Yule similarity measures [46]. Fig. 6 shows the visual difference produced by these distinct similarity measures on a zoomed area of the 'Lena' image. Fig. 6(a) shows the unmarked grayscale image and Fig. 6(b) shows the corresponding halftone image generated using Floyd–Steinberg [9] (no watermarking). Figs. 6(c)–(f) show the results of embedding color watermarks into the halftones using Dice, Sokal–Michener, Russel, and Yule similarity measures, respectively.

Fig. 5 shows a photo of the printed results for the 'Splash' image. While Fig. 6 shows details of the dithering changes when the host is watermarked, in Fig. 5 the picture was taken from a distance and, therefore, the black-and-white points are perceived as grayscale. In other words, even though Fig. 6 shows that the binary points are locally rearranged to store the watermark in halftone images, in Fig. 5 we notice that the content of the final watermarked halftone is preserved, although the image is slightly noisy.

Most image quality metrics are designed for multi-level images (grayscale or color) and cannot be used to assess the quality of binary images (halftones). So, even though objective quality assessment methods for halftone images can be used to design better printing and imaging systems [50], little work has been done in this area. In this work, we use the human visual peak signal to noise ratio (HPSNR) [11,51] to assess the quality of halftone images. HPSNR is basically a weighted version of the PSNR, which exploits the perceptual limitations of the HVS to determine if the dot patterns are being perceived as continuous gray levels. It is calculated using the following equation:

$$\mathrm{HPSNR}(x, y) = 10\log_{10}\frac{255^2 \cdot M \cdot N}{\sum_{i=1}^{M}\sum_{j=1}^{N}\delta(x_{ij}, y_{ij})^2}, \tag{6}$$

where

$$\delta(x_{ij}, y_{ij}) = \sum_{m,n} g_{m,n}(x_{i+m,j+n} - y_{i+m,j+n}),$$

x and y are the original and altered images, respectively, and $g_{m,n}$ corresponds to the Gaussian filter that simulates the lowpass characteristic of HVS.

Figs. 7 and 8 show the HPSNR values between marked halftones and their corresponding original grayscale images for different similarity measures. Figs. 7 and 8 correspond to images in datasets D1 and D2, respectively. From these figures, we can notice that HPSNR values for Dice, Sokal–Michener, and Russel measures produced the best results, suggesting that watermarked halftones have an acceptable quality. The only exception is the watermarked halftone obtained with the Yule similarity measure, which has a very low HPSNR. These results are in agreement with the visual results, as depicted in Fig. 6.

Fig. 9 shows the HPSNR pairwise comparison tables of the tested halftoning algorithms for the images Lena and Splash. In each table, the horizontal axis corresponds to the dithering algorithm used to compute the host halftone image, while the vertical axis corresponds to the dithering algorithm used to compute the color channel halftone. The darker blocks represent the combinations that provide the highest HPSNR values. From this figure, we observe that the dithering algorithm used to compute the host halftone has a bigger influence on the HPSNR than the dithering algorithm used to compute the halftone of the color channels. Moreover, the quality of the encoded halftone is higher when Atkinson's algorithm is used (first column of the tables). This pairwise comparison was performed for all tested images and the results obtained for the remaining images are similar, indicating that the performance of the combination of dithering algorithms is not very affected by image content.

3.2. Visual quality of restored watermarks

The performance of the proposed method is compared with two established methods: Queiroz [41] and Ko et al. [42]. To

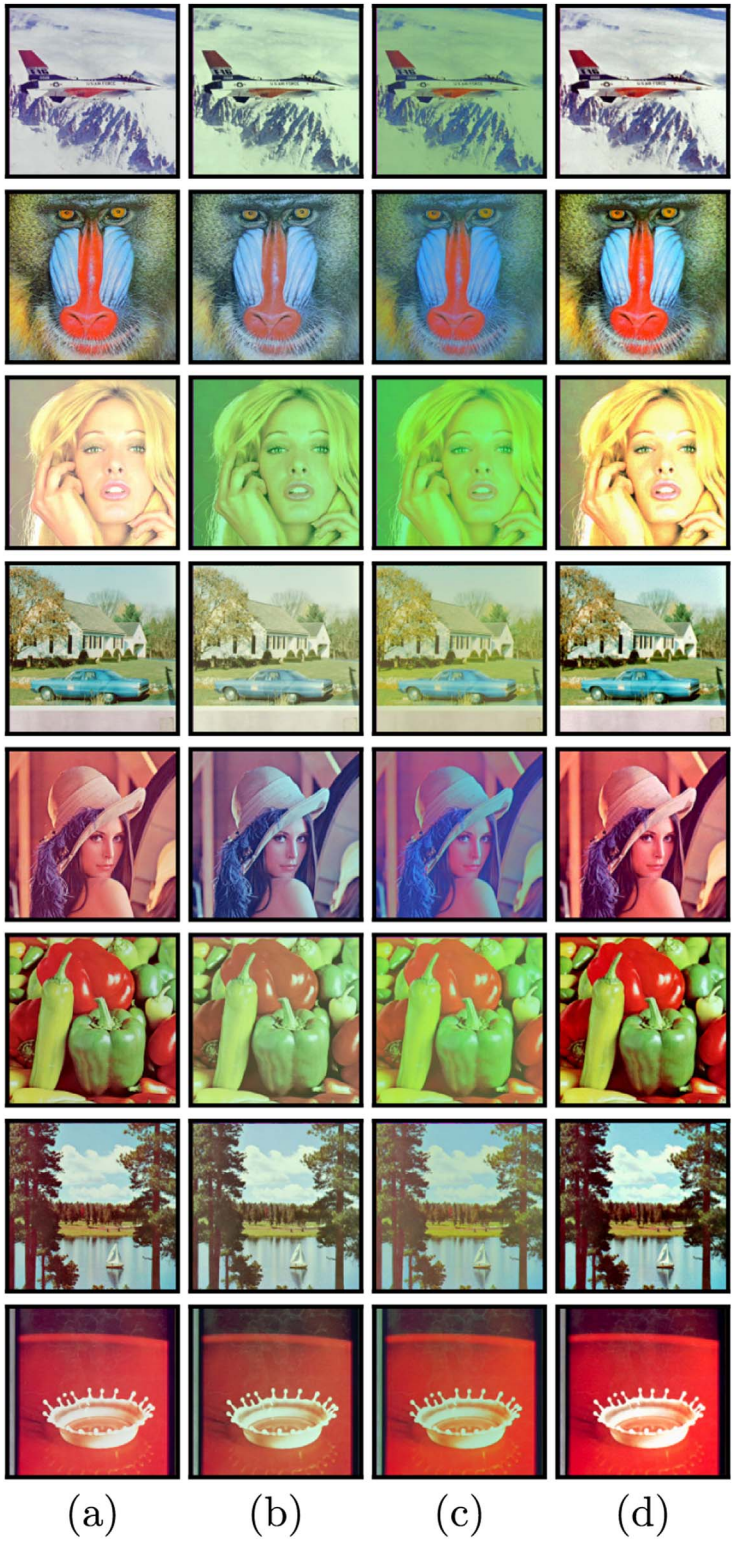


Fig. 10. Comparison of recovered color using images of D1. From top to bottom: Airplane, Baboon, Woman, Car, Lena, Peppers, Sailboat, and Splash. (a) Original, (b) Queiroz's method, (c) Ko's method, and (d) proposed. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

restore the color channels using the proposed method, we used the inverse halftoning algorithm proposed by Kite et al. [16], although other algorithms can be used.

Figs. 10 and 11 show the restored watermarked content of datasets D1 and D2, respectively. In both figures, the first columns are the original images. The second columns show the restoration of color images with Queiroz's method. Likewise, the third columns show the restorations using Ko's method. The last column shows the restoration of the color image using the proposed method.

As expected [41,42], results obtained using Queiroz's and Ko's methods present faded colors, as depicted in Figs. 10 and 11. We can notice that Ko's method produces results with fewer color artifacts, but the colors are still different from the colors in the

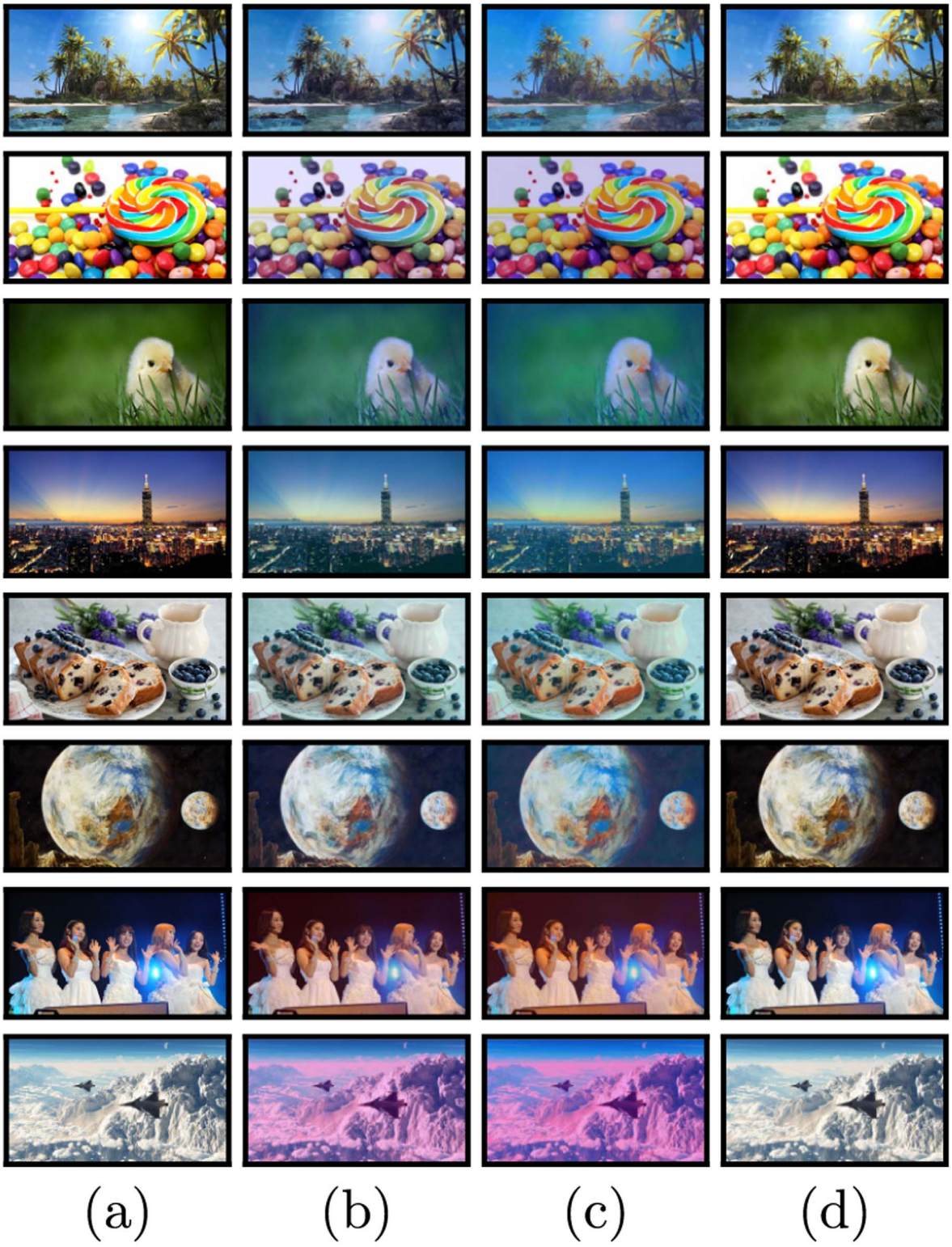


Fig. 11. Comparison of recovered color using images of D2. From top to bottom: Beach, Candies, Chicken, City, Cupcake, Earth, Festival, and Jet. (a) Original, (b) Queiroz's method, (c) Ko's method, and (d) proposed. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

originals. The color distortions are a consequence of the subsampling of the color channels that is performed during the encoding process (watermarking). On the other hand, in the proposed method the RGB channels are equally subsampled (from 24 bits per pixel to 3 bits per pixel) and a good inverse halftoning algorithm is used to restore the content [16]. As a consequence, the proposed method introduces less color distortions, as shown in the third column of Figs. 10 and 11.

We used two objective quality metrics to quantitatively evaluate color distortions. The first metric was the Color Multi-scale Structural Similarity Index (CMSSIM) [52], which was used to measure the overall similarity between original and restored images. The second metric was the CIE color difference measure (ΔE^*) [53] that was used to measure the accuracy of the recovered

Table 1
D1: Objective evaluation of structural similarity between the original color watermarks and the restored watermarks using the tested methods (CMSSIM) and (ΔE^*).

Image	Queiroz		Ko		Proposed	
	CSSIM	ΔE^*	CSSIM	ΔE^*	CSSIM	ΔE^*
Airplane	0.456	17.02	0.336	21.35	0.734	07.52
Baboon	0.525	14.09	0.456	16.55	0.751	10.20
Girl	0.432	24.65	0.357	27.39	0.855	10.37
House	0.758	10.49	0.727	11.95	0.876	06.24
Lena	0.238	20.75	0.243	22.82	0.948	05.40
Peppers	0.782	15.63	0.794	17.15	0.933	05.54
Sailboat	0.803	11.54	0.782	13.37	0.825	08.58
Splash	0.830	9.52	0.854	09.27	0.901	06.47
Average	0.603	15.46	0.569	17.48	0.853	07.53

Table 2
D2: Objective evaluation of structural similarity between the original color watermarks and the restored watermarks using the tested methods (CMSSIM) and (ΔE^*).

Image	Queiroz		Ko		Proposed	
	CSSIM	ΔE^*	CSSIM	ΔE^*	CSSIM	ΔE^*
Beach	0.572	01.12	0.537	02.37	0.868	04.44
Candies	0.578	04.59	0.628	06.53	0.967	00.77
Chicken	0.093	20.97	0.054	23.91	0.971	02.13
City	0.417	06.01	0.347	20.06	0.892	03.63
Cupdake	0.549	09.35	0.461	01.91	0.914	03.40
Earth	0.223	02.87	0.179	05.56	0.777	05.15
Festival	0.383	03.38	0.335	05.16	0.884	03.80
Jet	0.346	21.64	0.286	23.36	0.872	04.13
Average	0.395	08.74	0.354	11.11	0.893	03.43

colors. The better the quality of the reconstructed image, the higher the value of CMSSIM and the smaller the value of ΔE^*.

Tables 1 and 2 show the CMSSIM and ΔE^* values for all test images of datasets D1 and D2, respectively. Notice that the proposed method presents the best performance, in agreement with the qualitative results presented in Figs. 10 and 11. The CMSSIM values presented in these tables suggest that images reconstructed with the proposed method are very similar to the original images in terms of luminance, color, or both. In the same way, the ΔE^* values quantify the difference in performance between proposed and conventional methods, showing that the proposed method provides superior results in terms of structural and color fidelity.

3.3. Computational performance

To evaluate the computational performance of the proposed algorithm, we set the 'Dice' similarity measure as the optimization criterion and the Floyd–Steinberg as the halftoning algorithm. All simulations were performed under the same conditions. For each image, we repeated the simulation 10 times, collecting the encoding and decoding running times, and then calculating the average. This procedure avoids the bias caused by other processes running at the same time in the computer.

Figs. 12(a) and 13(a) show the average time to embed the watermarks into the images of the datasets D1 and D2, respectively. When comparing these graphs with the results shown in Sections 3.1 and 3.2, we concluded that the serial version of the proposed algorithm requires a higher processing time than the other methods. This higher encoding time is a consequence of the optimization problem (see Eq. (4)). Using parallel computing for this encoding stage significantly reduces the overall runtime, while maintaining the same results in terms of visual quality.

The advantage of the parallel implementation can be better observed in Fig. 14. This figure shows the average time to solve Eq. (4) in function of the number of triplets using four configurations: a laptop CPU (Intel i7-4700MQ), a desktop CPU (Intel i5-4460), a laptop GPU (Nvidia GeForce GTX 770m), and a desktop GPU (Nvidia Quadro K4000). We varied the number of triplets and plot the runtime required to complete the encoding step, using a serial or a parallel implementation. From these plots, we see that the CPU time (serial version) curve is always greater than the GPU (parallel) curve. This demonstrates that the proposed parallel implementation has a better performance.

Since the main bottleneck of the proposed algorithm is the optimization problem described in Eq. (4), the decoding algorithm was not parallelized. Average times for decoding algorithm are depicted in Figs. 12(b) and 13(b) . From these graphs, we see that, although the encoding time of the proposed algorithm is higher than the encoding time of the other two methods, the decoding time is smaller because the triplets are recovered using a simple hash table. After recovering the triplets and reconstructing the halftone color channels, the inverse halftone is performed three times.

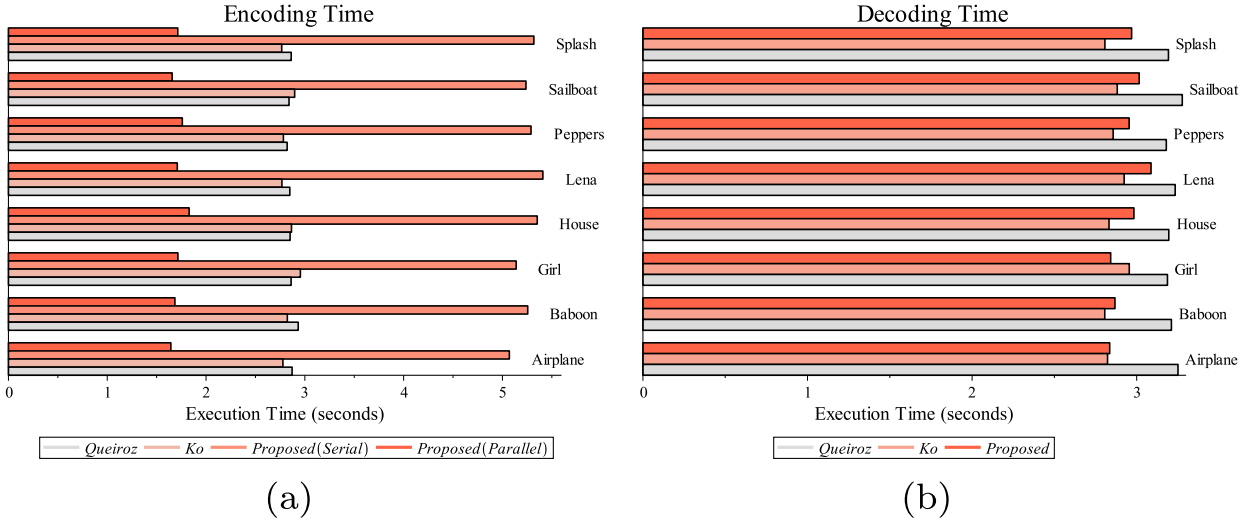


Fig. 12. Performance of tested algorithms using D1: (a) encoding time (watermark embedding), (b) decoding time (watermark extraction).

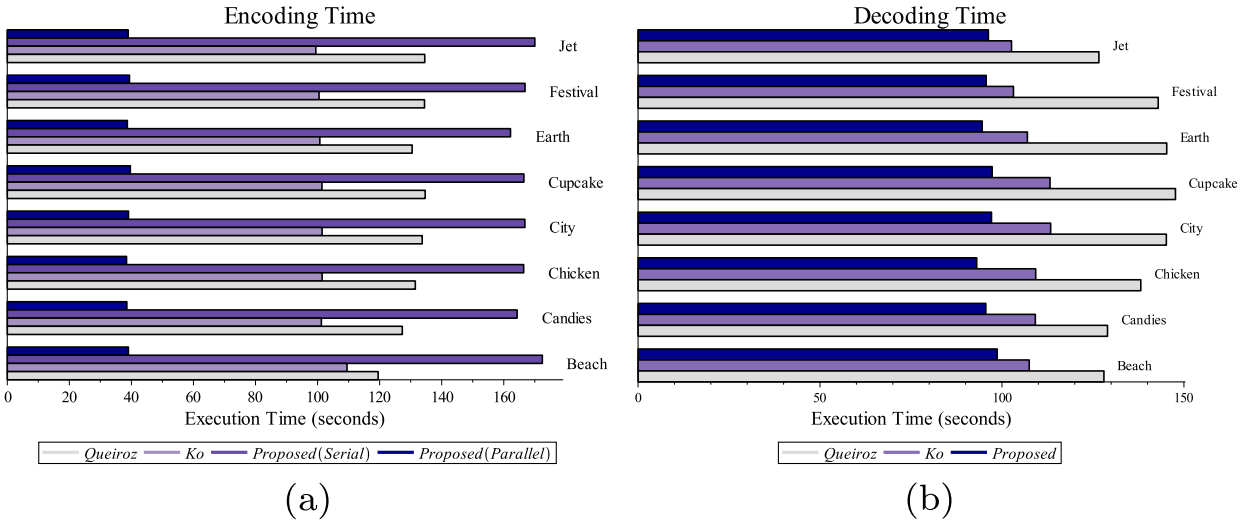


Fig. 13. Performance of tested algorithms using D2: (a) encoding time (watermark embedding), (b) decoding time (watermark extraction).

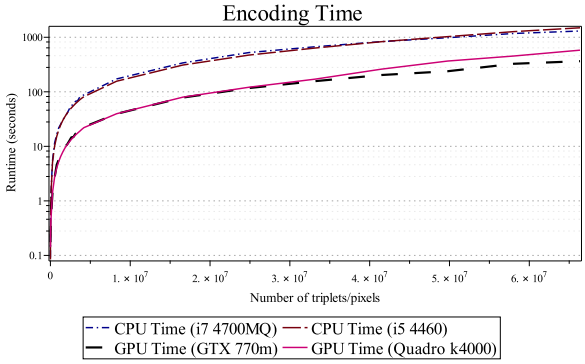


Fig. 14. Runtime of proposed algorithm (parallel on GPU vs serial on CPU) of D2.

4. Conclusions

We have presented a parallel algorithm for embedding a color image watermark into a halftone (binary) image. The proposed algorithm has a higher embedding capacity than the currently available methods. Experimental results show that the recovered colors have a higher fidelity than what is obtained using 'color-to-grayscale-and-back' methods. In addition, the proposed method is not limited to color reconstruction. Since the triplets containing the RGB information are embedded and recovered from the masked halftone, we can use the proposed approach to embed any color content into a binary host image (halftone). The proposed method can be used in applications, such as steganography, hardcopy data hiding, and printed image authentication.

Although the quality of the colors restored with the proposed method is superior, the required processing time using serial processing is high. To reduce the computation time, we proposed a parallel approach to perform the watermark embedding. Results show that the algorithm can be executed efficiently using a simple laptop GPU. However, the proposed algorithm can present further speedups by using multiple or better GPUs. This is possible because the considerable amount of input and output in this problem can be distributed to avoid a centralized I/O.

Future works include implementing the proposed algorithms with multiple GPUs and additional CUDA cores. Architectures with distributed memory can be used to exploit the parallelism of the problem. The algorithm can be executed, for example, in cloud-architectures, Message Passing Interfaces in distributed shared memory clusters, or in Xeon Phi co-processors. Also, the mask construction process is currently not fully optimized. A machine learning algorithm can be used to reduce the execution time for the mask construction, providing an efficient distribution of the mask groups. With optimized mask groups, redundant processing can be avoided and the optimization problem can be simplified.

Acknowledgment

This work was supported in part by Conselho Nacional de Desenvolvimento Cientfico e Tecnológico (CNPq), in part by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), and in part by the University of Brasília (UnB).

References

[1] D.E. Knuth, Digital halftones by dot diffusion, ACM Trans. Graph. 6 (4) (1987) 245–273.
[2] J.S. Viggiano, Modeling the color of multi-colored halftones, in: Proceedings of the TAGA, vol. 42, 1990, pp. 44–62.
[3] S.H. Kim, J.P. Allebach, Impact of hvs models on model-based halftoning, IEEE Trans. Image Process. 11 (3) (2002) 258–269.
[4] X. Zhang, J.P. Allebach, Quad-interleaved block level parallel direct binary search algorithm, Electron. Imaging 20 (2016) (2016) 1–6.
[5] R. Ulichney, The void-and-cluster method for dither array generation, SPIE Millestone Series MA, vol. 154, 1999, pp. 183–194.
[6] X. Liu, Y. Geng, Z.-J. Li, Compression method for ordered dither halftone image, J. Comput. Appl. 1 (2011) 039.
[7] J.F. Jarvis, C.N. Judice, W. Ninke, A survey of techniques for the display of continuous tone pictures on bilevel displays, Comput. Graph. Image Process. 5 (1) (1976) 13–40.
[8] P. Stucki, MECCA: a multiple-error correction computation algorithm for bi-level image hardcopy reproduction, IBM Thomas J. Watson Res. Div. (1981).
[9] R.W. Floyd, An adaptive algorithm for spatial gray-scale, in: Proceedings of the Society for Information Display, vol. 17, 1976, pp. 75–77.
[10] C. Schmaltz, P. Gwosdek, A. Bruhn, J. Weickert, Electrostatic halftoning, in: Computer Graphics Forum, vol. 29, Wiley Online Library, 2010, pp. 2313–2327, http://onlinelibrary.wiley.com/doi/10.1111/j.1467-8659.2010.01716.x/abstract.
[11] J.-M. Guo, Y.-F. Liu, Improved block truncation coding using optimized dot diffusion, IEEE Trans. Image Process. 23 (3) (2014) 1269–1275.
[12] P. Gwosdek, C. Schmaltz, J. Weickert, T. Teuber, Fast electrostatic halftoning, J. Real.-Time Image Process. (2014) 1–14.
[13] M. Mese, P.P. Vaidyanathan, Look-up table (lut) method for inverse halftoning, IEEE Trans. Image Process. 10 (10) (2001) 1566–1578.
[14] Y.-F. Liu, J.-M. Guo, New class tiling design for dot-diffused halftoning, IEEE Trans. Image Process. 22 (3) (2013) 1199–1208.
[15] P.G. Freitas, M.C. Farias, A.P. de Araújo, Fast inverse halftoning algorithm for ordered dithered images, in: 2011 24th SIBGRAPI Conference on Graphics, Patterns and Images (Sibgrapi), IEEE, 2011, pp. 250–257, http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6134739.
[16] T.D. Kite, N. Damera-Venkata, B.L. Evans, A.C. Bovik, A fast, high-quality inverse halftoning algorithm for error diffused halftones, IEEE Trans. Image Process. 9

(9) (2012) 1583–1592.
[17] F.A. Petitcolas, R.J. Anderson, M.G. Kuhn, Information hiding—a survey, Proc. IEEE 87 (7) (1999) 1062–1078.
[18] F. Hartung, M. Kutter, Multimedia watermarking techniques, Proc. IEEE 87 (7) (1999) 1079–1107.
[19] I.J. Cox, J. Kilian, F.T. Leighton, T. Shamoon, Secure spread spectrum watermarking for multimedia, IEEE Trans. Image Process. 6 (12) (1997) 1673–1687.
[20] D. Museleta, A. Trémeau, Recent trends in color image watermarking, J. Imaging Sci. Technol. 53 (1) (2009) 10201–10211.
[21] J.T. Brassil, S. Low, N.F. Maxemchuk, Copyright protection for the electronic distribution of text documents, Proc. IEEE 87 (7) (1999) 1181–1196.
[22] L. Tan, X. Sun, G. Sun, Print-scan resilient text image watermarking based on stroke direction modulation for Chinese document authentication, Radio-engineering 21 (1) (2012) 170–181.
[23] B. Yang, X. Sun, X. Chen, J. Zhang, X. Li, An efficient forensic method for copy–move forgery detection based on dwt-fwht, Journal of Information Security 4 (2) (2013), Article ID: 30057,12, http://dx.doi.org/10.4236/jis.2013.42010.
[24] M. Chroni, A. Fylakis, S.D. Nikolopoulos, Watermarking images in the frequency domain by exploiting self-inverting permutations, in: K.-H. Krempels, A. Stocker (Eds.), WEBIST, SciTePress, 2013, pp. 45–54, http://file.scirp.org/Html/3-7800146_30057.htm.
[25] R. Todmal Satish, K.Thammi. Reddy, A technique to find optimal location for wavelet-based image watermarking using genetic algorithm, Mach. Graph. Vis. 20 (2) (2011) 173–196.
[26] A. Umamageswari, G.R. Suresh, A new cryptographic digital signature for secure medical image communication in telemedicine, Int. J. Comput. Appl. 86 (11) (2014) 4–9.
[27] Q. Hou, J. Dai, L. Li, J. Lu, C.-C. Chang, Scanned binary image watermarking based on additive model and sampling, Multimed. Tools Appl. 74 (21) (2015) 9407–9426.
[28] M. Wu, B. Liu, Data hiding in binary image for authentication and annotation, IEEE Trans. Multimed. 6 (4) (2004) 528–538.
[29] L. Zongqing, Z. Hongbin, A new data hiding method in binary images, in: First International Conference on Innovative Computing, Information and Control, 2006. ICICIC'06, vol. 3, IEEE, 2006, pp. 66–69, http://ieeexplore.ieee.org/document/1692118/.
[30] F. Daraee, S. Mozaffari, Watermarking in farsi/arabic binary document images using fractal coding, in: 2011 8th International ISC Conference on Information Security and Cryptology (ISCISC), IEEE, 2011, pp. 67–72, http://ieeexplore.ieee.org/document/6062334/.
[31] B. He, Y. Wu, K. Kang, W. Guo, A robust binary text digital watermarking algorithm for print-scan process, in: 2009 WRI World Congress on Computer Science and Information Engineering, vol. 7, IEEE, 2009, pp. 290–294, http://ieeexplore.ieee.org/document/5170328/.
[32] H.-C. Wu, Y.-C. Liu, K.-C. Liu, C.-S. Tsai, A block-based authentication watermarking technique for binary images, in: Eighth International Conference on Intelligent Systems Design and Applications, 2008. ISDA'08, vol. 3, IEEE, 2008, pp. 513–518, http://ieeexplore.ieee.org/document/4696519/.
[33] J.-S. Pan, H. Luo, Z.-M. Lu, A lossless watermarking scheme for halftone image authentication, Int. J. Comput. Sci. Netw. Secur. 6 (2b) (2006) 147–151.
[34] J.-S. Pan, H. Luo, Z.-M. Lu, Look-up table based reversible data hiding for error diffused halftone images, Informatica 18 (4) (2007) 615–628.
[35] J.-M. Guo, Y.-F. Liu, High capacity data hiding for error-diffused block truncation coding, IEEE Trans. Image Process. 21 (12) (2012) 4808–4818.
[36] C.-H. Son, H. Choo, Watermark detection from clustered halftone dots via learned dictionary, Signal Process. 102 (2014) 77–84.
[37] J.-M. Guo, C.-C. Su, Y.-F. Liu, H. Lee, J.-D. Lee, Oriented modulation for watermarking in direct binary search halftone images, IEEE Trans. Image Process. 21 (9) (2012) 4117–4127.
[38] J.-M. Guo, Y.-F. Liu, Hiding multitone watermarks in halftone images, IEEE Multimed. 17 (1) (2010) 65.
[39] C.-H. Son, H. Choo, Color recovery of black-and-white halftoned images via categorized color-embedding look-up tables, Digit. Signal Process. 28 (2014) 93–105.
[40] C.-H. Son, K. Lee, H. Choo, Inverse color to black-and-white halftone conversion via dictionary learning and color mapping, Inf. Sci. 299 (2015) 1–19.
[41] R.L. de Queiroz, Reversible color-to-gray mapping using subband domain texturization, Pattern Recognit. Lett. 31 (4) (2010) 269–276.
[42] K.-W. Ko, O.-S. Kwon, C.-H. Son, Y.-H. Ha, Color embedding and recovery based on wavelet packet transform, J. Imaging Sci. Technol. 52 (1) (2008) 10501–10511.
[43] R.L. de Queiroz, K.M. Braun, Color to gray and back: color embedding into textured gray images, IEEE Trans. Image Process. 15 (6) (2006) 1464–1470.
[44] H. Kekre, S. D. Thepade, R. N. Chaturvedi, "color to gray and back" using dst-dct, haar-dct, walsh-dct, hartley-dct, slant-dct, kekre-dct hybrid wavelet transforms, in: Proceedings of the Third International Conference on Soft Computing for Problem Solving, Springer, 2014, pp. 613–623, http://link.springer.com/chapter/10.1007/978-81-322-1768-8_54.
[45] S.-S. Choi, S.-H. Cha, C.C. Tappert, A survey of binary similarity and distance measures, J. Syst., Cybern. Inform. 8 (1) (2010) 43–48.
[46] B. Zhang, S. N. Srihari, Binary Vector Dissimilarity Measures for Handwriting Identification. Proc. SPIE 5010, Document Recognition and Retrieval X, 28 (January 20, 2003); http://dx.doi.org/10.1117/12.473347.
[47] B. Zhang, S.N. Srihari, Binary vector dissimilarity measures for handwriting identification, vol. 5010, SPIE, 2003, pp. 28–38.http://dx.doi.org/10.1117/12.473347.
[48] A.G. Weber, The USC-SIPI image database version 5, USC-SIPI Report 315, 1997, pp. 1–24.
[49] J. Hoberock, N. Bell, Thrust: A parallel template library, version 1.7.0, 2010. URL: ⟨http://thrust.github.io/⟩.
[50] P. Itoua, A. Beghdadi, et al., Objective perceptual evaluation of halftoning using image quality metrics, in: 2010 10th International Conference on Information Sciences Signal Processing and their Applications (ISSPA), IEEE, 2010, pp. 456–459, http://ieeexplore.ieee.org/document/5605446/.
[51] J.-M. Guo, Y.-F. Liu, Joint compression/watermarking scheme using majority-parity guidance and halftoning-based block truncation coding, IEEE Trans. Image Process. 19 (8) (2010) 2056–2069.
[52] M. Hassan, C. Bhagvati, Structural similarity measure for color images, Int. J. Comput. Appl. 43 (14) (2012) 7–12.
[53] G. Sharma, W. Wu, E.N. Dalal, The ciede2000 color-difference formula: implementation notes, supplementary test data, and mathematical observations, Color Res. Appl. 30 (1) (2005) 21–30.