

Voicing of Animated GIF by Data Hiding A Technique to Add Sound to the GIF Format

Sonia Djaziri-Larbi, Awatef Zaien, Sylvie Sevestre-Ghalila

▶ **To cite this version:**

Sonia Djaziri-Larbi, Awatef Zaien, Sylvie Sevestre-Ghalila. Voicing of Animated GIF by Data Hiding A Technique to Add Sound to the GIF Format. Multimedia Tools and Applications, 2015, 10.1007/s11042-015-2491-y . hal-01371344

HAL Id: hal-01371344
https://hal.science/hal-01371344

Submitted on 25 Sep 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Multimedia Tools and Applications manuscript No.
(will be inserted by the editor)

Voicing of Animated GIF by Data Hiding

A Technique to Add Sound to the GIF Format

Sonia Djaziri-Larbi · Awatef Zaien · Sylvie Sevestre-Ghalila

Received: date / Accepted: date

Abstract GIF animations are silent image sequences widely used on the web thanks to their wide support and portability. In this work, we propose an original technique based on data hiding, to add sound tracks in GIF animations. Data hiding is usually used to embed security codes in a host medium to prevent from illegal copying or to protect copyrights (watermarking) or to send secret messages to a dedicated receiver (steganography). We propose to use host GIF images as a "transmission channel" to convey "hidden" sound bits with lowest perceptual image distortion and without altering the wide portability of the GIF format, by means of data hiding. The inserted bits are neither secret nor intended for security issues. They are intended to be played by an audio player synchronously with the GIF player to add sound to the GIF animation. The embedding process is a low complexity, luminance based steganography algorithm, that slightly modifies the pixels colors of the GIF images to insert the sound bits. The extraction of the inserted audio is completely blind: the audio is directly extracted from the pixels of each cover image. The proposed **GIF voicing** was tested with different GIF sequences (cartoons and real scenes) and no audio degradation was reported while a slight, most imperceptible, color modification was noticed in case of an important amount of inserted data. The cover images have undergone objective

S. Djaziri-Larbi
Université Tunis El Manar, Ecole Nationale d'Ingénieurs de Tunis, Signals and Systems Lab, BP37, 1002 Belvédère, Tunis(ia)
Tel.: +216-98901006
E-mail: sonia.larbi@enit.rnu.tn

A. Zaien
CEA Linklab, Telnet Innovation Labs, Pole Technologique El Ghazala, 2083 Ariana, Tunisia
E-mail: zaien.awatef@gmail.com

S. Sevestre-Ghalila
CEA Linklab, Telnet Innovation Labs, Pole Technologique El Ghazala, 2083 Ariana, Tunisia
E-mail: sylvie.ghalila@cea.fr

quality criteria and informal subjective evaluation and has proved to be of good quality.

Keywords GIF Format · Animated GIF · GIF Voicing · Data Hiding · Steganography · Sound Embedding

1 Introduction

The Graphics Interchange Format (GIF) is a bitmap image format that supports animation. It has been introduced with the advent of the World Wide Web to bring more attractiveness to web pages. This format became extremely popular and it is widely supported and established as the default choice on the Web [17] since it was the first graphic file displayed by web browsers [7]. Also GIF files are completely platform independent. Furthermore, since the disclosure of the GIF 89a specifications in 2006, many applications have been developed for GIF sequences manipulation. Powerful optimization algorithms have been developed to further reduce the file size of animated GIF.

Nowadays, the GIF format is again taking over the net with the recent popularity of social networks and smart phones. This is mainly due to its large portability and simplicity, that make its use possible on any browser without plugins [17]. The growing wide spread of the GIF format suggests to think about adding a sound track to **the silent** GIF animations. Indeed, adding sound to the animated GIF may add attractiveness for various applications such as online ready-to-wear advertisement and online movie rental, where customers can access a short video description of the product in different languages, etc.

In this work, we present a technique to add audio tracks to animated GIF by means of data hiding, where the main motivation of using data hiding is to preserve the bitmap of the format in order to maintain the advantage of its portability. The inserted bits are neither secret nor related to any security issue, the data hiding technique is only borrowed to convey the sound information with the GIF file in order to be played synchronously. Indeed, the main applications of data hiding in video and audio are intended for security issues, such as copyright protection and authentication. But several studies modeled data hiding as particular data transmission systems [3], where the channel is the host medium, and it conveys hidden digital information intended for a particular receiver or application. According to this communication model, some data hiding systems have been proposed, where the hidden data conveys information related to bandwidth extension for narrow band speech transmission [18] or for watermark-aided signal processing [4]. In [2], the authors list a variety of steganography applications which are different from the conventional ones, such as TV broadcasting, video-audio synchronization, referencing patients data in medical imaging systems, etc. In [16] the authors propose to hide speech signals in video using source and channel coding of the speech which is then embedded in the coefficients of the orthogonal wavelet transformed frames.

The proposed method for voicing GIF animations lies within the latter data hiding framework, which is different from the conventional watermarking and steganography concepts: the embedded audio bits, represent a kind of side information dedicated for an audio player which extracts the received sound bits from the GIF images and plays the sound synchronously with the GIF player. The advantage of using this concept to embed sound in GIF animations consists in saving bit-rate (as the electronic file size of the GIF file is not increased) and in avoiding the development of new data containers, i.e. the cover data format remains unchanged. On the other hand, the sound embedding modifies the pixels colors and hence inevitably introduces modifications to the images, which must be constrained to remain as imperceptible as possible.

According to the conventional definitions of steganography and watermarking [2, 8, 15], the proposed voicing method is rather based on a steganography approach as 1) the message must be invisible (hidden) and 2) the embedding capacity should be as high as possible and 3) the original image is not available at extraction. However, the proposed application differs from conventional steganography concerning the objectives: the sound bits are hidden in the images to not alter the cover image nor the file format and not for secrecy reasons as it is the case in conventional steganography.

The constraints on the proposed voicing technique are minimal image and sound distortion in order to preserve the perceptual quality, maximal embedding capacity to enable the insertion of the corresponding sound, and low complexity of the embedding/extraction techniques for high processing speed. In this paper, the used data hiding technique is the Fridrich palette-based algorithm [5], developed initially for image steganography.

The paper is organized as follows. In section 2 we summarize the characteristics of the GIF format. Section 3 is dedicated to basics of palette-based data hiding and to the proposed voicing embedding/extraction algorithms, which performance is presented and discussed in section 4. Finally, in section 5 we give a description of the developed application; a GIF player for the proposed voiced GIF animations.

2 Inside the GIF Format

GIF is an indexed color image format licensed by CompuServe in 1987 and 1989. It supports up to 8 bits per pixel, allowing a single image to reference a palette of up to 256 colors, selected from the 24-bit RGB color space. The limited range of colors makes the format more suitable for simple graphics, such as black and white images, line drawings and small text. GIF also supports animations and allows a separate palette of 256 colors for each frame. An animation contains a set of images screened sequentially giving the illusion of motion. Animated GIF is generally used for small animations and low-resolution film clips. Besides, powerful optimization algorithms have been proposed to reduce the size of animated GIF, which consist in comparing consecutive frames of



Fig. 1 Transparency optimization: the upper image sequence (I_1, I_2, I_3) is not optimized and the lower sequence (J_1, J_2, J_3) is transparency optimized (Copyright Telnet).

a GIF sequence and discarding pixels that do not change across frames. The three optimization algorithms, from least to most compression, are presented in the following [11].

2.1 Dirty rectangle optimization

Dirty rectangle optimization consists in the minimum bounding box method, it involves cropping frames in a GIF animation to their smallest needed rectangle. These frames are then played one superimposed on the other, using pixel coordinates for placement.

2.2 Transparency optimization

This optimization method consists in making redundant pixels transparent after having applied the *dirty rectangle* optimization. One index in the color map is designated as "transparent," and all pixels in the image that are painted that color will be transparent when viewed in a browser, as displayed by Fig. 1.

2.3 LZW optimization

LZW[1] optimization is based on the *transparency optimization* and LZW compression, it optimizes the coding of transparent pixels. Using these optimization algorithms, it is possible to reduce significantly the size of an animation that contains redundant elements.

[1] LZW: Lempel-Ziv-Welch, lossless data compression algorithm.

3 Voicing of Animated GIF through Data Hiding

The goal of this paper is to add sound tracks to animated GIF sequences using data hiding, where the embedded data is the digital sound in binary representation. In the following, we emphasize the constraints and objectives of the proposed voicing technique, then we present the main data hiding techniques for palette-based images, and finally we detail the retained method, the Fridrich's algorithm [5].

3.1 Constraints and Objectives of the Proposed GIF Voicing

A wide range of data hiding algorithms for images exist. As stressed in the introduction, the objective of the proposed voicing technique is limited to the embedding of sound data in the corresponding GIF images: no secrecy nor security issues are relevant. The embedded sound is intended to be played synchronously with the associated GIF animation. The suitable choice for the proposed application is constrained by the following requirements:

- Minimal perceptible distortion of the host GIF: embedding data in the GIF frames will undoubtedly introduce distortion, which may be assessed by objective criteria such as the PSNR[2]. Objective distortion may be important while the perceptual, i.e. visible, distortion which reflects the subjectively perceived degradation, is low, and vice versa.
- High embedding capacity: Indeed, the sound needs a relatively high bitrate to render a good perceptual quality. For example, CD[3] Quality corresponds to a bitrate of 705 kbps[4]. If the GIF animation has a frame rate of 5 frames per second, then ca. 141 kbpf should be embedded in each frame. This represents a huge amount of data, and depending on the frames definition, this is likely to introduce visible distortion. Thus, we will use audio compression techniques to reduce the amount of embedded sound bits (cf. Table 1.
- No audible distortion of the extracted sound: The embedded sound bits may be modified if the cover images undergo some attacks. But the proposed application is not a target for pirates, and thus this situation will not be considered. Other image processing manipulations are discussed in section 4.
- Low complexity of the embedding and extraction processes.

3.2 Data Hiding Techniques for Animated GIF

In the literature, two main approaches for embedding data in palette-based images have been described. The first approach embeds data bits in the palette

[2] PSNR: Peak Signal to Noise Ratio
[3] CD: Compact Disc
[4] kbps: kilo bit per second.

[9]. The main advantage of this approach is its simplicity. An example of such a technique is the freeware Gifshuffle [12]; the algorithm shuffles the color entries and uses different combinations of color entries to hide messages. The host image remains visibly intact, only the colors order in the palette is changed. However, the embedding capacity is limited by the size of the palette. The second approach embeds the data bits into image data. This strategy has higher embedding capacity, but it is more challenging to develop such an embedding technique without introducing perceptible distortion to the host image. The most popular of these techniques are probably the EZ stego method [13] and Fridrich's parity based method [5]. The former is based on sorting the palette by luminance and substituting the least significant bits of the pixels indices pointing to the reordered palette. However, this method does not always generate high quality stego-images, mainly because the colors with similar luminance values may be relatively far from each other. Fridrich's steganography method embeds a data bit into each pixel of the host image by searching for the closest color entry in the palette with the desired parity bit. Fridrich's method introduces much less distortion into the host image than EZ stego, while enabling for a high embedding capacity, as each pixel of the image may host one bit of data. More recently, different new data hiding methods for palette based images have been proposed, which provide high embedding capacity and promise low, and even no image distortion. Among them we cite [22][23], where the proposed embedding methods rely on multi-bit color-mapping using Depth-First-Search, and [26] where multi-bit color-mapping re-assignment is exploited to convey several hidden bits. In [9], a high capacity embedding method with no image distortion is presented, at the cost of a very limiting condition on the original color map which should not exceed 128 different colors. It is based on replacing two initial adjacent colors of the original color map by a weighted quantization of both initial colors. Another data hiding technique [25] inserts bits in the edge (contour) pixels of the image, supposed to be robust to image transformation. Some other interesting methods make use of color quantization based on iterative optimization of a cost function[24][20].

The majority of the above cited methods are not (or at least at a high cost) applicable to GIF animation unless each frame has its own color map. Indeed, it seems very difficult to optimize a cost function or to optimally re-quantize a single color map for all frames of a GIF animation. This drawback not only increases the electronic size of the GIF animation but also increases the complexity of the embedding algorithm. Furthermore, most of these cited studies do compare the obtained results to those of Fridrich's method [5][6], as it is considered as a reference in palette based image steganography. Hence, and even if some of these recent methods promise a lower image distortion, we have retained Fridrich's method as the embedding technique to insert sound bits in animated GIF, because of its important embedding capacity, low host image distortion, and especially because of its very simple data embedding and extraction processes.

3.3 Fridrich's Algorithm

The algorithm is described in [5] in its simplest variant. For each pixel p_i of the current frame $\mathbf{I}_m$, the color distance between the color (r_i, g_i, b_i) of pixel p_i and all other color entries of the palette (indexed by j) is evaluated by:

$$D_j = \sqrt{(r_i - r_j)^2 + (g_i - g_j)^2 + (b_i - b_j)^2}, \tag{1}$$

where r, g and b refer respectively to red, green and blue color components and i is the pixel index.
Then, starting with the closest color (i.e. smallest D_j, $i \neq j$), a search is performed until the closest color entry is found, which parity bit $(r_j + g_j + b_j)$ mod 2 equals the sound bit $d \in \{0, 1\}$ to be embedded. Hence, the objective is to find the color indexed by j that satisfies:

$$\arg\min_j(D_j) \ / \ (r_j + g_j + b_j) \bmod 2 = d, \text{ and } d \in \{0, 1\} \tag{2}$$

Once this color is found, the color index of pixel p_i is changed to point out to this new closest color. This process is repeated until all sound bits corresponding to a given frame $\mathbf{I}_m$ are embedded. Hence, the highest capacity of each frame equals its definition, i.e. the number of pixels of that frame.

Note that, if the original parity bit equals the sound bit to be embedded then the pixel color is not changed, yet the sound bit is encoded. Thus, after embedding, the number of modified pixels in the host image is lower (by ca. the half) than the number of inserted bits. The extraction of the inserted bits is carried out by computing the parity bit of each pixel of the received image.

In the following, and to emphasize the difference between steganography and the proposed technique regarding the secrecy of the message, the modified GIF images/sequences by sound embedding will be referred to as *augmented* images/sequences.

[5] PCM: Pulse Code Modulation, here sampled at f_s=44.1 kHz and 16 bits/sample.
[6] mp3: MPEG audio coder layer 3.
[7] bpf: bits per frame.

sound format	**PCM**[5]	**MPEG coder**[6]	
sound bit-rate	705.6 kbps (mono)	96 kbps	128 kbps
required capacity $f_r = 5$ frames/s	141120 bpf[7]	19200 bpf	25600 bpf
required frame definition	376 × 376	139×139	160×160

Table 1 Examples of audio bit-rates and audio coders. 3rd row gives the required embedding capacity (in bpf) in case of an animated GIF (without optimization) with $f_r = 5$ frames/s. 4th row gives an approximation of the required image definition.

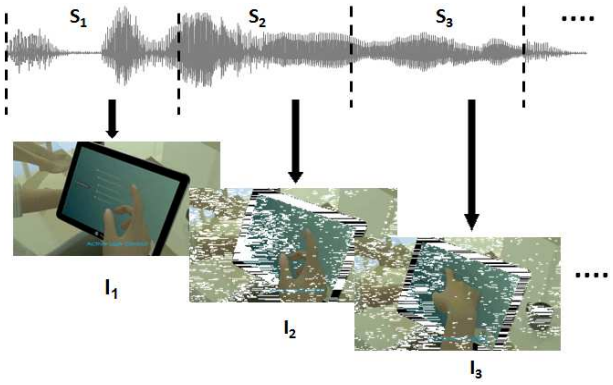


Fig. 2 Principle of sound embedding in animated GIF: the sound vector is subdivided into subvectors S_m. Each of them is embedded in the corresponding image I_m. (Copyright Telnet).

3.4 Voiced GIF Coder: Sound Embedding

Fig. 2 illustrates the principle of the proposed embedding scheme, where the sound is depicted as a waveform vs time index for illustration purposes. Yet the sound is not embedded as a waveform in the image: It is firstly preprocessed to obtain a bitstream vector with a suitable size to be then inserted in the GIF images by means of the data hiding process described above.

3.5 Sound pre-processing

The digital sound to be embedded is available as a bitstream vector, containing L bits. The total size L of this vector for a given sound track (with given duration) depends on its bit-rate in kbps[8]. The bit-rate itself depends on the audio coder used to compress the sound. Table 1 gives examples of sound bit-rates depending on the used audio coder (compression). The bit-rates of Table 1 correspond to monophonic audio. The 3$^{\text{rd}}$ row of Table 1 gives an example of the required embedding capacity, which is the number of sound bits to be embedded in each GIF frame of an animated sequence having a frame-rate f_r= 5 frames/s. Knowing that the highest embedding capacity equals the frames definition, it is clear from this example that the use of an audio coder is necessary to reduce the size of the sound bits in order a) not to exceed the maximal embedding capacity and b) to reduce the amount of embedded data to keep the image distortion imperceptible.

Besides, when the host GIF sequence has been optimized using the transparency algorithm (cf. section 2), the highest embedding capacity varies across frames: it becomes lower than the image definition when the considered frame

[8] kbps: kilo bits per second.

contains transparent pixels, as those pixels are excluded from the embedding process.

3.6 Sound embedding

The digital sound track to be embedded, which has the same duration as the animated GIF, is a bitstream with length L bits. It is decomposed into M sub-vectors $\underline{s}_m,\ m = 1, \cdots, M$, where M is the number of frames $\mathbf{I}_m$ of the host GIF sequence. Each sub-vector $\underline{s}_m$ is embedded in the corresponding image $\mathbf{I}_m$ in a chronological way using the Fridrich's algorithm: all bits of $\underline{s}_m$ are embedded in the pixels of frame $\mathbf{I}_m$. This procedure is described in Fig. 3 and Fig. 4, where c_m denotes the maximal capacity of frame $\mathbf{I}_m$. Indeed, as emphasized previously, optimized animated GIF results in variable embedding capacity as transparent pixels can not host hidden data.

The embedding pattern may be a simple row by row one, as well as a pre-defined pattern, zigzag or chess for example.

3.7 Voiced GIF Decoder: Sound Extraction

The decoding process (cf. Fig. 5) is very simple: the extracted sound bits $\hat{d}_k$ are equal to the parity bits of the pixels colors of each host image, sequentially and according to the embedding pattern:

$$\hat{d} = (\hat{r}_j + \hat{g}_j + \hat{b}_j) \bmod 2, \tag{3}$$

where the notation $\hat{.}$ refers to the sound bits and images after embedding.
The extracted sound is then played synchronously with the GIF sequence. A voiced GIF player has been developed for this purpose and is described in section 5.

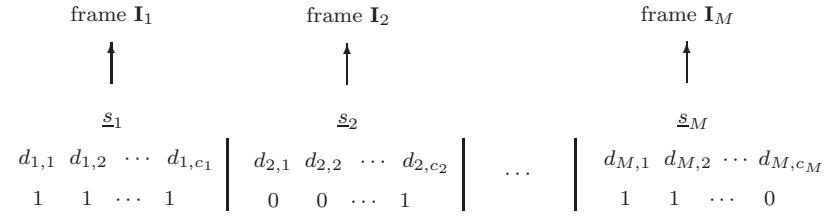


Fig. 3 Sound embedding: digital sound is divided into subvectors $\underline{s}_m$ and embedded in frames $\mathbf{I}_m$, where $m = 1 \cdots M$.

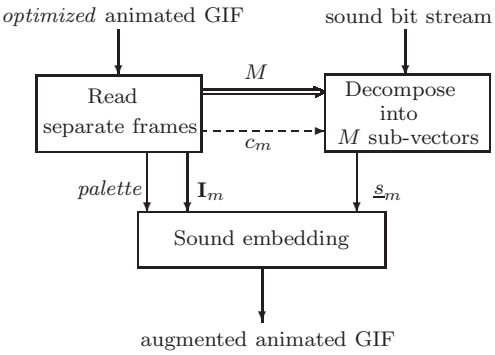


Fig. 4 Voiced GIF coder: sound embedding

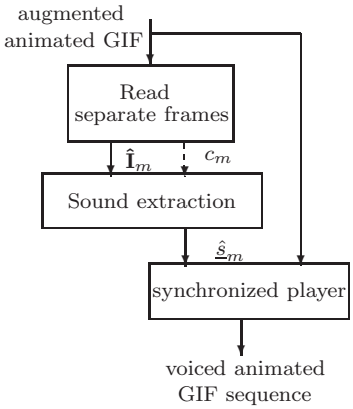


Fig. 5 Voiced GIF decoder: sound extraction

4 Experimental Results

4.1 Experimental Procedure and Test Material

We present results obtained with 4 different GIF animations: 2 cartoons excerpts from Disney's Snow White and Cinderella and advertisement cartoons and real scene excerpts produced by the company Telnet[9]. The sequences have been converted from video to GIF format using the Gifsicle tool [10]. The associated sound files have also been extracted from the video format and compressed by an MPEG 1 layer 3 high quality audio coder (LAME [10]) at a bit-rate of 96kbps to reduce their size for embedding into the GIF frames. The characteristics of the test GIF animations are displayed in Table 2, where column 4 gives the ratio in % of the inserted bits to the image definition and column 5 gives the ratio of modified pixels to image definition.

If the animated GIF sequence has not undergone transparency optimization, the embedding capacity c_m (in pixels/frame) is the same for all frames. In this case, the associated compressed sound file, with total bit size L, may be divided arbitrary into M equally long sub-vectors $\underline{s}_m$ containing each L/M bits, as shown in Table 2, and then embedded in the corresponding frames. As mentioned in section 3.3, it is possible to chose a particular spatial embedding pattern provided that it is known at the extraction step. Otherwise, if the sequence is transparency optimized, the maximal embedding capacity c_m varies across frames, depending on the number of transparent pixels of each frame. Hence, the size of binary sound vectors $\underline{s}_m$ varies across frames. It is difficult

[9] TELNET is a Tunisian group of companies specialized in innovation and high technology consulting (www.groupe-telnet.net).

[10] Lame encoder: Available from <http:// lame.sourceforge.net/ index.php>

in this case to embed the sound bits according to a particular pattern, because of the random spatial positions of the transparent pixels.

4.2 Experimental Results and discussion

The proposed voicing technique was tested with the test material of Table 2. The sound bits were embedded in the GIF animations (without transparency optimization) following a chess pattern. As mentioned in Table 2, L/M = 19200 bits were inserted in each frame, corresponding to the given ratio (in %) of inserted bits to the total available pixels. This ratio depends on the image definition of the tested GIF file. However, it is worth to note that the ratio of the actually modified pixels is always ca. the half of that of the inserted bits (last column in Table 2). This is confirmed by Fig. 6, where we plot the ratio of modified pixels to inserted bits across the frames of the Telnet advertisement cartoons and real scene. Indeed, Fridrich's algorithm modifies the pixel's color only if its parity bit is different from the embedded sound bit.

The quality of the augmented GIF frames were assessed by PSNR (Peak Signal to Noise Ratio) and by SSIM (Structural Similarity Index), which are the most popular full-reference metrics. The former evaluates the quality by assessing the intensity degradation and the latter is a measure of structural similarity between the reference and the processed image. SSIM is presented in [21] to be more representative of human visual perception. Fig. 8 displays the PSNR and SSIM obtained with the different GIF files and according to the in Table 2 specified embedding conditions: in both Disney Cartoons ca. 22% of the pixels were actually modified while in case of Telnet GIF files only $\sim$ 10% of the pixels were modified due to their higher definition. As expected, the results indicate that the quality of the augmented images depends on the size of embedded bits. Indeed, the PSNR of both Telnet GIF files reaches $\sim$ 46 dB, which indicates a very good quality, while the Disney cartoons show a lower PSNR at around 36 dB indicating a lower but still good quality. The SSIM metric (Fig. 8) is accordingly high when the embedding is at 10% and decreases when the embedding becomes more important (here $\sim$ 25%).
Fig. 9 illustrates an example of an augmented GIF frame using a chess pattern

GIF file	**GIF** (20 sec., $f_r = 5$ fps,M=100)			
	definition	c_m (pix./frame)	inserted bits/c_m	modified pix./c_m
2 Disney Cartoons	180×240	43200	44%	22%
Cartoons Telnet	288×360	103680	18.5%	9.3%
Real scene Telnet	266×360	95760	20%	10%
mp3 audio file	bit-rate	duration	L	L/M
	96kbps	20 sec.	1.92 10^6 bits	19200 bits

Table 2 Characteristics of test material. M: number of frames, f_r: frame rate; L: bit size of compressed audio, L/M: number of audio bits to be inserted in each GIF frame.

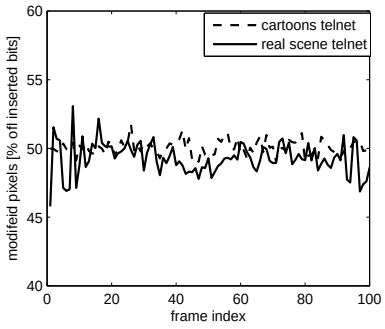


Fig. 6 Ratio in % of inserted bits to modified pixels.

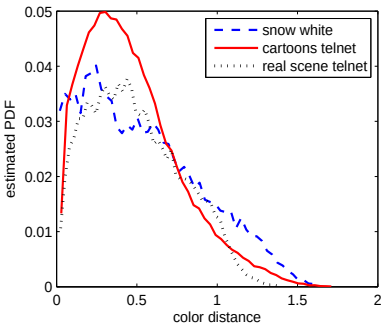


Fig. 7 Estimated PDFs of the palette color distances of the tested GIF files.

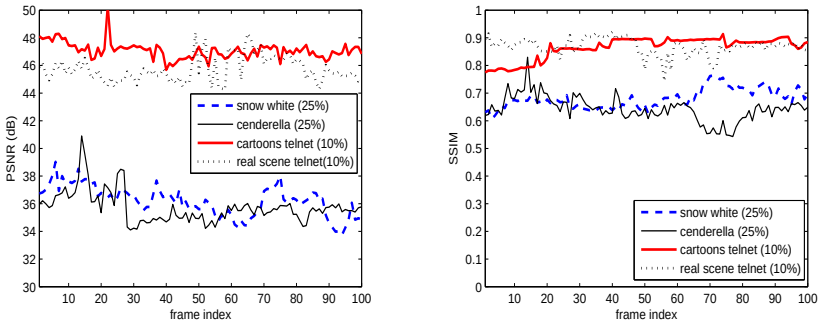


Fig. 8 PSNR and SSIM according to test material and test conditions of Table 2. The embedding pattern is a chess one.

(one pixel each ten pixels is modified), extracted from the real scene of the Telnet advertisement.

To explain the different quality of the augmented GIF with the same definition and embedding rate (case of the Telnet GIF files of Fig. 8), we analyse in Fig. 7 the estimated PDFs (Probability Density Function) of all the possible color distances (cf. equation (1)) of the color maps of the tested GIF animations. This figure gives information about the distribution of the color distances D in a given color map: the PDF of the cartoons Telnet indicates that the more the color distances are concentrated around small values the better the quality of the augmented frames is. This explains the higher PSNR obtained with the cartoons Telnet compared to the real scene Telnet where both have the same ratio of modified pixels (ca. 10% in Table 2).
Furthermore, the luminance-ordered palettes of cartoons Telnet and real scene Telnet, displayed respectively in Fig. 10 and Fig. 11, show that the real scene GIF contains more abrupt changes in colors due to the presence of detailed texture in the images, while the cartoons file has smoother color transitions.





Fig. 9 GIF frame without optimization extracted from the real scene of Telnet advertisement **(Copyright Telnet)**: original frame (left) and chess *voiced* frame with 10 % modified pixels.

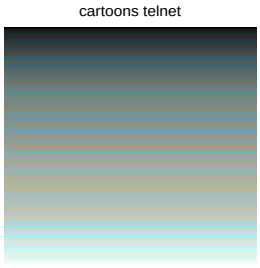


Fig. 10 Luminance ordered palette of the cartoons Telnet.

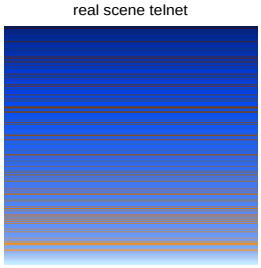


Fig. 11 Luminance ordered palette of the real scene Telnet.

Besides, informal subjective quality assessment have been conducted following double stimulus categorical rating [14], in which the observers judge the quality of a pair of images on a fixed five-point scale: excellent, good, fair, poor or bad. The original and the test images are displayed in random order one after the other for 3 seconds each. As this test is time consuming and because each test GIF file has 100 frames, frames were selected randomly (i.e. not all frames of each test file were presented to the subjects). The results correspond to the objective quality assessment, as the augmented GIF files with 10 % modified pixels obtained a majority of "excellent" scores, while the GIF files with 25 % modified pixels obtained 40 % "fair" and 60 "good". The tests are considered informal as the 6 participating observers were not really naive subjects as some of them are working in image processing and the test should be generalized with more GIF files at different embedding rates to be statistically significant.

4.3 Robustness and Optimization of the Coder

The first results obtained in this work are challenging to further improve the proposed GIF voicing method by using more sophisticated embedding strate-

gies in order to enhance the quality of the augmented GIF through minimization of the introduced distortion.

Fridrich's embedding method do not modify the initial color palette, and according to the above displayed results, it becomes obvious that, given a fixed amount of embedded bits, the quality of the augmented images depends on the original colors available in the palette and on the distribution of the colors distances. A color map with very different colors will result in high colors distances and thus in a more important image distortion. Hence, Fridrich's method is somehow limited by the content of the color palette.

Several embedding methods in palette based images are proposed in the literature, the most efficient of them attempt to modify iteratively the colors of the palette depending on the bits to be inserted [19][24][20], in order to minimize the image distortion. However, in case of animated GIF where many images are linked to a single palette, such optimization algorithms become very complex and do not necessarily converge to an optimal color map.

The more recent multi-bit high embedding capacity methods proposed in [22], [23] and [26] seem to provide high quality augmented images, however at a higher complexity cost, they are being evaluated for integration in the proposed application.

The GIF voicing application is not concerned with pirate attacks as it has no security issues. Robustness considerations are limited to the extraction robustness of the sound bits and should be addressed from common digital image/signal processing point of view. The robustness of the proposed technique to common image processing, as cropping, re-quantization, compression, etc. is the same as that of Fridrich's method, which is not robust to these common image manipulations. But it is possible to extract the inserted bits and to re-embed the sound in the GIF frames after manipulation by using the same embedding algorithm.

5 Voiced GIF Player

Our project started with a C program that allows creating voiced animated GIF and extracting the sound from the augmented GIF. A range of libraries were used like FFMPEG, ImageMagick and Gifsicle. The second phase was to develop a more user-friendly application in C++. This application is close to a video player, but for voiced animated GIF. The final phase consisted in developing an Android application, *GIFSound*, which is firstly a player of GIF animations containing sound, but also a function allowing to build such voiced GIF animations has been added.

The voiced visualization of the voiced GIF requires the synchronization of the playback of the extracted sound from each frame with the visualization of the frame itself. Our solution is based on synchronization using SMIL Timesheets from the Synchronized Multimedia Integration Language which allows to trigger the display of the frame at the same time as listening to the sound associated therewith. The advantage of this tool is the range of applications that it

can handle as it is used both for web applications with HTML5 and javascript as for desktop applications [1].

6 Conclusion

This paper presents an original and low complexity technique to add sounds to animated GIF sequences without increasing the electronic file size and without altering the wide support and portability of this format, although at the cost of low, most imperceptible distortion. The sound bits are embedded in the pixels of the GIF frames based on Fridrich's data hiding method. At reception, the inserted data bits are simply extracted from the parity bits of the RGB colors of the frames pixels. The quality of the augmented images depends mainly on the ratio "embedded sound information to image definition", but also on the color map of the GIF file. In this context, the sound is compressed by high quality perceptual coding (MPEG) and the tested conditions displayed that the quality at 20 embedding rate is good to excellent as measured by the PSNR and SSIM and according to informal subjective rating. Further improvements are being studied, including more sophisticated embedding strategies which better preserve image quality at the cost of a higher algorithmic complexity.

Acknowledgement

The authors would like to thank Rabaa Youssef, CEA-LinkLab, for her very useful advices on HTML5 as well as the Company Telnet Innovation Labs for financial and technology support of this project.

Appendix: Pseudo codes

This appendix gives basic pseudo codes of the voicing algorithm: the main code and the Function of Fridrich steganography method in the case where a bit is embedded in each pixel of the image. Note that only the voicing of a single GIF frame is described.

References

1. F. Cazenave, V. Quint, and C. Roisin. Timesheets.js: when SMIL meets HTML5 and CSS3. In *11th ACM symposium on Document engineering*, New York, 2011.
2. A. Cheddad, J. Condell, K. Curran, and P. Mc Kevitt. Digital image steganography: Survey and analysis of current methods. *Signal processing*, 90(3):727–752, 2010.

Main: VoicingGif

1: **Read:** pixels[H][W], map[256][3] // load original indexed GIF image in array *pixels* with H image height, W image width and corresponding color map in array *map*.
2: **Read:** sound[$H * W$]; // load sound bits in array *sound*
3: stegoPixels[H][W]; // array of indexed stego image
4: Set $k = 0$;
5: **for** $r = 1$ To H **do**
6: **for** $c = 1$ To W **do**
7: $k \leftarrow k + 1$;
8: stegoPixels[r, c] $\leftarrow$ pixels[r, c];
9: parity $\leftarrow$ (map[pixels[r, c],1] + map[pixels[r, c],2] + map[pixels[r, c],3])mod2;
10: **if** (parity <> sound[k]) **then**
11: stegoPixels[r, c] $\leftarrow$ StegoFridrich(map, sound[k], pixels[r, c]);//Call function *StegoFridrich* to search for the nearest color with required parity bit.
12: **end if**
13: **end for**
14: **end for**

Function: StegoFridrich(map, soundBit, currentPixel)
// implementation of equation (2): search for nearest color with required parity bit

1: Set maxDistance $=\sqrt{3 * 255 * 255}$; $j = 0$; newColor=255;
2: **while** ($j < 255$) and (j <> currentPixel) **do**
3: $P_j \leftarrow$ (map[j,1] + map[j,2] + map[j,3])mod2;
4: **if** (P_j = soundBit) and (dist(map, currentPixel, j) < maxDistance) **then**
5: newColor $\leftarrow j$;
6: maxDistance $\leftarrow$ dist(map, currentPixel, j);//function *dist(map,currentPixel,j)* computes the Euclidian distance between the colors indexed by j and *currentPixel* of the colormap *map*, according to (1))
7: **end if**
8: $j \leftarrow j + 1$;
9: **end while**
10: **return** newColor

3. I. J. Cox, M. L. Miller, and A. L. McKellips. Watermarking as communications with side information. *Proceedings of the IEEE*, 87(7):1127–1141, 1999.
4. S. Djaziri-Larbi, G. Mahé, I. Marrakchi, M. Turki, and M. Jaïdane. Doping and witness watermarking for audio processing. In *7th Int. Workshop on Systems, Signal Process. and their Applications*, Algeria, 2011.
5. J. Fridrich. A new steganographic method for palette-based images. In *IS&T PICS conference*, 1999.
6. J. Fridrich and D. Rui. Secure steganographic methods for palette images. In *3rd Int. Workshop on Information Hiding*. Springer Verlag, 2000.
7. Steve Johnson et al. *Adobe Illustrator CS5 on Demand*. Que Publishing, 2010.
8. S. Katzenbeisser and F. A. P. Petitcolas, editors. *Information Hiding Techniques for Steganography and Digital Watermarking*. Artech House, 2000.
9. S.-M. Kim, Z. Cheng, and K.-Y. Yoo. A new steganography scheme based on an index-color image. In *6th Int. Conf. on Information Technology*, Las Vegas, 2009.

10. E. Kohler. Gifsicle 1.7. Available from <http:// www.lcdf.org/ gifsicle>.
11. J. Krasner. *Motion graphics design and fine art animation: principles and practice*. Elsevier, 2004.
12. M. Kwan. Gifshuffle 2.0. Available from <http:// www.darkside.com.au/gifshuffle/>, 2010.
13. R. Machado. Ez stego. Available from <http:// www.stego.com>, 1997.
14. R. K. Mantiuk, A. Tomaszewska, and R. Mantiuk. Comparison of four subjective methods for image quality assessment. In *Computer Graphics Forum*. Wiley Online Library, 2012.
15. T. Morkel, J. HP Eloff, and M. S. Olivier. An overview of image steganography. In *Information Systems Security International Conference (ISSA)*, pages 1–11, 2005.
16. D. Mukherjee, J. J. Chae, and S. K. Mitra. A source and channel coding approach to data hiding with application to hiding speech in video. In *Proc. International Conference on Image Processing*, volume 1, pages 348–352. IEEE, 1998.
17. J. Niederst and J. Robbins. *Web design in a nutshell: A desktop quick reference*, volume 2. O'Reilly Media, Inc., 2001.
18. A. Sagi and D. Malah. Bandwith extension of telephone speech aided by data embedding. *Eurasip J. Appl. Signal Process.*, 2007.
19. C.H. Tzeng, Z.F. Yang, and W. H Tsai. Adaptive data hiding in palette images by color ordering and mapping with security protection. *IEEE Trans. on Communications*, 52(5), 2004.
20. X. Wang, T. Yao, and C.-T. Li. A palette-based image steganographic method using colour quantisation. In *IEEE Int. Conf. on Image Process.*, 2005.
21. Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Trans. on Image Proc.*, 13(4), 2004.
22. H. Wu and H. Wang. Multibit color-mapping steganography using depth-first search. In *International Symposium on Biometrics and Security Technologies*, pages 224–229. IEEE, 2013.
23. H. Wu, H. Wang, H.g Zhao, and X. Yu. Multi-layer assignment steganography using graph-theoretic approach. *Multimedia Tools and Applications*, pages 1–26, 2014.
24. M. Y. Wu, Y.K. Ho, and J. H. Lee. An iterative method of palette-based image steganography. *Pattern Recogn. Lett.*, 25(3), 2004.
25. D. Zhang, R. Zhang, X.n Niu, and Y. Yang. A digital watermarking algorithm for high capacity index image robust to format transformation. In *3rd International Conference on Computer Science and Information Technology*, volume 3, pages 216–220, 2010.
26. X. Zhang, S. Wang, and Z. Zhou. Multibit assignment steganography in palette images. *Signal Processing Letters*, 15:553–556, 2008.