Towards an Animated JPEG

Joël Theytaz, Lin Yuan, David McNally and Touradj Ebrahimi

Multimedia Signal Processing Group, EPFL, Lausanne, Switzerland

ABSTRACT

Recently, short animated image sequences have become very popular in social networks. Most animated images are represented in GIF format. In this paper we propose an animated JPEG format, called aJPEG, which allows the standard JPEG format to be extended in a backward compatible way in order to cope with animated images. After presenting the proposed format, we illustrate it using two prototype applications: the first in form of a GIF-to-aJPEG converter on a personal computer and the second in form of an aJPEG viewer on a smart phone. The paper also reports the performance evaluation of aJPEG when compared to GIF. Experimental results show that aJPEG outperforms animated GIF in both file size overhead and image quality.

Keywords: Animated JPEG, GIF, backward compatibility, APP marker

1. INTRODUCTION

Animated still image file formats such as animated Graphics Interchange Format (known as GIF) and motion JPEG provide the means to add a visually compelling motion component to still images. Compared to video files, working with still image file formats simplifies both server and client side operations and significantly reduces the computational complexity during both creation and playback. For historical reasons, animated GIF has established itself as the de-facto standard for this type of content. Given the state of the art in image compression and taking into account the trend towards multimedia formats that are royalty-free, seamless and managed by international standardization committees, GIF does not seem to be the best solution to provide the necessary features in an online world demanding ever more dynamic and captivating content.

Animated images have recently become very popular in social networks. Alex Chung, the founder and CEO of Giphy*, one of the biggest online databases for GIF files, states that: "Eventually all Web images will be animated in Harry Potter style".[1] He also says, "... the internet is sterile. It is emotionless, it lacks feeling, and its approach to content can make it feel dull." Adopting this view point, a large fraction of still images shared over the Internet will eventually be replaced by animated images. Indeed, animation empowers users to share and express emotions in a more powerful and individualistic way.

For the purpose of what follows we wish to distinguish between "video" and "animation". Video provides an audio-visual experience and typically serves to convey a complex and self contained message. In contrast to video, animation is characterized by Moreau[2] as: "a mini video, with no sound, that can be watched from start to finish in as little as one or two seconds in a simple, auto-looping fashion." Furthermore animations "offer a more convenient, faster and totally silent way to express something." An animation is viewed in the broader context of where it appears and is therefore "the perfect combination between image and video that really captures our attention."

Today the internet world of animated images is dominated by the GIF format. Other file formats such as Animated Portable Network Graphics (APNG)[3] or Multiple-image Network Graphics (MNG)[4] have been put forward. But none has attained GIF's popularity and widespread use despite two major shortcomings of GIF. On one hand, GIF supports a maximum of 256 colors (8 bit) per image frame. Given GIF's origins as an image file format used mainly for computer generated content, this restriction was of no particular concern. Yet, using GIF for photographic content reveals this limitation (see Figure 1) in the shape of severe color banding. On the

Further authors information: (send correspondence to Touradj Ebrahimi)
Joël Theytaz: E-mail: joel.theytaz@epfl.ch, Lin Yuan: E-mail: lin.yuan@epfl.ch
David McNally: E-mail: david.mcnally@epfl.ch, Touradj Ebrahimi: E-mail: touradj.ebrahimi@epfl.ch

*`http://giphy.com/`

Applications of Digital Image Processing XXXIX, edited by Andrew G. Tescher, Proc. of SPIE Vol. 9971, 99711X · © 2016 SPIE · CCC code: 0277-786X/16/$18 · doi: 10.1117/12.2240283

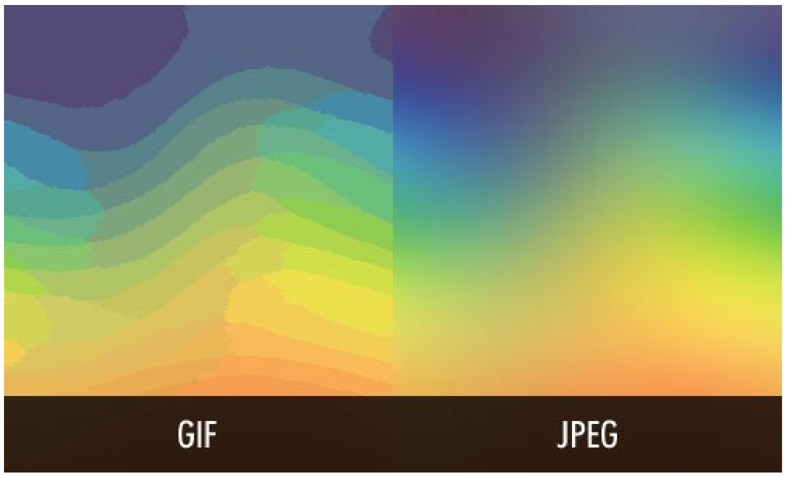


Figure 1: GIF vs. JPEG.

other hand, the underlying compression used to generate GIF images is lossless and hence sub-optimal for use in consumer photographic content. With animated image content now often generated on the basis of photographic images, JPEG appears to be the most suitable contender on which to base a new animated image file format.

In this paper we present **aJPEG**, an animated image file format based on JPEG file and image compression architecture which could serve as a better alternative to animated GIF. In the aJPEG architecture, we put particular emphasis on backwards compatibility with legacy JPEG decoders in order to allow an as seamless experience as possible to an as large as possible number of end-users. The performance of the proposed aJPEG format is also evaluated in comparison with animated GIF.

The paper is structured as follows. Section 2 presents related work, and cites and outlines the fundamental tools used in this work. Section 3 discusses the system design with particular emphasis on the proposed aJPEG format and architecture. Section 4 is split into two parts: the first describes an aJPEG transcoder, while the second describes two prototype applications of using such a format, a personal computer program and a mobile application, respectively. Section 5 reports the performance evaluation of aJPEG when compared to conventional animated GIF. Finally in Section 6 we draw conclusions from our work.

2. RELATED WORK

In the following, we briefly outline three animated image file formats. It is not comprehensive but puts particular emphasis on their respective suitability for social media and emerging online applications.

2.1 GIF

Today, animated GIF is the dominant animated image file format. It derives its name from Graphics Interchange Format and was first introduced by CompuServe in 1987 under the name 87a. GIF was positioned as an alternative to formats such as PiCture eXchange (PCX) and *MacPaint* offering support for color and smaller file sizes. During 1989 CompuServe released version 89a of its file format which was to become known as GIF and added support for animation, transparent background colors, storage of metadata and text overlays. From the outset, GIF was subject to license and royalty constraints due to its use of Lempel-Ziv-Welch (LZW) compression,[5,6] a technology under patent by Unisys corporation until 2004.[7] While initially released as an image file format for CompuServe customers, GIF became a de-facto standard on the early World Wide Web when support for the file format was added to the Netscape 2.0 navigator in 1995.

GIF image encoding supports an input color space of 8 bits per primary color. The encoder then projects colors in the image to be encoded to a maximum of 256 colors and tabulates these in a palette which is addressed through an 8-bit index. This restricted number of colors was quite sufficient in the 1980s but severely limits the visual quality of GIF images for photographic images and content generated using modern computer graphics (see Figure 1).

With the introduction of animated GIF, CompuServe added the concept of pixel transparency. In secondary images, pixels can be flagged to be transparent. The decoder will then substitute these transparent pixels with the corresponding pixels from the primary image. This highly desirable mechanism improves compression efficiency and in the case of computer generated content, supports more efficient and richer content creation procedures.

GIF image compression is particularly suitable for sharp edges in graphical material with no noise. As such, it supports efficient encapsulation of simple graphics, logos and small, animated cartoon sequences. Today the use of GIF extends to sprites and avatars in online games and mobile apps. With social network platforms such as Facebook, adding support for GIF, users are widely adopting the opportunity to post animated images, giving GIF a new lease on life.

The bitstream syntax used in GIF file format is shown in Figure 2 and a description of GIF format can be found on this website.[8] The full specifications for the GIF file format is available here[†].

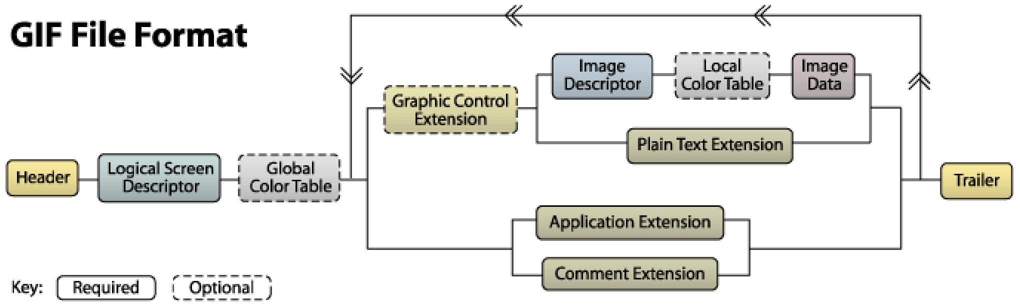


Figure 2: GIF file format.[8]

2.2 Motion JPEG and Motion JPEG 2000

Motion JPEG (MJPEG) addresses the sub-optimal compression in GIF by replacing LZW with the efficient image compression embodied in the JPEG standard.[9] The basic architecture of an MJPEG file follows that of a GIF file: Each image constituting an MJPEG sequence is individually compressed using JPEG. These images are then combined into a single file which additionally includes display, context and metadata. As such, MJPEG is an intraframe coding scheme and no temporal redundancy in the image sequence is exploited to achieve higher compression. The MJPEG file for a given image sequence is therefore larger than the equivalent sequence obtained from state-of-the-art video compression scheme such as H.265/HEVC. But this is offset by much reduced complexity, both during encoding and decoding.

Many variants and implementations for MJPEG have been put forward. Yet, no unifying standard or specification was ever adopted and published by the JPEG standardisation committee for an MJPEG file format. As a consequence, MJPEG has not achieved widespread acceptance due to incompatibility and interoperability issues which exhist between implementations put forth by different developers and system vendors.

Motion JPEG 2000 follows the same implementation strategy as animated GIF and MJPEG, building on a simple file structure and an intraframe coding strategy for each image in the sequence. Avoiding the fragmentation that blocked the widespread adoption of MJPEG, the JPEG 2000 standard was adopted and published by JPEG standardisation committee in a Part 3 extension of JPEG 2000 standard specifying support for motion and animation[‡].

Yet JPEG 2000 and, as a consequence, Motion JPEG 2000 are victims of the success of the legacy JPEG standard which, today, is still the dominant image coding standard employed in all consumer and prosumer products. This underlines the requirement for backward compatibility to JPEG of a motion image format that could serve to displace GIF in Internet and social media applications.

[†]`https://www.w3.org/Graphics/GIF/spec-gif89a.txt`
[‡]`https://jpeg.org/jpeg2000/`

3. AJPEG SYNTAX AND STRUCTURE

This section discusses in details the syntax and structure of the proposed aJPEG file format. Before discussing the aJPEG format, we briefly introduce the JPEG Application Segments, based on which an animated JPEG solution is achieved.

3.1 JPEG Application Segments

A JPEG image consists of a sequence of segments, each beginning with a marker. Each marker begins with a `0xFF` byte followed by a byte indicating what kind of marker it is. Among all these segments, there are several application-specific segments (called Application Segments) which are used to signal specific JPEG formats or to store certain metadata information about an image. Different Application Segments are marked with different identifiers, called APP markers. For instance, JFIF (JPEG File Interchange Format)[10] and Exif (Exchangeable image file format)[11] are the two most popular JPEG formats using two different Application Segments respectively. JFIF is the most popular format for storage and transmission of images on the World Wide Web, while Exif is a popular way for digital cameras and other photographic capture devices to tag capture and location related metadata about photos. The two formats are often not distinguished from each other and are simply referred to as JPEG each with their own APP marker (APP0 for JFIF, APP1 for Exif) in the header of a JPEG file. Recently, a new standard called JPEG XT[12] addressed the needs of photographers for higher dynamic range (HDR) images[13–15] in both lossy and lossless coding[16, 17] while retaining backward compatibility with legacy JPEG decoders. The central idea underlying the backward compatible coding of HDR content is to encode a low dynamic range version of the HDR image generated by a tone-mapping operator using a conventional JPEG encoder and to insert the extra encoded information for HDR in an application marker. In addition, Yuan and Ebrahimi[18] proposed a JPEG Transmorphing algorithm which applies JPEG Application Segments for hiding visual information inside the original image. Adopting a similar idea, any useful information can be embedded in the application segments of a JPEG file.

3.2 aJPEG Overview

The central idea behind the proposed Animated JPEG is to encode a default frame of an animation sequence as standard JPEG while storing the information about the other frames in the Application Segments of the JPEG file header. Therefore, any legacy JPEG decoder will be able to decode the default frame. In order to decode the other frames and display the animation, a dedicated aJPEG decoder or transcoder would be necessary. The bitstream syntax of the proposed aJPEG format is shown in Figure 3. It starts with a standard JPEG header, consisting of various APP markers for different purposes. We took a unique APP marker for the use of aJPEG. In our current implementation, APP11 is used. The metadata and image data of the animated image frames, except for the default frame, is inserted in a sequence of APP11 markers. The inserted data consists of the necessary information to recover the animated image frames, including the number of inserted frames, frame rate, and the image data of those frames. Finally, the image data block (in blue) holds the image data of the default frame, which can be read by any legacy JPEG decoder. Details of the format syntax and structure are described in the following subsections.

3.3 aJPEG Header

The metadata of the inserted frames starts as an aJPEG header, the structure of which is shown in Figure 4. The aJPEG header uses an APP11 segment to signal the metadata of the inserted animation frames of the aJPEG file. It starts with a header tag "AJPEG" identifying the aJPEG format. This tag occupies five bytes in the segment. Then, the next byte is used to signal the mode of an aJPEG format, which indicates the method for encoding the inserted image frames. In our current implementation, the inserted frames are encoded in JPEG compression with the same quality factor as the default frame. In practice, any other encoding methods can be used to compress the inserted frames. At the end of the aJPEG header, two more bytes are used to indicate the number of inserted image frames, noted as M.

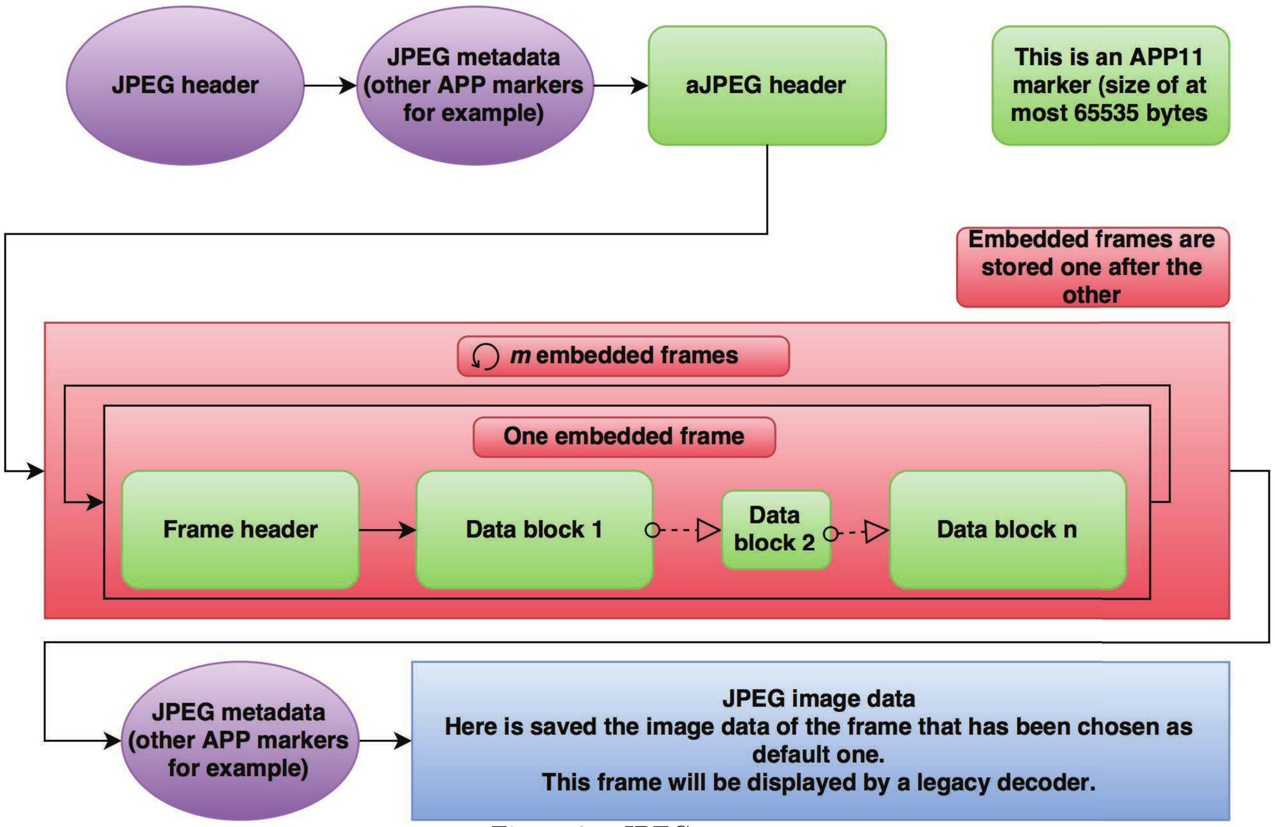

JPEG header
JPEG metadata (other APP markers for example)
aJPEG header
This is an APP11 marker (size of at most 65535 bytes
Embedded frames are stored one after the other
m embedded frames
One embedded frame
Frame header
Data block 1
Data block 2
Data block n
JPEG metadata (other APP markers for example)
JPEG image data
Here is saved the image data of the frame that has been chosen as default one.
This frame will be displayed by a legacy decoder.

Figure 3: aJPEG syntax.

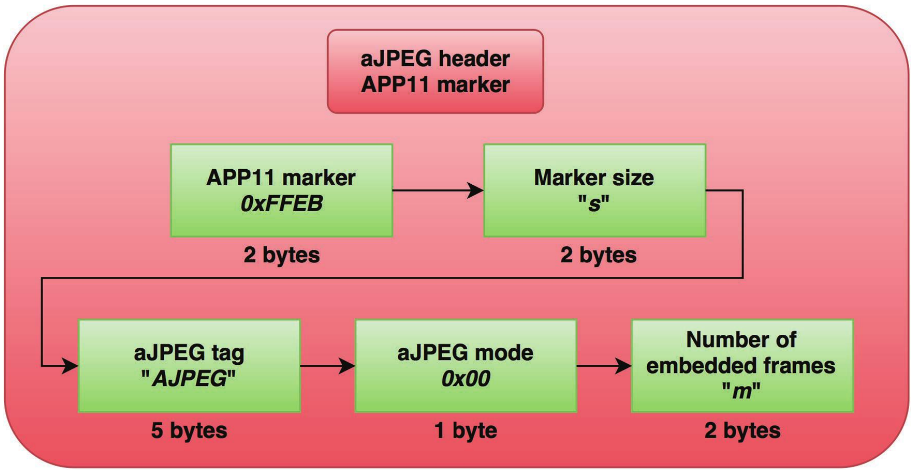


Figure 4: aJPEG header.

3.4 Inserted Frames

After the aJPEG Header, a sequence of segments are used to store the image data of animated frames one after another. Each image frame starts with a frame header, the structure of which is illustrated in Figure 5. Similar to any other APP marker, this marker starts with a marker ID (#11) and marker size. The rest of this marker signals the following metadata of the particular frame:

- **Image size** s**:** file size of the frame in bytes.

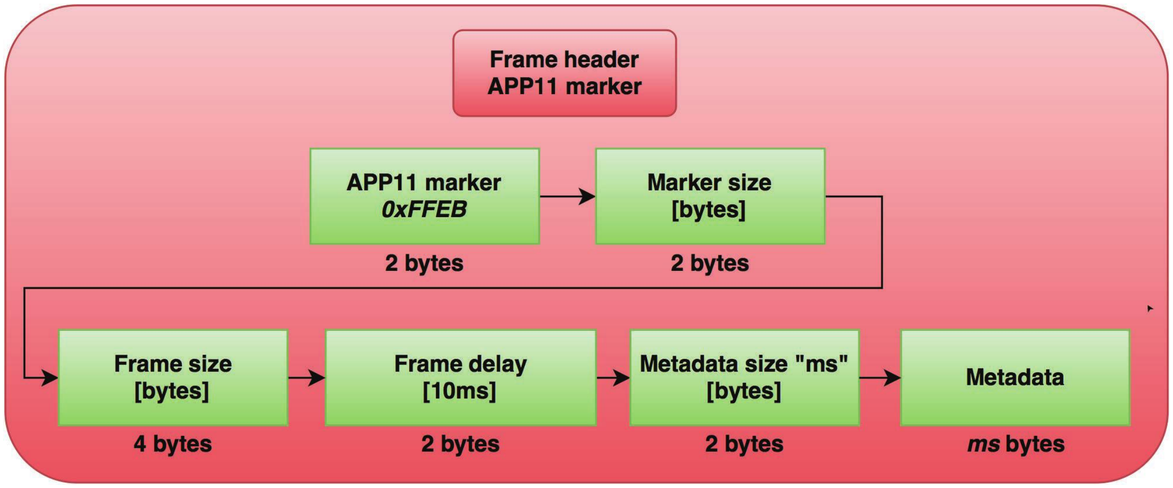


Figure 5: Frame header.

- **Frame delay:** the length of time (in hundredths of a second) for which the frame is displayed during animation.
- **Extra metadata:** extra information about the frame, e.g. annotation and geo-tag.

After the frame header, the bitstream of each image frame $i, i \in [1, M]$ is inserted in a sequence of segments byte by byte. Since JPEG does not allow for an APP segment larger than 65,535 bytes[§], the bitstream of each image frame i may need to be separately stored in n_i APP markers:

$$n_i = \lceil \frac{s_i}{65533} \rceil, \tag{1}$$

where s_i is the size of the frame i in bytes and $\lceil . \rceil$ is the ceiling function.

3.5 Compressed Image Data

The image data of the default frame is compressed in the same way as the standard JPEG compression, consisting of color space transformation, downsampling, discrete cosine transform (DCT), quantization, and entropy coding. Since the aJPEG file still contains the standard JPEG headers, a legacy JPEG decoder can read the compressed default frame as a standard JPEG file. With a dedicated aJPEG decoder or transcoder, the other image frames can be read from the aJPEG APP markers and played back as an animated sequence. Therefore, an aJPEG file is backward compatible with legacy JPEG standard, with the advanced functionality of being capable of rendering animated content.

4. IMPLEMENTATION AND PROTOTYPE APPLICATIONS

We have implemented a dedicated aJPEG transcoder for encoding and decoding of the proposed animated JPEG image format, based on an open source JPEG library version 6b maintained by the Independent JPEG Group (IJG) [¶]. To demonstrate the use of aJPEG, we created two prototype applications for GIF-to-aJPEG conversion and aJPEG playback, on a personal computer and a smart phone, respectively.

[§]Each APP marker signals its "marker size" using two bytes (16 bits), resulting in a maximal of 65,535 ($2^{16} - 1$) bytes as the maximal length of a segment. Taking into account the two bytes in the marker for recording "marker size", a total of 65,533 bytes for the rest of the marker can be used.

[¶]`http://www.ijg.org/`

4.1 aJPEG Transcoder

The aJPEG transcoder is implemented as a command line executable called `ajpegtran`:

- `build`: encoding a sequence of JPEG images to an aJPEG file.
- `extract`: decoding image frames in JPEG from an aJPEG file.

The `build` (encoding) process takes four basic parameters, and can be executed using the following command:

```
$ ./ajpegtran -build [defaultFrame] [outputFile] [inputFolder] [delay]
```

where `[defaultFrame]`, `[outputFile]`, `[inputFolder]` specify the path of the default frame, the output aJPEG file and the directory containing all inserted frames respectively. `[delay]` indicates the animation delay in hundredths of a second.

The `extract` function takes only the path of aJPEG file as input and extract all its image frames into a new directory. The following command line is used for the `extract` (decoding) function:

```
$ ./ajpegtran -extract [inputFile]
```

where `[inputFile]` indicates the path of the input aJPEG file. All the exacted frames are generated in a new directory `images/` under the same path as the input aJPEG file.

4.2 Prototype 1: Personal computer aJPEG Converter/Player

To illustrate the use of the proposed aJPEG format, we have developed a personal computer program in Java that can convert a GIF file to an aJPEG file or play the animation from an aJPEG image. Figure 6(a) shows the graphical user interface of the program.

In this program, one can load an input file, in either GIF or aJPEG format. When an animated GIF image is loaded (GIF-to-aJPEG Converter mode), the program converts it into an aJPEG image and plays the resulting animation. The whole process includes the following operations:

1. Extract and convert each frame of GIF to JPEG, using a user defined quality factor (75 by default).
2. Encode all image frames into an aJPEG file, using the first frame of GIF as the default frame to be displayed by legacy JPEG viewer. Frame delay of the generated aJPEG is the same as the original GIF.
3. Play the image frames of generated aJPEG as an animation. A speed factor can be set by user to control the speed (frame rate) of the playback.

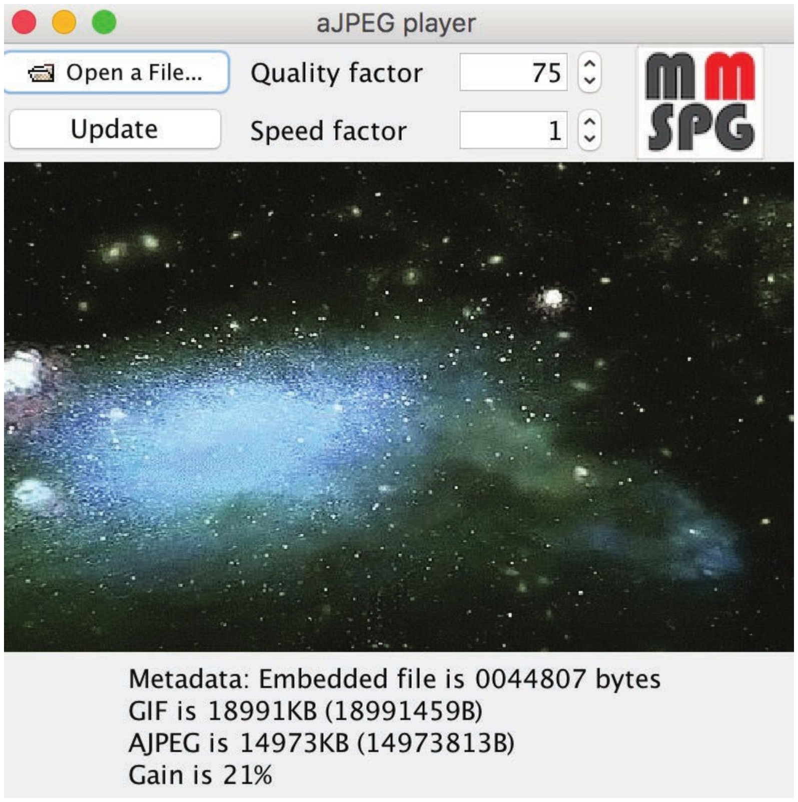
When an aJPEG image is loaded, the program simply reads each frame and plays the animation. In this case, the program acts as a simple aJPEG viewer or player. Furthermore, in the GIF-to-aJPEG Converter mode, the program displays, at the bottom of the program interface, the comparison between the input GIF and generated aJPEG with respect to their file sizes.

4.3 Prototype 2: Smart phone aJPEG Converter/Player

We have also developed an Android-based aJPEG Converter/Player. Similar to the personal computer program in Section 4.2, the application can convert an animated GIF to aJPEG, or play animation from an aJPEG input (either locally stored or remotely accessed from a URL) on the smart phone. Due to restrictions in smart phones memory, the GIF-to-aJPEG conversion cannot cope with very large GIF files in the implementation. The screenshot of the application is shown in Figure 6(b).

5. PERFORMANCE EVALUATION

This section reports the performance evaluation of the proposed aJPEG format. We conducted a series of experiments to assess compression efficiency (file size) and objective quality of aJPEG in comparison to GIF.



Quality factor 75

Speed factor 1

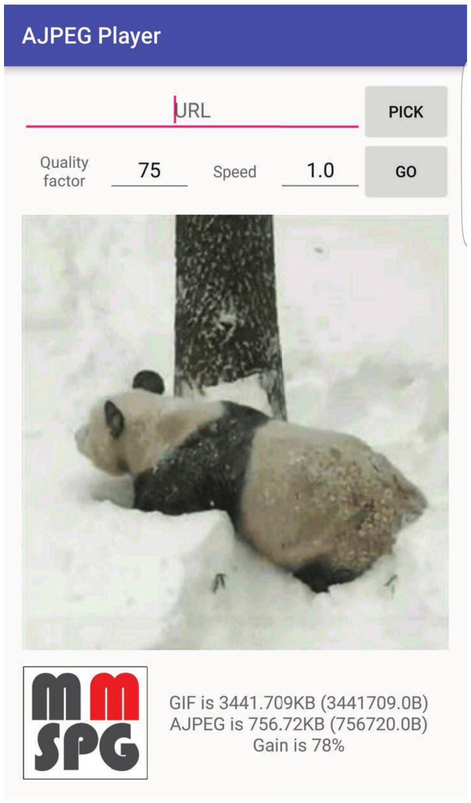
Metadata: Embedded file is 0044807 bytes

GIF is 18991KB (18991459B)

AJPEG is 14973KB (14973813B)

Gain is 21%

(a) Personal computer aJPEG Converter/Player



Quality factor 75

Speed 1.0

GIF is 3441.709KB (3441709.0B)

AJPEG is 756.72KB (756720.0B)

Gain is 78%

(b) Smart phone aJPEG Converter/Player

Figure 6: Screenshots of two prototypes for aJPEG.

5.1 Datasets

To conduct the experiments, two datasets were used, with detailed information listed below:

Tumblr GIF Description Dataset (TGIF) [19] The TGIF dataset‖ contains 100K animated GIFs and 120K sentences describing their visual content. We took six GIFs from the whole dataset, which cover different types of image content. The six GIFs are noted as *Man*, *Flower*, *Snow*, *Car*, *Bump* and *Candy* respectively. An example frame of each GIF is shown in Figure 7.

EPFL-PoliMI Video Quality Assessment Database [20,21] The EPFL-PoliMI dataset contains 156 video streams for video quality assessment. From the dataset, we took six video sequences which were also used in subjective evaluations in.[21] The six sequences are all in I420 raw progressive format, with 10 seconds long at 4CIF spatial resolution (704×576 pixels). They are referred to as *Crowdrun*, *Duckstakeoff*, *Harbour*, *Ice*, *Parkjoy* and *Soccer* respectively. Example images of the 6 video sequences are shown in Figure 8.

5.2 Overhead Evaluation

In the first experiment, we evaluated the overhead of aJPEG images compared to GIFs, in terms of their file sizes. Firstly, we converted each of the six GIFs from TGIF dataset into a set of aJPEG files compressed with different quality factors $Q \in [50, 90]$. In each aJPEG file, the default frame and inserted frames were compressed with the same quality factor. For each content, we computed the file sizes of both GIF file and aJPEG files. The comparison results for the six different contents are shown in Figure 9. From the results, one can observe that

‖`http://raingo.github.io/TGIF-Release/`

(a) Man

(b) Flower

(c) Snow

(d) Car

(e) Bump

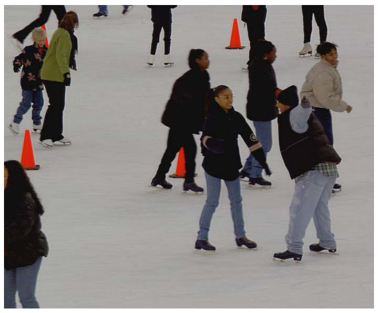
(f) Candy

Figure 7: Six image content from TGIF dataset.

(a) Crowdrun

(b) Duckstakeoff

(c) Harbour


(d) Ice

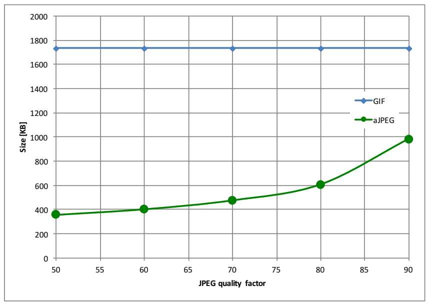
(e) Parkjoy

(f) Soccer

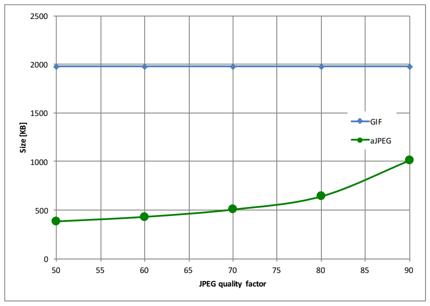
Figure 8: Six video content from EPFL-PoliMI dataset.

the file sizes of aJPEG images compressed with different quality factors are always smaller than that of GIF, for every image content.

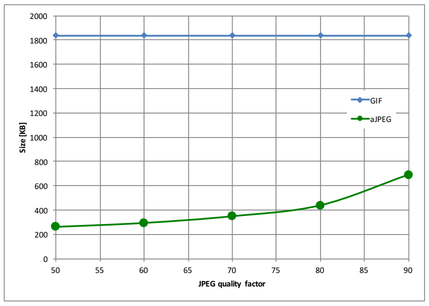
To further confirm our observation, we conducted a similar experiment using the other six video sequences from EPFL-PoliMI dataset. This time, for each sequence, we converted the first 100 raw image frames to an animated GIF file and five aJPEG files compressed with different quality factors ($[50, 90]$). The comparisons in file size between GIF and aJPEG are shown in Figure 10. The same observation was found: for every video



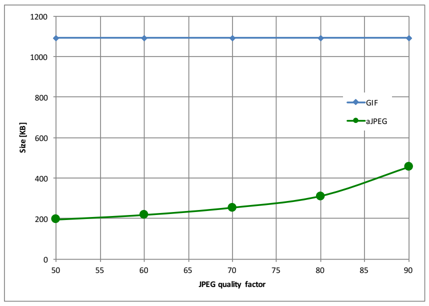
(a) Man



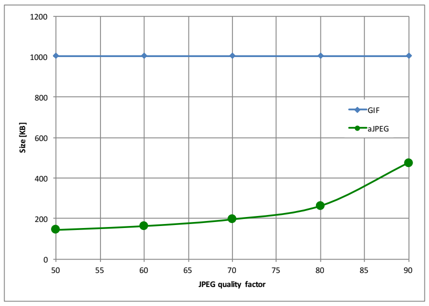
(b) Flower



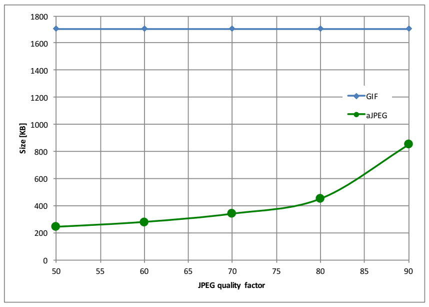
(c) Snow



(d) Car



(e) Bump



(f) Candy

Figure 9: GIF vs. aJPEG with respect to file size, for TGIF dataset.

sequence, the file size of its aJPEG version was always smaller than that of GIF. Then, we computed the ratio in file size between GIF and the aJPEG compressed at quality factor of 80, listed in Table 1. One can observe that the aJPEG files at quality factor 80 are at least 2.8 times smaller than GIF. Considering the fact that JPEG compression with a quality factor of 75 usually provides good-enough visual quality, aJPEG shows a significant advantage over GIF in terms of compression efficiency.

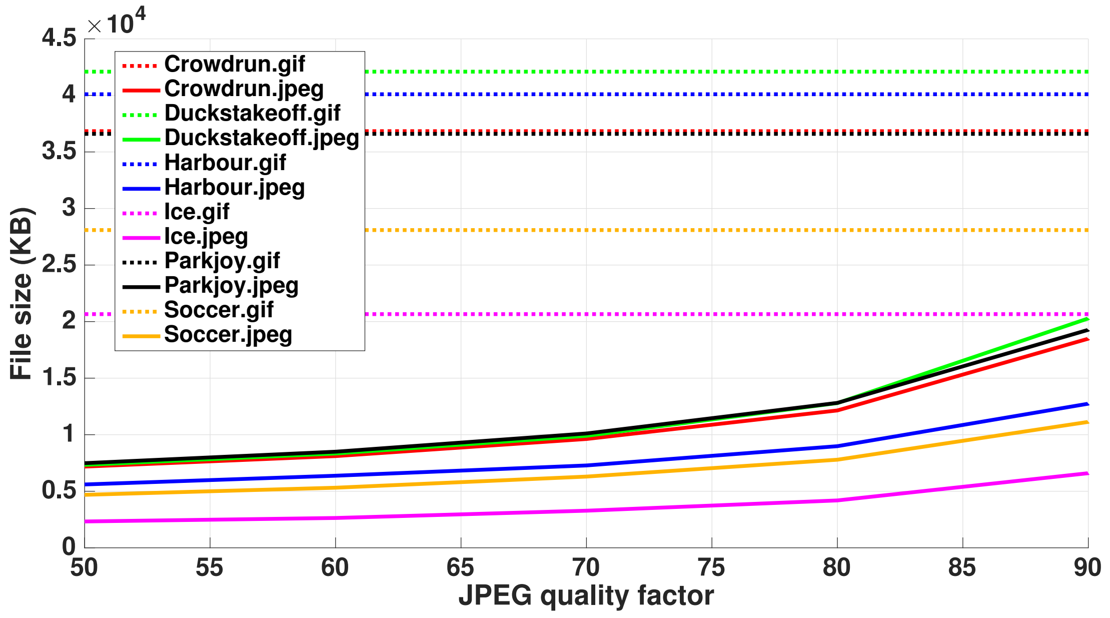


Figure 10: GIF vs. aJPEG with respect to file size, for EPFL-PoliMI dataset.

Table 1: Ratio of file sizes between GIF and aJPEG (quality factor of 80) for different content.

	Man	Flower	Snow	Car	Bump	Candy	Crowd.	Duck.	Harbour	Ice	Park.	Soccer
Ratio	2.84	3.08	4.19	3.5	3.84	3.79	3.03	3.29	4.46	4.93	2.85	3.6

5.3 Quality Evaluation

To further evaluate the performance of aJPEG with regards to its quality, we conducted another set of experiments to compare aJPEG and GIF, using two objective metrics: peak signal-to-noise ratio (PSNR) and structural similarity (SSIM).[22] In this experiment, we used the six video sequences from EPFL-PoliMI dataset, and generated from raw video the GIF and aJPEG files in the same way as in Section 5.2. Then, we computed the PSNR and SSIM of GIF and JPEG frames against the reference raw video frames. For each content, only the first 100 frames were considered. To do so, we converted the GIF and aJPEG frames to YUV color space and calculated PSNR and SSIM on their Y channels. The comparison results for PSNR and SSIM are shown in Figure 11 and Figure 12 respectively.

For every content, image frames from aJPEG compressed with the quality factor of 90 always show better quality than GIF in both metrics. aJPEG image frames compressed with quality factor 80 also have better or equivalent quality for most content, except for *Duckstakeoff* and *Parkjoy*, where the difference (between GIF and aJPEG) in both quality measures is very small. In addition, for each content, comparison results using PSNR and SSIM are almost consistent, although PSNR has proven to be inconsistent with human visual perception.[23]

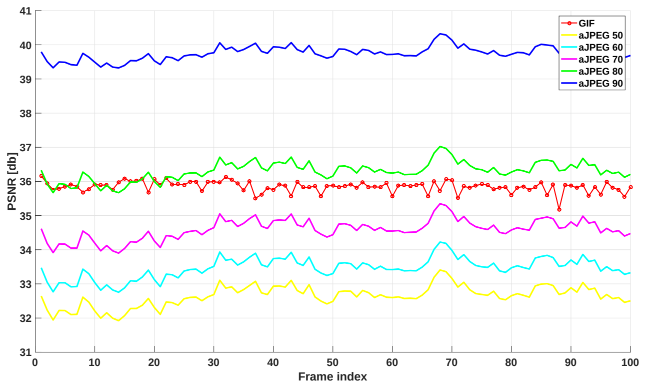
Combining the experimental results on both compression efficiency and objective quality assessment, one can conclude that aJPEG images compressed with a quality factor of 80 or higher usually outperform animated GIFs.

6. CONCLUSION

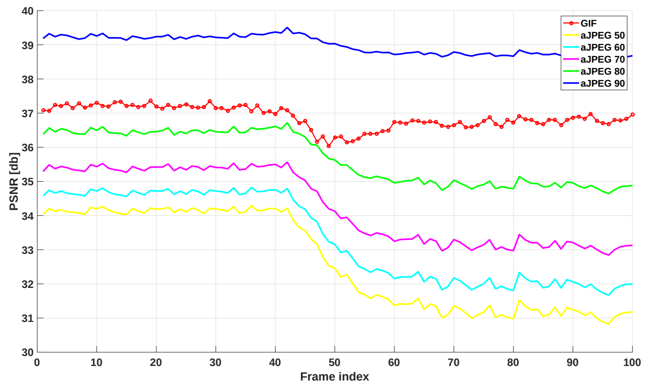
This paper presents aJPEG, an animated image file format based on legacy JPEG file format and image compression architecture which could serve as a better alternative to animated GIF. The fundamental idea of aJPEG is to code a default frame as standard JPEG while preserving the information about the other frames of the animation in JPEG APP markers. Therefore, any legacy JPEG decoder is able to decode the default frame. In order to decode the other frames and display the animation, a dedicated aJPEG decoder or transcoder is needed. The evaluation experiments were conducted to inspect the performance of the proposed aJPEG format in comparison with animated GIF. Experiments showed that aJPEG image compressed with JPEG quality factor higher than 80 usually outperforms conventional animated GIF in both file size and image quality.

ACKNOWLEDGMENTS

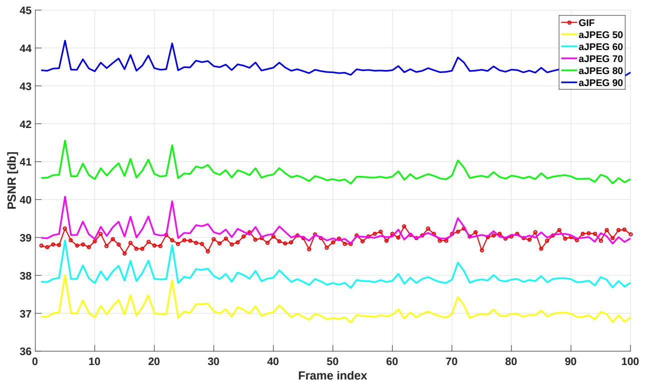
The research behind this publication was conducted in the framework of Swiss National Foundation for Scientific Research project LEADME (200020 – 149259) and H2020 Innovation Action project immersiaTV, funded by Swiss Secretariat for Education, Research and Innovation - SERI.



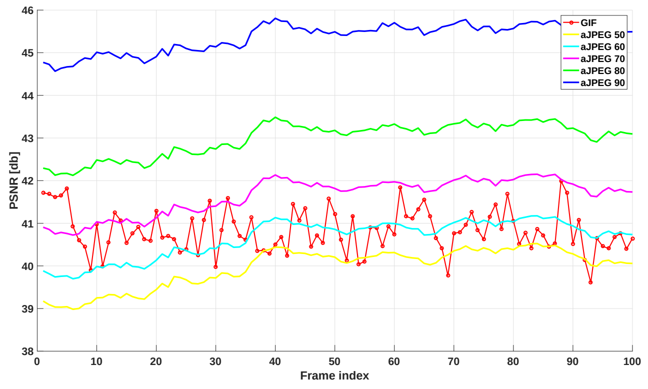
(a) Crowdrun



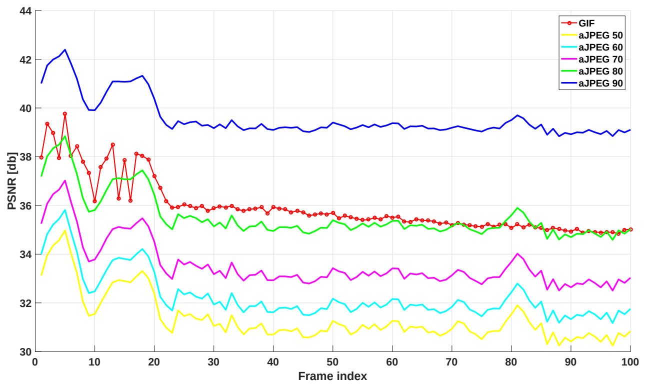
(b) Duckstakeoff



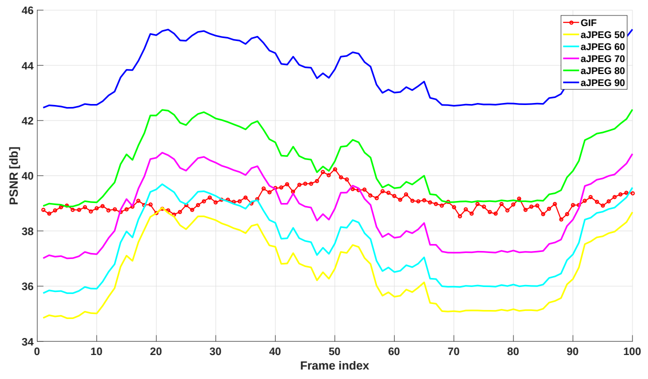
(c) Harbour



(d) Ice

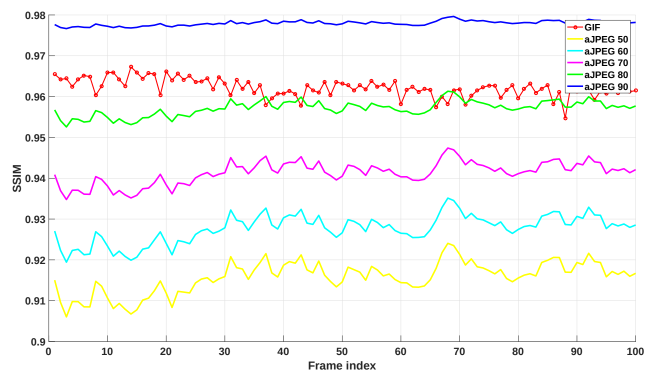


(e) Parkjoy

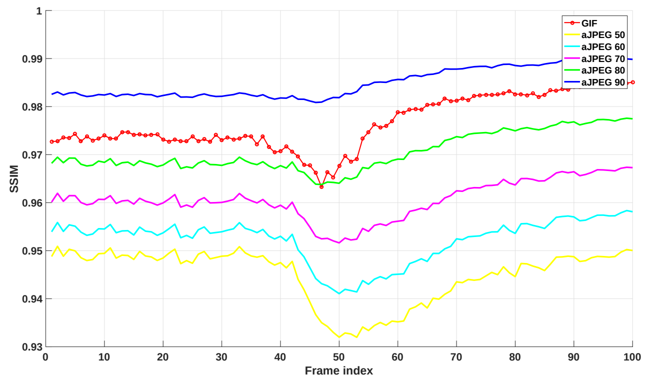


(f) Soccer

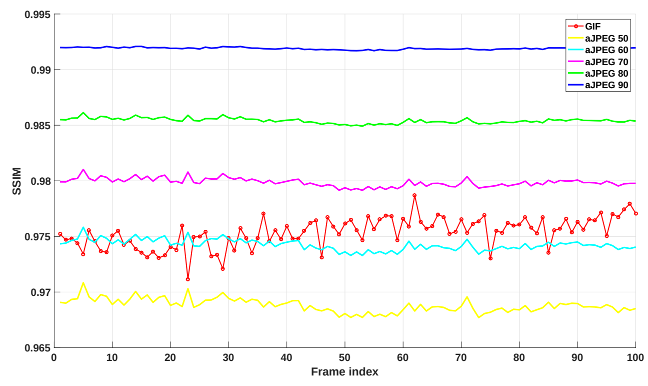
Figure 11: Quality comparison between GIF and aJPEG with respect to PSNR.



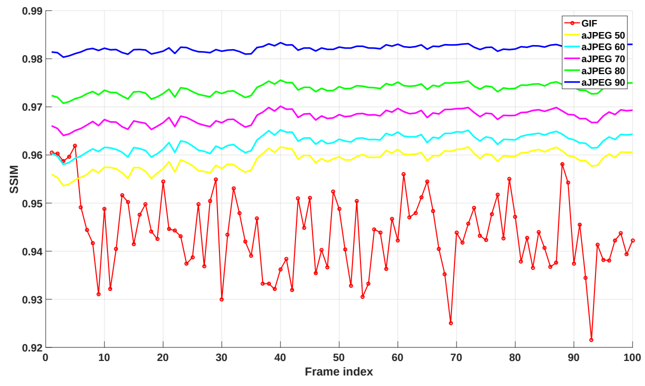
(a) Crowdrun



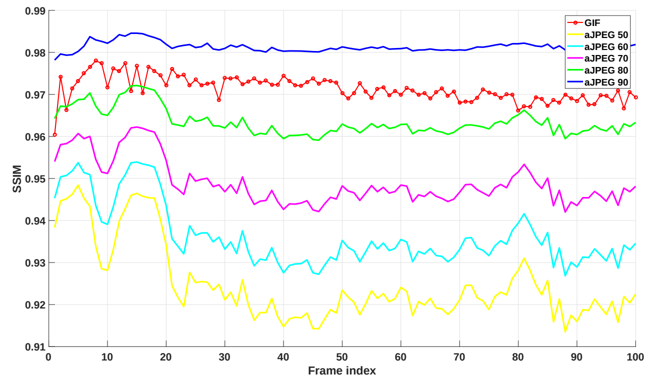
(b) Duckstakeoff



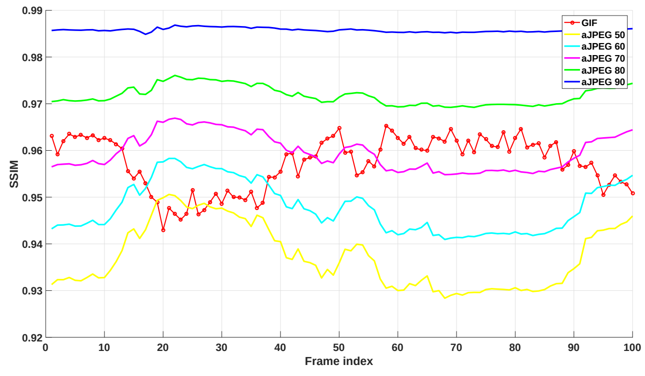
(c) Harbour



(d) Ice



(e) Parkjoy



(f) Soccer

Figure 12: Quality comparison between GIF and aJPEG with respect to SSIM.

REFERENCES

[1] Brown, M., "Giphy CEO Says Eventually All Web Images Will Be Animated in 'Harry Potter' Style." `https://www.inverse.com/article/14908-giphy-ceo-says-eventually-all-web-images-will-be-animated-in-harry-potter-style` (2016). [Online; accessed Mai-2016].

[2] Moreau, E., "The Rise of the Animated GIF." `http://webtrends.about.com/od/Gifs/a/Animated-Gif.htm` (2015). [Online; accessed Mai-2016].

[3] Parmenter, S., Vukicevic, V., and Smith, A., "APNG Specification." `https://wiki.mozilla.org/APNG_Specification` (2015). [Online; accessed Mai-2016].

[4] Roelofs, G., "Multiple-image Network Graphics." `http://www.libpng.org/pub/mng/` (2015). [Online; accessed Mai-2016].

[5] Ziv, J. and Lempel, A., "Compression of Individual Sequences via Variable-rate Coding," *IEEE Trans. Inf. Theor.* **24**, 530–536 (Sept. 2006).

[6] Welch, T. A., "A technique for high-performance data compression," *Computer* **17**, 8–19 (June 1984).

[7] Welch, T., "High speed data compression and decompression apparatus and method," (Dec. 10 1985). US Patent 4,558,302.

[8] Flickinger, M., "What's In A GIF - Bit by Byte." `http://www.matthewflickinger.com/lab/whatsinagif/bits_and_bytes.asp` (2016). [Online; accessed Mai-2016].

[9] Wallace, G. K., "The JPEG still picture compression standard," *Communications of the ACM* , 30–44 (1991).

[10] Brower, B., Clark, R., Hinds, A. T., Lee, D. T., and Sullivan, G. J., "Information technology - Digital compression and coding of continuous-tone still images: JPEG File Interchange Format (JFIF)," (2011). available on www.jpeg.org as document WG1N5642, published by ISO as 10918-5.

[11] "Exchangeable image file format for digital still cameras: Exif Version 2.2," (2002). Standard of Japan Electronics and Information Technology Industries Association.

[12] Richter, T., "On the standardization of the JPEG XT image compression," in [*Picture Coding Symposium (PCS), 2013*], 37–40 (Dec 2013).

[13] Ward, G. and Simmons, M., "JPEG-HDR: A backwards-compatible, high dynamic range extension to JPEG," in [*ACM SIGGRAPH 2006 Courses*], *SIGGRAPH '06*, ACM, New York, NY, USA (2006).

[14] Korshunov, P. and Ebrahimi, T., "A JPEG backward-compatible HDR image compression," in [*Proc. SPIE*], **8499**, 84990J–84990J–12 (2012).

[15] Richter, T., "Backwards compatible coding of high dynamic range images with JPEG," in [*Data Compression Conference (DCC), 2013*], 153–160 (March 2013).

[16] Richter, T., "On the integer coding profile of JPEG XT," in [*Proc. SPIE*], **9217**, 921719–921719–19 (2014).

[17] Pinheiro, A. G., Fliegel, K., Korshunov, P., Krasula, L., Bernardo, M. V., Pereira, M., and Ebrahimi, T., "Performance evaluation of the emerging JPEG XT image compression standard," in [*IEEE 16th International Workshop on Multimedia Signal Processing, MMSP 2014*], 1–6 (September 2014).

[18] Yuan, L. and Ebrahimi, T., "Image transmorphing with JPEG," in [*Image Processing (ICIP), 2015 IEEE International Conference on*], 3956–3960 (Sept 2015).

[19] Li, Y., Song, Y., Cao, L., Tetreault, J. R., Goldberg, L., Jaimes, A., and Luo, J., "TGIF: A New Dataset and Benchmark on Animated GIF Description," *CoRR* **abs/1604.02748** (2016).

[20] Simone, F. D., Tagliasacchi, M., Naccari, M., Tubaro, S., and Ebrahimi, T., "A H.264/AVC video database for the evaluation of quality metrics," in [*2010 IEEE International Conference on Acoustics, Speech and Signal Processing*], 2430–2433 (March 2010).

[21] Simone, F. D., Naccari, M., Tagliasacchi, M., Dufaux, F., Tubaro, S., and Ebrahimi, T., "Subjective assessment of H.264/AVC video sequences transmitted over a noisy channel," in [*Quality of Multimedia Experience, 2009. QoMEx 2009. International Workshop on*], 204–209 (July 2009).

[22] Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P., "Image Quality Assessment: From Error Visibility to Structural Similarity," *Trans. Img. Proc.* **13**, 600–612 (Apr. 2004).

[23] Huynh-Thu, Q. and Ghanbari, M., "Scope of validity of PSNR in image/video quality assessment," *Electronics Letters* **44**, 800–801 (June 2008).