IMAGE TRANSMORPHING WITH JPEG

Lin Yuan and Touradj Ebrahimi

Multimedia Signal Processing Group, EPFL, Lausanne, Switzerland

ABSTRACT

Picture-related applications are extremely popular because pictures present attractive and vivid information. Nowadays, people record everyday life, communicate with each other, and enjoy entertainment using various interesting imaging applications. In many cases, processed images need to be recovered to their original versions. However, most approaches require storage or transmission of both original and processed images separately, which result in increased bandwidth and storage resources to be used. In contrast, in this paper, we present a JPEG transmorphing algorithm, which converts an image to its processed version while preserving sufficient information about the original image in the processed image. It does this by inserting partial information about the original image in the application markers of the processed JPEG image file, so that the original image can be later recovered. Experiments are conducted and results show that the proposed method offers a number of attractive features and a good performance in many applications.

Index Terms— JPEG, Transmorphing, application marker, image compression, data hiding, privacy protection, image editing

1. INTRODUCTION

With the popularization of high-quality digital cameras, smart mobile devices with high-resolution cameras, as well as user-friendly imaging and social networking applications, taking pictures, then editing and sharing, have become part of everyday life for many. Popular imaging applications include Photoshop (professional graphics editor), Picasa (integrated image organizer), Instagram (photo sharing), Snapchat (photo messaging), and tons of image-related games.

The success of digital imaging applications is in part due to the development of effective standards such as JPEG [1] and JPEG 2000 [2]. JPEG is one of the early standards and is *de facto* the most popular compression format to store or transmit images thanks to its efficiency and low complexity. JPEG 2000 is a more recent standard for still image coding, offering efficient image compression, progressive transmission, seamless scalability, region of interest coding, and error resilience. However, even though JPEG 2000 outperforms JPEG in terms of compression efficiency, JPEG has remained the most popular format in a large variety of consumer imaging applications.

JPEG also offers a solution to tag images. JPEG/Exif (Exchangeable image file format) [3] is a popular way for digital cameras and other photographic capture devices to tag capture and location related metadata about photos. JPEG/JFIF (JPEG File Interchange Format) [4] is the most popular format for storage and transmission of images on the World Wide Web. The two formats are often not distinguished from each other and are simply referred to as JPEG each with their own application segments (APP0 for JFIF, APP1 for Exif) in the header of a JPEG file. Recently, a new standardization activity called JPEG XT [5] has been initiated, addressing the needs of photographers for higher dynamic range (HDR) images [6, 7, 8] in both lossy and lossless coding [9, 10] while retaining backward compatibility to the established 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. Adopting this idea, any useful information can be embedded in the application markers of a JPEG file.

In many situations, a modified image needs to be recovered to its original version. The typical solution is to keep both the original and the modified versions of the image, resulting in an increase in needs for resources and careful management efforts. In this paper, we propose a more efficient solution that preserves sufficient information about the original image embedded in the modified version allowing its recovery. Two experiments are conducted to assess the efficiency and usability of the proposed algorithm.

The rest of the paper is structured as follows. Section 2 describes the proposed algorithm in detail. Then Section 3 describes the conducted experiments and the analysis of experimental results. Finally, Section 4 concludes the paper and discusses potential future work.

This work has been conducted in the framework of the Swiss SERI C12.0081 and Eurostars ToFuTV.

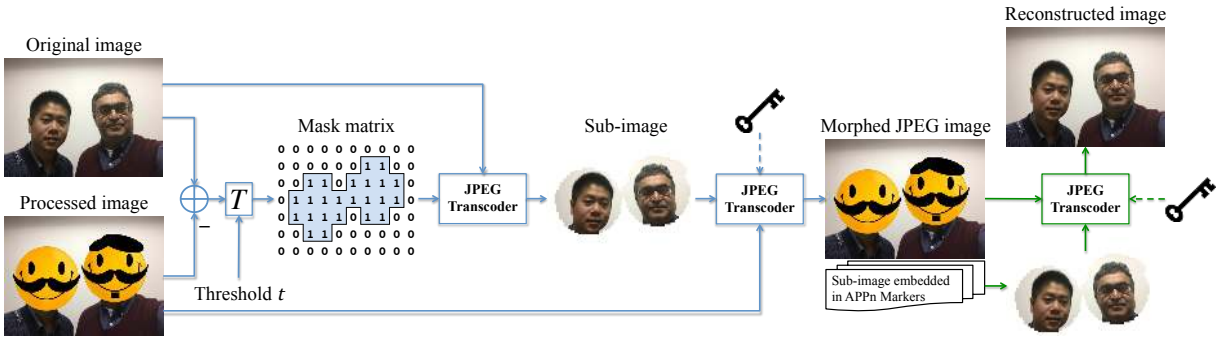


Fig. 1. JPEG Transmorphing algorithm.

2. JPEG TRANSMORPHING: THE ALGORITHM

The architecture of the JPEG transmorphing algorithm is illustrated in Fig. 1. The whole algorithm consists of two processes: transmorphing and reconstruction.

Assume that a digital image has been processed by one or a series of image editing tools applied either to the entire image or on specific regions in it. A typical example could be masking a human face in an image by replacing it with a smiley face. We define a JPEG transmorphing algorithm composed of the following operations:

Generation of mask matrix First, the difference between pixel values of the original and modified images is computed, and a binary image is generated by applying any adequate threshold. The elements 1 in the binary image are dilated to match 16×16 Minimum Coded Unit (MCU) blocks boundaries and then the binary image is subsampled by the factor of 16 such that each element points to a corresponding MCU block. The resulted binary image is referred to as *mask matrix*, in which elements 1 denote the regions or MCU blocks where the image is modified, noted as *modified regions* or *modified blocks*. The generation of mask matrix is performed for every color component in the image (e.g., RGB) followed by merging their corresponding mask matrices with logical OR operations. In many cases, calculation of mask matrix can be much simplified as the applications may know exactly the modified image regions. For instance, in a mobile application, a user can select and modify certain image regions by finger touch and the application can record the coordinates of touched points and would be able to generate the mask matrix easily.

Creation of sub-image Based on the mask matrix, a sub-image of the original image corresponding to the modified regions is created and encoded with JPEG, by assuming all DCT coefficients outside of the modified regions as zero. If the original image is in JPEG format, this can be done by transcoding the modified regions of the original image to sub-image. The sub-image contains the information about the original image corresponding to the modified regions.

Writing data in APPn markers Finally, the byte-stream of the JPEG sub-image, along with some metadata, is inserted in one or several application segments (APPn Markers) of the processed JPEG image. A security option has been added in the design of the proposed JPEG transmorphing algorithm to meet the needs for image security and privacy protection applications. In cases where security is needed, a Secure JPEG [11] framework can be applied on the sub-image with an appropriate image security tool (AES or scrambling) by means of a secret key. Hence, the inserted metadata contains the following information: (i) size of the sub-image in bytes, (ii) the method used to secure sub-image, and (iii) elements of the mask matrix. We use eight bytes (a `long` integer) in APPn data to record the file size of the sub-image, and three bytes to represent the security measure (protection type and parameters, without secret key) applied on the sub-image. The elements of the mask matrix are encoded to a bitstream which is then written to another few bytes in the APPn marker. The overhead added to the transmorphed JPEG image is mainly impacted by the size of the JPEG encoded sub-image, while the metadata overhead is negligible.

Reconstruction process aims at recovering the original image from the transmorphed image. This can be done by reversing the transmorphing operations described above: extracting sub-image byte-stream and metadata from APPn markers, creating the sub-image and mask matrix, and then replacing the DCT coefficients corresponding to the modified blocks of the morphed image with that of the sub-image. In case the sub-image is protected, a secret key needs to be provided to decrypt or descramble the extracted sub-image.

Such a transmorphing algorithm involves four inputs: original image, processed image, mask matrix and secret key. In practice, the input original and processed images can be in any format but are assumed to have the same pixel resolution. The output transmorphed image and the reconstructed image are of course in JPEG format. Our algorithm is implemented based on an open source JPEG library version 6b maintained by the Independent JPEG Group (IJG)[1].

[1] http://www.ijg.org/

3. EXPERIMENT AND ANALYSIS

In the JPEG transmorphing algorithm, inserting additional information into a JPEG image file increases the storage or bandwidth requirements. Besides, because image editing is usually done in the pixel domain and the modified image is re-encoded into JPEG, not all minor differences between the original and the modified images due to JPEG re-encoding are taken into account. Therefore, the reconstructed image is not guaranteed to be exactly the same as the original image, unless threshold value is set to zero. In this section, we present two experiments to assess (i) the relation between the size of overhead and the size of modified regions, and (ii) the influence of thresholding on size of overhead and the quality of reconstructed images.

3.1. Size of Overhead vs. Size of Modified Regions

In the first experiment, we used 1000 images from The Images of Groups Dataset [12]. The maximal length or width of those images is 1024 pixels and their file sizes range from 100 KB to 329 KB, with 163 KB as the average size. For each image, we detect the positions of human faces and insert the face regions (rectangle shape) into the image itself using the JPEG transmorphing algorithm. In this case, we assume that those images have been modified in regions of people's face and that the processed images have the same file sizes as their original version. Haar feature face detection from OpenCV was used[2]. For each of the 1000 images, the increase of bitrate versus the size of modified regions (both in percentage) is plotted as a dot in Fig. 2.

The result presents a linear trend between the increase in image bitrate and the size of modified regions, though it is not an exact linear relationship, due to the difference in image content. Compared to keeping both original and processed images, which introduces doubled overhead to storage and bandwidth, this method requires relatively lower level of overhead to bitrate, depending on the size of modified regions. For some images, even though the number of modified blocks is zero, the overhead to bitrate is non-zero. This is because no face is detected in those images and therefore an "empty" sub-image along with metadata is inserted, which still carries a small volume of information, although the DCT coefficients of the "empty" sub-image are all zero.

3.2. Size of Overhead vs. Reconstruction Quality

We then analyze the trade-off between size of overhead and reconstruction quality, which is controlled mainly by the threshold value. In this experiment, we used six images[3] which are respectively processed by six different image processing tools in Adobe Photoshop: masking, oil-painting, blurring, pixelation, inpainting, and thresholding. All the original and processed images were decoded and re-encoded to JPEG with a standard decoder and encoder from IJG using the same compression settings (100 quality factor and 4:2:0 chroma subsampling). Those images are shown in Fig. 3 and their basic information is listed in Table 1. We ran JPEG transmorphing and reconstruction on each pair of the original and processed images using 10 different threshold values: $0, 1, 2, 5, 10, 20, 30, 40, 50$, and 60. The increase in bitrate of the transmorphed image compared to the processed image, as well as the peak signal-to-noise ratio (PSNR) between the reconstructed and original images are computed for each threshold value. Experimental results of all six images are presented in Fig. 4.

In general, as threshold value increases, size of overhead and PSNR both decrease, which is because a lower threshold can detect smaller differences between original and modified images and therefore ensures more information about the original image to be preserved in the transmorphed JPEG file. In case the threshold is set to zero, the sub-image is the same as the original image so the complete information about the original image is inserted, which enables a perfect reconstruction but maximal overhead size. On the other hand, if the threshold is too high, some modified blocks will be ignored

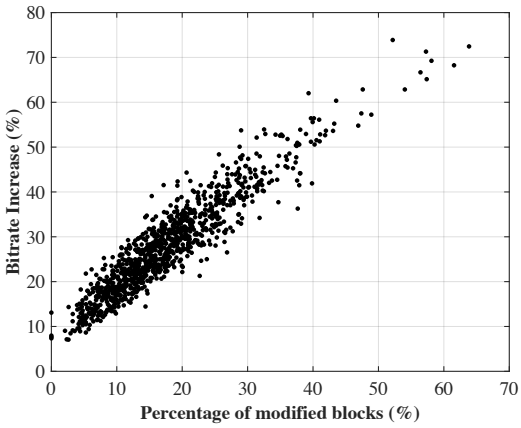


Fig. 2. Size of bitrate overhead vs. size of modified regions.

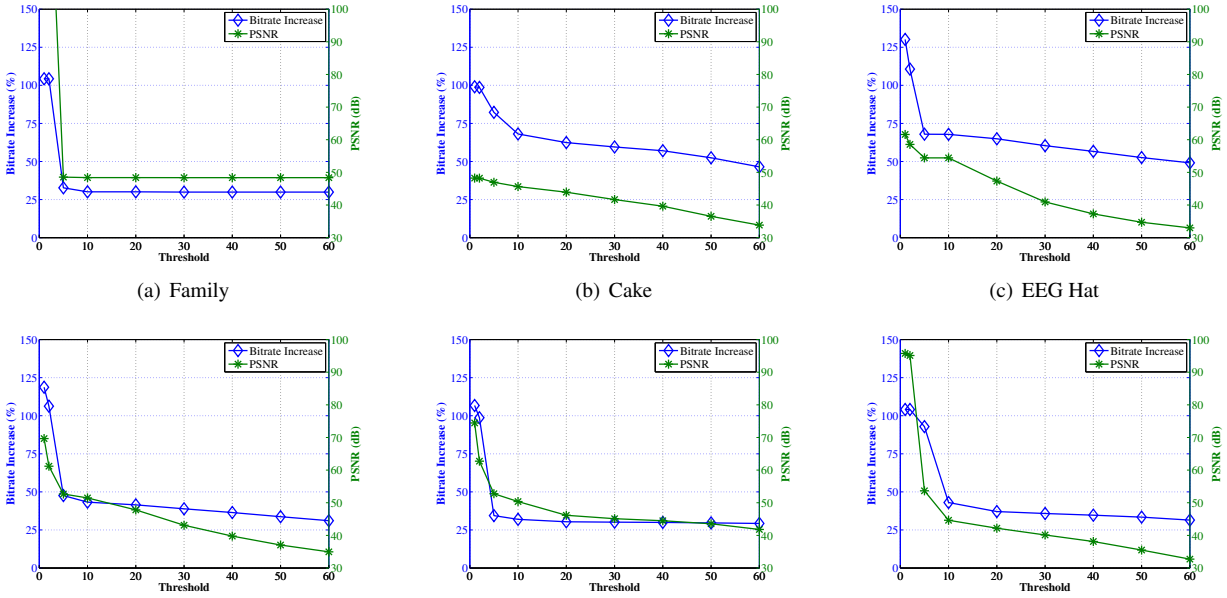
Table 1. Information of six images for the second experiment

Image name	Pixels	Original file size	Processed file size
Family	282×500	161 KB	155 KB
Cake	500×375	157 KB	157 KB
EEG Hat	960×1280	685 KB	507 KB
Christmas	3264×2448	3.8 MB	3.2 MB
Castle	4000×2667	6.5 MB	6.1 MB
Bear	5000×3333	8.3 MB	8.0 MB

[2]http://docs.opencv.org/modules/objdetect/doc/cascade_classification.html

[3]Images 'Family' and 'Cake' are from Flickr Retrieval [13]

Fig. 3. Six pairs of images for the second experiment. From left to right: Family, Cake, EEG Hat, Christmas, Castle and Bear, respectively. Original images in the first row; Processed images in the second row.



(a) Family (b) Cake (c) EEG Hat

(d) Christmas (e) Castle (f) Bear

Fig. 4. Influence of threshold on overhead size and reconstruction quality.

(see Fig. 5 for illustration) so that the quality of the reconstructed image is much reduced. In practice, a trade-off between bitrate and reconstruction quality can be archived by using a proper threshold value. As shown in the results, for all images, 10 is considered to be a good threshold value that ensures a satisfactory PSNR above 40 dB while keeping a small overhead size approximating to the actual size of modified regions (in percentage).

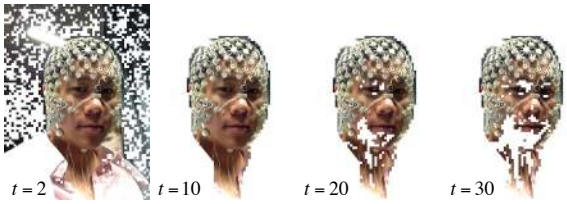


Fig. 5. Sub-images created using different thresholds.

4. CONCLUSION

This paper presents a JPEG transmorphing algorithm that allows for reverting back when modifying an image without need to keep a copy of the original. Results have shown acceptable bitrate overhead and good quality of reconstruction. Future work lies in further analysis of the influence of other parameters such as the quality factor of modified JPEG coded image, and adaptation of the algorithm for image security and privacy protection applications.

5. REFERENCES

[1] G. K. Wallace, "The JPEG still picture compression standard," *IEEE Transactions on Consumer Electronics*, vol. 38, no. 1, pp. xviii–xxxiv, Feb 1992.

[2] A. Skodras, C. Christopoulos, and T. Ebrahimi, "The JPEG 2000 still image compression standard," *IEEE Signal Processing Magazine*, vol. 18, no. 5, pp. 36–58, Sep 2001.

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

[4] B. Brower, R. Clark, A. T. Hinds, D. T. Lee, and G. J. Sullivan, "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.

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

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

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

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

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

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

[11] F. Dufaux and T. Ebrahimi, "Toward a Secure JPEG," in *Proc. SPIE*, 2006, vol. 6312.

[12] A. Gallagher and T. Chen, "Understanding images of groups of people," in *Proc. CVPR*, 2009.

[13] M. J. Huiskes and M. S. Lew, "The MIR Flickr retrieval evaluation," in *MIR '08: Proceedings of the 2008 ACM International Conference on Multimedia Information Retrieval*, New York, NY, USA, 2008, ACM.