Motion JPEG Video Authentication based on Quantization Matrix Watermarking: Application in Robotics

Lamri Laouamer
Lab-STICC,European University of Brittany
Brest, 29238, France
Al Qassim University
Buraydah,15452,KSA

Abdelhamid Benhocine
Dept. Info. Syst
Al Qassim University
Buraydah,15452,KSA

Laurent Nana
Lab-STICC
European University of Brittany
Brest, 29238, France

Anca Pascu
Lettres et sciences
European University of Brittany
Brest, 29238, France

ABSTRACT

The images authentication transmitted through the communication networks must verify the proof of the originality and robustness against the hacker attacks. The existing techniques such as the cryptographic methods are not sufficient. An effectiveness and robust solution is proposed in this paper. This solution is based on the watermarking of the video and especially the Motion JPEG stream. We focused on one of the major properties of the JPEG image which is the quantization matrix. The watermarking is performed on this matrix. We detail the obtained results against several attacks.

General Terms

Image security, image watermarking, data authentication.

Keywords

MJPEG video stream, Quantization matrix, Watermarking, Attacks.

1. INTRODUCTION

The security of the visual data collected through a communication network becomes a very important aspect to proof the authenticity and the origin of the transmitted data. The digital processing methods are so available and easy to use that it makes the falsification and the hacking so easy [1]. It is therefore necessary to incorporate in the capture systems and the control stations techniques that secure data against all kinds of malicious manipulations. The first techniques used to solve this problem are based on cryptography [8]. Using correctly the video must respect several constraints such as: the generation of a minimum volume of additional data, the malicious attacks detection, the tolerance against involuntary attacks (compression, changing format, etc. ..), the attacks location on the images constituting the video stream and the data recovery after attacks. Unfortunately, the used cryptographic techniques do not satisfy the constraints mentioned above [2, 3]. The generated data are so much and the introduced security is very limited to the verification of the data integrity rather than the visual content.

Against these limitations relative to cryptographic methods used in content protection, many researchers have turned to other approaches that allow security to take into account the constraints mentioned before. It means the non neglect of the cryptographic concepts that may play an additional advantage in strengthening security. These new approaches consist of digital watermarking the data and authenticating the content. These approaches allow us to extract the watermark (embedded secret) and provide information on data origin, authenticity and all the forgery made.

There are already advanced works that allow us to believe that the authentication data by marking the video is possible [1, 2]. Among the few existing authentication methods, we find the works in [1, 3, 4, 5, 6]. These authors use the watermarking in the both domains (color and transformed) with the techniques that are already presented in image watermarking technology. They assume that a perfect equality between the original and the extracted watermark is a proof of authenticity of the data and any difference means a possible attack. The same evaluations on image/video watermarking are also concerned with the video authentication field where the authors faced the same situations seen in image/video watermarking in terms of robustness against attacks.

In this paper, we present a new watermarking technique for addressing the problem of recovering rights and authentication of images included in Motion JPEG video streams sent by different wireless robots (*wifibots*). This technique is based on watermarking the compressed data or the quantization matrix which is one of the main steps in JPEG compression. The quantization matrix is standard but each camera may be recognized by its own one after watermarking so that the identification of the source is easy.

2. JPEG COMPRESSION

The JPEG compression process is depicted in Fig. 1.

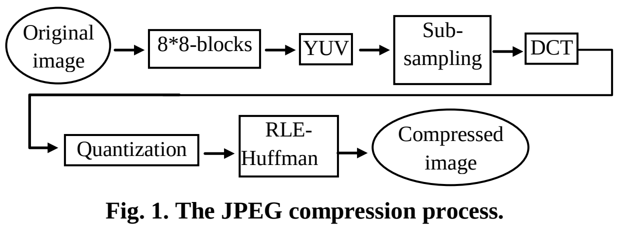

Fig. 1. The JPEG compression process.

The decompression process is the inverse.

After partitioning the RGB original image into 8*8-blocks $B_{r,s}$ r,s=0...7, each $B_{r,s}$ is transformed into 8*8-YUV block $C_{r,s}$ using the following matrix:

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.437 \\ 0.615 & -0.515 & -0.100 \end{bmatrix} * \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

then we proceed to 4:2:0 sub sampling Y, U and V, that means we leave Y unchanged (the eye is sensitive to luminance represented by Y), and consecutive 4*4 blocks in U and V are replaced by one element with average value (the eye is less sensitive to chrominance represented by U an V). Information is lost at this step.

U11	***U12***	U13	U14	U15	U16	U17	U18
U21	***U22***	U23	U24	U25	U26	U27	U28

8*8-Matrix U

(U11+U12+ U21+U22)/4	(U13+U14+ U23+U24)/4	(U15+U16+ U25+U26)/4	(U17+U18+ U27+U28)/4

4*4-New matrix U

To get the frequencies, we apply DCT independently to each block; note that the low frequencies contain the essential information in an image, so they must not change but the high frequencies can be eliminated by quantization. The DCT is defined by

$$d_{ij} = \sum_{r,s=0,7} \frac{w(r)w(s)}{4} cos\frac{\pi}{16}r(2i+1)cos\frac{\pi}{16}s(2i+1)C_{rs}$$

where $w(0)=1/\sqrt{2}$ and $w(r>0)=1$. The next step implements another lossy part of JPEG compression where coefficients $d_{i,j}$ are divided by quantization matrix and rounded to integers.

$$D_{ij} = round\left[\frac{d_{ij}}{Q_{ij}}\right], \qquad i,j \in \{0,\dots,7\}$$

The decompression process works in the reverse order. Blocks of quantized DCT coefficients are recovered from the JPEG file and multiplied by quantization steps, $\hat{d}_{ij} = Q_{ij} \, . D_{ij}$. The resulting values are transformed using the inverse Discrete Cosine Transformation (IDCT) and rounded to integers in the range [0, 255] (for 8 bits image). The decompressed block $\hat{B}$ is:

$$\hat{B}_{ij} = trunc([IDCT(Q_{ij} \, . D_{ij})]) \;\; , \;\; i,j \in \{0,\dots,7\}$$

3. PROPOSED APPROACH

The recovery and the decompression processes of the images through a *wifibot* are achieved by three main procedures: the image recovery from the camera, the image decompression and finally the image storing/displaying [7]. Fig. 2 shows how the image pixels are transmitted.

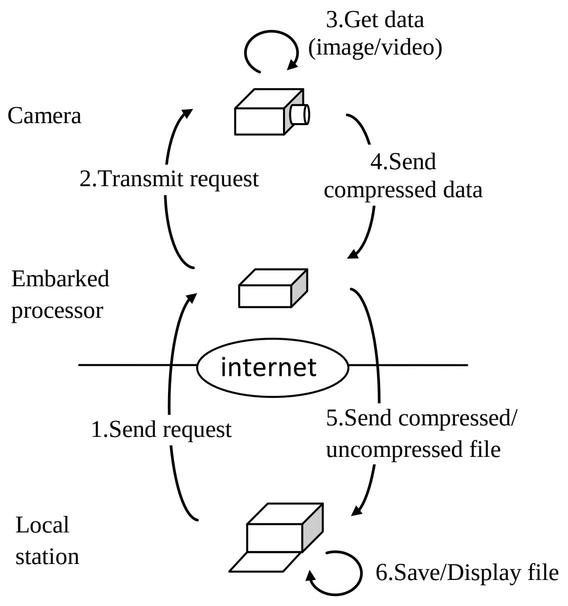

Fig.2. Request transmission

These procedures are defined in the *jpeglib* of the *wifibot* [7]. The first procedure consists to transfer the image from the *wifibot* camera with JPEG format to the *wifibot* processor (The compression program is directly implemented in the camera). This step is described by the *wbGetImage()* function. The second procedure is designed to decompress the image transmitted by the *wifibot's* camera, it is described by the *wbDecompressImage()* function. The last algorithm is defined by the *wbWriteImage()* function which stores the decompressed image in an uncompressed format (such as PPM). Normally, this function should not be used frequently because of the slowness of the *wifibot* file system. The following programs are implemented in the *wifibot* processor and executed via the interface implemented in the local station. When sending a request from the local station, various data are sent, like the resolution (for example 640*480), the speed, the rotation angle of the camera, the direction, the number of frames/second, choosing between receiving image or video, etc,...

```
#define SIZE 150000
int main(void)
{
int sz, width, height;
unsigned char *image;
unsigned char buffer[SIZE];
image = (unsigned char*)
            malloc(640*480*3*sizeof(unsigned char));
sz = wbGetImage(buffer);
sz = wbDecompressImage((unsigned char*)image,
                        buffer, sz, &width, &height);

wbWriteImage("out.ppm", (unsigned char*)image,
              sz, width, height);
free(image);
return 0;
}
```

The main program supposes a compressed image with size 150 Ko (SIZE=150000) and a decompressed image with size 900 Ko (resolution=640*480*3).

```
int wbGetImage(unsigned char *buffer)
{
unsigned char buf_rec[SIZE];
int img_size = 0;
int retval;
int sk;
sk = wbInitialiseSocket();
send(sk, "GET /IMAGE.JPG HTTP/1.0\r\nUser-Agent:
        \r\nAuthorization: Basic \r\n\r\n", 128, 0);
retval = recv(sk, buf_rec, SIZE, 0);
while (retval != 0)
{
    memcpy(buffer+img_size, buf_rec, retval);
    img_size += retval;
    retval = recv(sk, buf_rec, SIZE, 0);
}
for(i=0; i<img_size-1; i++)
{
    if (buffer[i] == 0xFF && buffer[i+1] == 0xD8)
    {
    img_size -= i;
    memmove(buffer, buffer+i, img_size);
    break;
    }
}
close(sk);
return img_size;
}
```

Before calling wbGetImage(), *buffer* contains the JPEG compressed image or MJPEG video **with** http header and variable sz contains its size. After calling wbGetImage(), *buffer* contains the JPEG compressed image or MJPEG video **without** http header and sz the new size. Note that FFD8 and FFD9 are respectively the marker of start and end of JPEG images, so that MJPEG videos are easy to manipulate. At this step either we transfer the data in *buffer* directly into a file (so with JPEG format), or we decompress them into *image* and save them in a file with any format, (for example PPM) for displaying.

```
int wbDecompressImage
(unsigned char *image,
unsigned char *buffer,
int sz, int *width, int *height)
{
struct jpeg_decompress_struct cinfo;
struct jpeg_error_mgr jerr;
unsigned char *line;

cinfo.err = jpeg_std_error(&jerr);
jpeg_create_decompress(&cinfo);
jpeg_tab_src(&cinfo, buffer, sz);
jpeg_read_header(&cinfo, TRUE);
jpeg_start_decompress(&cinfo);
*height = cinfo.output_height;
*width = cinfo.output_width;
line = image;
while(cinfo.output_scanline < (*height))
{
line = image +
    3 * (*width) * cinfo.output_scanline;
jpeg_read_scanlines(&cinfo, &line, 1);
}
jpeg_finish_decompress(&cinfo);
jpeg_destroy_decompress(&cinfo);
return 3* (*height) * (*width);
}
```

```
int wbWriteImage(char *fichier, unsigned char *img,
                 int sz, int width, int height)
{
FILE *fp;

fp = fopen(fichier, "w");
if (fp == NULL)
{
fprintf(stderr, "Error");
return(1);
}

fprintf(fp, "P6 %d %d 255 ", width, height);
for (i=0; i<taille; i++) fprintf(fp, "%c", img[i]);

return fclose(fp);
}
```

Our "bytes of interest" are in *buffer*.

The quantization matrix is detected by the hexadecimal marker *FFDB* and its length (8*8 bytes). In fact, two quantization matrices are used in JPEG compression: the first one Q_Y is for Y and corresponds to the first FFDB and the second one Q_{UV} is for U and V and corresponds to the second FFDB. We only consider Q_Y.

The compressed data corresponding to each image are detected by FFDA and FFD9.

To achieve watermarking of the compressed image/video **before** sending the data to the local station, we suggest two possibilities:

1. We modify the quantization matrix used in each JPEG image constituting the MJPEG video stream. The modification may be the same or not the same for all images. This way is used for authentication of our data and possibly for identifying the *wifibot* (if many).
2. We watermark each image compressed data using linear interpolation of these data and the compressed data of a watermark stored in the embarked processor. One can use the algorithms in [10]. This way is suitable for copyright.

4. TESTS

Fig. 3 is an example of an MJPEG video stream of 40 ms with 11 frames, so its speed is about 30 frames / second.

Each frame in the JPEG video includes a standard quantization matrix Q_Y, so one can define a different Q_Y for each frame.

When receiving the stream, a program in the local station can check frames one by one through the quantization matrix. If it is different from ours, then we deduce that it is a foreign image which can be embedded by voluntary or involuntary way.

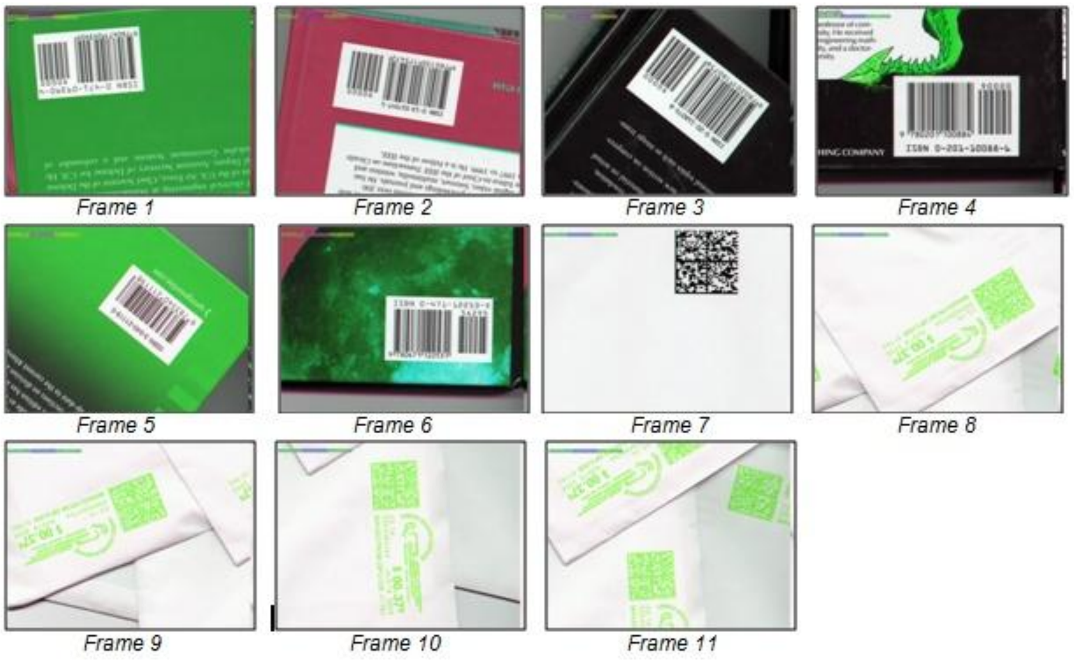

Fig. 3. Example of an MJPEG video stream.

Fig.4 and Fig.5 illustrate an example of the quantization matrix watermarking respectively by adding and subtracting different value of ε ∈{1, 5, 10, 50} to frame 5. We note that a small modification of the quantization matrix (eg. ε = ± 1) does not affect the image quality.

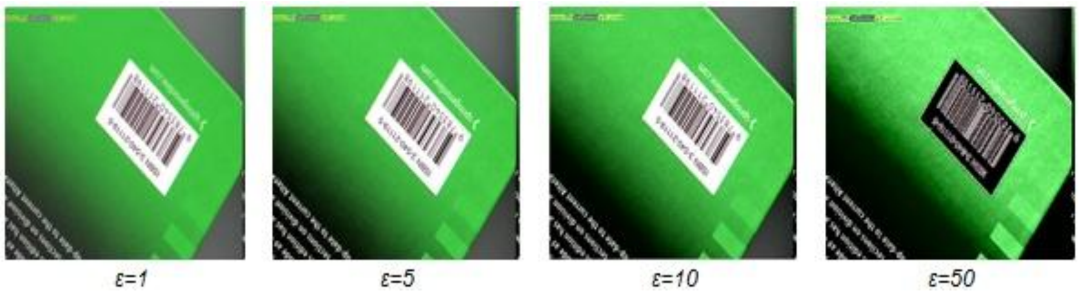

Fig 4. Quantization matrix modification by addition of different values of ε.

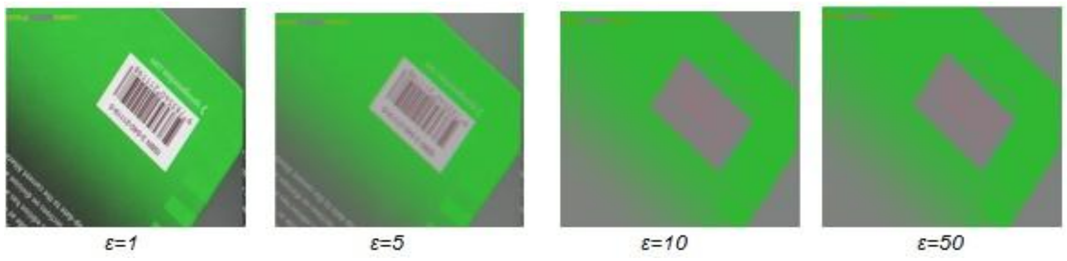

Fig 5. Quantization matrix modification by subtraction of different values of ε.

5. CONCLUSION

We presented in this paper a new approach for video watermarking for an application in robotics. We were interested particularly in the video stream of Motion JPEG. This kind of video is most common in the *wifibot*. The proposed approach is based on the watermarking of the quantization matrix which is one of the most important properties in the JPEG files. Watermarking the quantization matrix represents a major advantage in terms of time. It consists of just watermarking a little quantity of data relatively to watermarking all the image and it manipulates the compressed image instead of the image itself. The results are very significant since the local station can recognize the *wifibot* sending the stream. This will motivate us to integrate this module in the sending process of the video through a communication network.

6. REFERENCES

[1] Bartolini, F. Tefas, A. Barni, M. Pitas, I, *" Image authentication techniques for surveillance applications "*, Proceedings of the IEEE, Vol. 89, Issue 10, October 2004.

[2] Rey, C. and Dugelay, J.L, "*A survey of watermarking algorithms for image authentication*", EURASIP Journal on Applied Signal Processing, June 2002, pp. 613– 621.

[3] Boult, T.E, *"PICO: privacy through invertible cryptographic obscuration",* Proceedings of the Computer Vision for Interactive and Intelligent Environment CVIIE'05, 2005.

[4] Lin, C. and Chang, S, "*A robust image authentication method distinguishing JPEG compression from malicious manipulation*", IEEE Trans. On Circuits and Systems of Video Technology, vol.. 11, no. 2, February 2001, pp. 153–168.

[5] Sun, Q.B., Chang, S.F., Kurato, M. and Suto, M, "*A new semi-fragile image authentication framework combining ECC and PKI infrastructure*", ISCAS02, Phoenix, USA, May 2002.

[6] Kesavan Gopal, Madhavi Latha , *"Watermarking of Digital Video Stream for Source Authentication"*, IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 4, No 1, July 2010.

[7] Julien Le Guen and Cédric Bernier, *" Utilisation des wifibots"*, équipe MaIA, 2007.

[8] Che-Yen Wen and Kun-Ta Yang," *Image authentication for digital image evidence*", Forensic Science Journal, Vol 5, pages1-11, 2006.

[9] Thamodaran.K, Kuppusamy K, "*A Noval Security Mechanism for Image Authentication and Copyright Protection*", International Journal of Computer Applications, July 2010.

[10] Benhocine A, Laouamer L, Nana L et Pascu A, *"A New Approach Against Color Attacks of Watermarked Images"*. Proceedings of the International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIHMSP), IEEE Computer Society. Harbin, Chine. pp. 969-972. Août 2008.