International Journal of Database Theory and Application
Vol. 2, No. 2, June 2009

Data Hiding in Video

Arup Kumar Bhaumik[1], Minkyu Choi[2], Rosslin J.Robles[3], and Maricel O.Balitanas[4]

[1]Heritage Institute of Technology, Kolkata-700107, India
[2, 3,4]Hannam University, Daejeon, Korea
arupbhaumik@gmail.com[1], freeant7@naver.com[2], {rosslin_john[3], jhe_c1756[4]}@yahoo.com

Abstract

Here, we propose a data hiding and extraction procedure for high resolution AVI (Audio Video Interleave) videos. Although AVI videos are large in size but it can be transmitted from source to target over network after processing the source video by using these Data Hiding and Extraction procedure securely. There are two different procedures, which are used here at the sender's end and receiver's end respectively. The procedures are used here as the key of Data Hiding and Extraction.

Keywords: *Data hiding, AVI, security, multimedia.*

1. Introduction

Currently, Internet and digital media are getting more and more popular. So, requirement of secure transmission of data also increased. Various good techniques are proposed and already taken into practice.

Data Hiding is the process of secretly embedding information inside a data source without changing its perceptual quality. Data Hiding is the art and science of writing hidden messages in such a way that no one apart from the sender and intended recipient even realizes there is a hidden message. Generally, in Data Hiding, the actual information is not maintained in its original format and thereby it is converted into an alternative equivalent multimedia file like image, video or audio which in turn is being hidden within another object. This apparent message is sent through the network to the recipient, where the actual message is separated from it.

The requirements of any data hiding system can be categorized into security, capacity and robustness Cox et al. (1996). All these factors are inversely proportional to each other creating the so called data hiding dilemma. The focus of this paper aims at maximizing the first two factors of data hiding i.e. security and capacity coupled with alteration detection. The proposed scheme is a data-hiding method that uses high resolution digital video as a cover signal. the proposed recipient need only process the required steps in order to reveal the message; otherwise the existence of the hidden information is virtually undetectable. The proposed scheme provides the ability to hide a significant quality of information making it different from typical data hiding mechanisms because here we consider application that require significantly larger payloads like video-in-video and picture-in-video.

The purpose of hiding such information depends on the application and the needs of the owner/user of the digital media. Data hiding requirements include the following:

a. Imperceptibility- The video with data and original data source should be perceptually identical.
b. Robustness- The embedded data should survive any processing operation the host signal goes through and preserve its fidelity.
c. Capacity-Maximize data embedding payload.
d. Security- Security is in the key.

Data Hiding is the different concept than cryptography, but uses some of its basic principles [1].

In this paper, we have considered some important features of data hiding. Our consideration is that of embedding information into video, which could survive attacks on the network.

2. Previous works

As video file consist of several image sequence, so considering the data hiding technique of image will also apply for video data hiding we get:

2.1. Least-significant bit modifications

The most widely used technique to hide data, is the usage of the LSB. Although there are several disadvantages to this approach, the relative easiness to implement it, makes it a popular method. To hide a secret message inside a image, a proper cover image is needed. Because this method uses bits of each pixel in the image, it is necessary to use a lossless compression format, otherwise the hidden information will get lost in the transformations of a lossy compression algorithm.

When using a 24 bit color image, a bit of each of the red, green and blue color components can be used, so a total of 3 bits can be stored in each pixel. Thus, a 800 × 600 pixel image can contain a total amount of 1.440.000 bits (180.000 bytes) of secret data. For example, the following grid can be considered as 3 pixels of a 24 bit color image, using 9 bytes of memory:

(00100111 11101001 11001000)

(00100111 11001000 11101001)

(11001000 00100111 11101001)

When the character A, which binary value equals 10000001, is inserted, the following grid results:

(00100111 11101000 11001000)

(00100110 11001000 11101000)

(11001000 00100111 11101001)

In this case, only three bits needed to be changed to insert the character successfully.

On average, only half of the bits in an image will need to be modified to hide a secret message using the maximal cover size. The resulting changes that are made to the least significant bits are too small to be recognized by the human eye, so the message is effectively hidden.

While using a 24 bit image gives a relatively large amount of space to hide messages, it is also possible to use a 8 bit image as a cover source. Because of the smaller space and different properties, 8 bit images require a more careful approach. Where 24 bit

images use three bytes to represent a pixel, an 8 bit image uses only one. Changing the LSB of that byte will result in a visible change of color, as another color in the available palette will be displayed. Therefore, the cover image needs to be selected more carefully and preferably be in grayscale, as the human eye will not detect the difference between different gray values as easy as with different colors.

2.2. Masking and filtering

Masking and filtering techniques, usually restricted to 24 bits or grayscale images, take a different approach to hiding a message. These methods are effectively similar to paper watermarks, creating markings in an image. This can be achieved for example by modifying the luminance of parts of the image. While masking does change the visible Properties of an image, it can be done in such a way that the human eye will not notice the anomalies.

Since masking uses visible aspects of the image, it is more robust than LSB modification with respect to compression, cropping and different kinds of image processing. The information is not hidden at the ”noise” level but is inside the visible part of the image, which makes it more suitable than LSB modifications in case a lossy compression algorithm like JPEG is being used [2].

2.3. Transformations

A more complex way of hiding a secret inside an image comes with the use and modifications of discrete cosine transformations. Discrete cosine transformations (DST)), are used by the JPEG compression algorithm to transform successive 8 x 8 pixel blocks of the image, into 64 DCT coefficients each. Each DCT coefficient F(u, v) of an 8 x 8 block of image pixels f(x, y) is given by:

$$F(u,v) = \frac{1}{4}C(u)C(v)\left[\sum_{x=0}^{7}\sum_{y=0}^{7} f(x,y) * cos\frac{(2x+1)u\pi}{16} cos\frac{(2y+1)v\pi}{16}\right]$$

where $C(x) = 1/\sqrt{2}$ when x equals 0 and $C(x) = 1$ otherwise. After calculating the coefficients, the following quantizing operation is performed:

$$F^{Q}(u,v) = \left\lfloor \frac{F(u,v)}{Q(u,v)} \right\rfloor$$

Where Q(u, v) is a 64-element quantization table. A simple pseudo-code algorithm to hide a message inside a JPEG image could look like this:

```
Input: Message, cover image
Output: steganographic image containing message
While data left to embed do
        Get next DCT coefficient from cover image
        If DCT not equal to 0 and DCT not equal to 1 then
                Get next LSB from message
                Replace DCT LSB with message bit
        End if
        Insert DCT into steganographic image
End while
```

Although a modification of a single DCT will affect all 64 image pixels, the LSB of the quantized DCT coefficient can be used to hide information. Lossless compressed images will be suspect able to visual alterations when the LSB are modified. This is not the case with the above described method, as it takes place in the frequency domain inside the image, instead of the spatial domain and therefore there will be no visible changes to the cover image [3].

When information is hidden inside video the program or person hiding the information will usually use the DCT (Discrete Cosine Transform) method. DCT works by slightly changing the each of the images in the video, only so much though so it's isn't noticeable by the human eye. To be more precise about how DCT works, DCT alters values of certain parts of the images, it usually rounds them up.

For example if part of an image has a value of 6.667 it will round it up to 7. Data Hiding in Videos is similar to that of Data Hiding in Images, apart from information is hidden in each frame of the video. When only a small amount of information is hidden inside of video it generally isn't noticeable at all, however the more information that is hidden the more noticeable it will become.

3. Proposed work

The main high resolution AVI fie is nothing but a sequence of high resolution image called frames. Initially I will like to stream the video and collect all the frames in bitmap format (Figure 1). And also collect the following information:

a. Starting frame: It indicates the frame from which the algorithm starts message embedding.
b. Starting macro block: It indicates the macro block within the chosen frame from which the algorithm starts message embedding.
c. Number of macro blocks: It indicates how many macro blocks within a frame are going to be used for data hiding. These macro blocks may be consecutive frame according to a predefined pattern. Apparently, the more the macro blocks we use, the higher the embedding capacity we get. Moreover, if the size of the message is fixed, this number will be fixed, too. Otherwise it can be dynamically changed.
d. Frame period: It indicates the number of the inter frames, which must pass, before the algorithm repeats the embedding. However, if the frame period is too small and the algorithm repeats the message very often, that might have an impact onto the coding efficiency of the encoder [4].

Apparently, if the video sequence is large enough, the frame period can be accordingly large. The encoder reads these parameters from a file. The same file is read by the software that extracts the message, so as both of the two codes to be synchronized.

After streaming the AVI video file into AVI frames I will like to use the conventional LSB replacement method. LSB replacement technique has been extended to multiple bit planes as well. Recently [5] has claimed that LSB replacement involving more than one least significant bit planes is less detectable than single bit plane LSB replacement. Hence the use of multiple bit planes for embedding has been encouraged. But the direct use of 3 or more bit planes leads to addition of considerable amount of noise in the cover image. Still as my work is in high resolution video so I am getting a RGB combination of each pixel as in Figure 2 hence if I consider one LSB I will have a choice of 3 bits for each pixel. That will overcome the clam of [5]. And will give a higher security of the Data Hiding method.

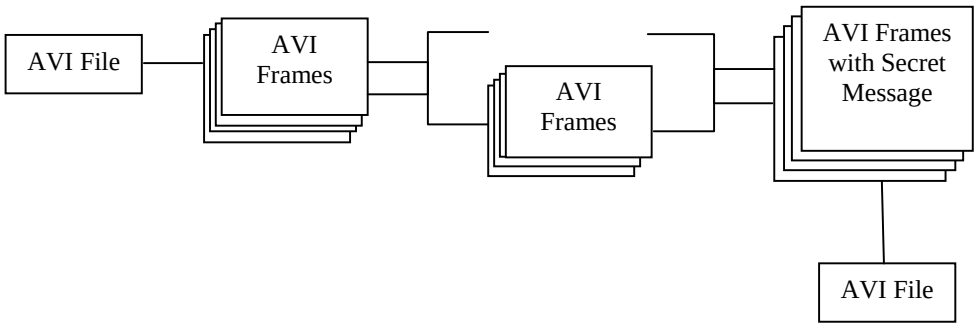

Figure 1. AVI video Streaming and Data Hiding Algorithm.

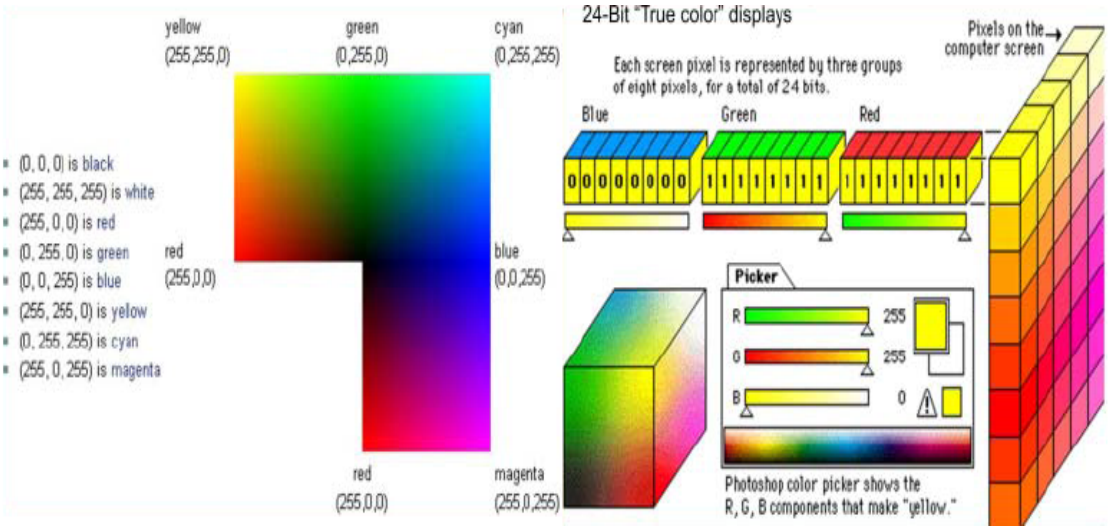

Figure 2. The RGB bit Pattern.

4. Data size estimation

Each frame of the Video is taken a data source for Data Hiding. First the maximum size of the hiding data is calculated as shown in Figure 3. The size of the image is 2000 × 1000 and modified it to 2048 × 1024. On further calculations we get 786,432,000 chars that can be embed. We have followed the following equation mentioned below:

(((Width × height) × 3 bits)/8 bits)/3 bytes × 3000 frames = char/video

And the image Bitmap size = 2048 × 1024

Step of calculations the maximum of hiding information:

a. Each frame consist = 2048 × 1024 = 2, 097, 152 Pixels.
b. Each pixel include 3 bytes (One byte we use single bit for encode data hiding) R = 1 bit, G = 1 bit and B = 1 bit.
c. Each frame = Pixels ×3 = 2, 097, 152 × 3 = 62, 915, 456 bits.

d. Each frame we can maximum hiding data is 62, 915, 456 bits/8 bits = 786, 432 bytes.
e. If this video 3000 frames = 786, 432 × 3000 = 2, 359.296, 000 bytes (1 bytes = 1 Character).
f. For 1 Character of Unicode we need 3 bytes/1 character of Unicode = 2, 359.296,000 bytes/3 =786, 432, 000 chars.

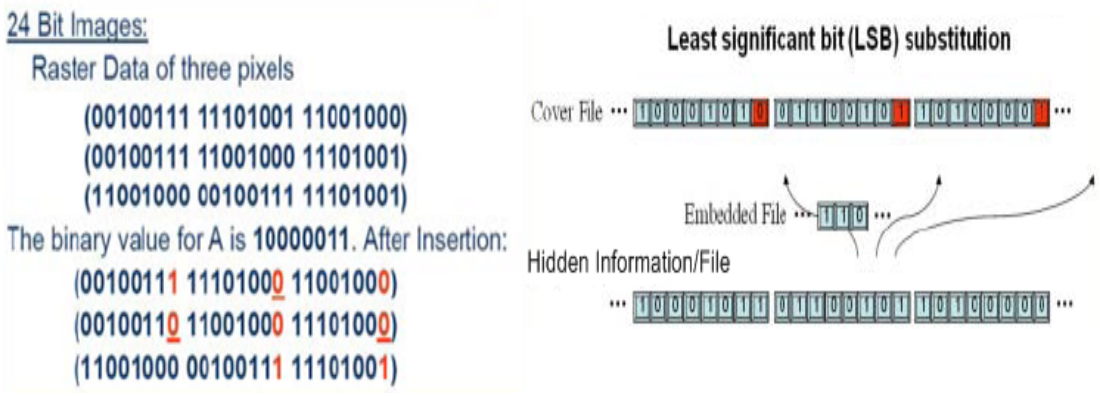

Figure 3.

5. Conclusion

In this paper I propose a data hiding technique for high resolution video. Our intension is to provide proper protection on data during transmission. For the accuracy of the correct message output that extract from source we can use a tools for comparison and statistical analysis can be done. Its main advantage is that it is a blind scheme and its affect on video quality or coding efficiency is almost negligible. It is highly configurable, thus it may result in high data capacities. Finally, it can be easily extended, resulting in better robustness, better data security and higher embedding capacity.

6. Acknowledgement

This work was supported by the Security Engineering Research Center, granted by the Korea Ministry of Knowledge Economy. This work has successfully completed by the active support of Prof. Tai-hoon Kim, Hannam University, Republic of Korea and Prof. Purnendu Das, Heritage Institute of Technology, Kolkata, India.

References

[1] Debnath Bhattacharyya, P. Das, S. Mukherjee, D. Ganguly, S.K. Bandyopadhyay, Tai-hoon Kim, “A Secured Technique for Image Data Hiding”, Communications in Computer and Information Science, Springer, June, 2009, Vol. 29, pp. 151-159.
[2] S.K. Bandyopadhyay, Debnath Bhattacharyya, D. Ganguly, S. Mukherjee and P. Das, “A Tutorial Review on Steganography”, International Conference on Contemporary Computing (IC3-2008), Noida, India, August 7-9, 2008, pp. 105-114.
[3] Steganography and Steganalysis, J.R. Krenn, January 2004.
[4]Spyridon K. Kapotas, Eleni E. Varsaki and Athanassios N. Skodras, “Data Hiding in H.264 Encoded Video Sequences”, IEEE 9th Workshop on Multimedia Signal Processing, October 1-3, 2007, Crete, pp. 373-376 .

[5] A. Ker,“Steganalysis of Embedding in Two Least-Significant Bits”, IEEE Trans. on Information Forensics and Security, vol. 2, no. 1, March 2007, pp. 46-54.