LSB based Image Steganography by using the Fast Marching Method

Xiaoli Huan[1], Hong Zhou[2], Jiling Zhong[3]
Department of Computer Science, Troy University, Troy, Alabama, USA[1, 3]
Department of Mathematical Sciences, University of Saint Joseph, West Hartford, Connecticut, USA[2]

***Abstract*—This paper presents a novel approach for image steganography based on the Least Significant Bit (LSB) method. Most traditional LSB methods choose the initial embedding location of the cover image randomly, and the secret messages are embedded sequentially without considering the image pixels' values and positions. Our approach utilizes the user-selected seeds in the cover image to avoid the smooth/flat areas where cause a higher detection rate. Then the fast marching method is used to calculate T (the time of arrival of the front of the seeds) and propagate the seeds by computational dynamics. The front propagation process decides the embedding positions of the secret messages. The same algorithm can be used to retrieve the hidden information as well. The coordinates of the seeds are used as the shared key only known to the sender and receiver to add additional security protection. Peak Signal to Noise Ratio (PSNR) is evaluated to measure the quality of resulting images. The experiments show that the proposed approach generates results with high payload capacity and satisfied imperceptibility.**

Keywords—Image steganography; LSB; the fast marching method; coordinates; PSNR

I. Introduction

Steganography has been an ancient practice to hide secret information within a media in such a way other people cannot easily detect the presence of the hidden contents. Cryptography and steganography are both techniques used to prevent the third party from reading the secret messages. However, they differ in the respect that cryptography makes the data exposable but not understandable without having the proper key to decode, while steganography hides the secret data inside a media and this modification of the original media cannot be easily perceived. In nowadays, steganography is used in many legal or illegal applications. For example, embedded digital watermarking techniques are developed to identify the ownership of the property. It is also reported that terrorist groups had used steganography to exchange information due to their affordability compared to dedicated secure networks [1].

The basic structure of image steganography is composed of the following:

- Secret-message: The information is to be hidden and delivered.
- Cover-image: An original image is used as a media to embed the secret-message.
- Stego-image: After the cover-image embeds the secret-message, the resulting image is known as the stego-image.
- Stego-key: Additional information is used for embedding and extracting the secret-message. The stego-key is a shared key known to the sender and receiver only.

II. Related Work

Least Significant Bit (LSB) steganography is a popular technique in which the least significant bits (lowest bits) of pixels of the cover-image embed the secret-message. The changes to the cover-image are minimal and imperceptible to the human visual system [2]. However, the secret-message can be easily detected in the traditional LSB methods since the embedding positions are generated randomly and data are embedded sequentially [3]. These methods call for higher security features.

Steganographic methods which utilize a pixel's dependency on its neighborhood and psycho-visual redundancy to determine the smooth areas and edged areas in the gray level images are presented in [4]. However, in this method distortion is introduced and anyone is possible to recover the image due to its lack of stego-key protection. The approach in [5] uses a secret key to hide a secret-message in different channels of the LSB of a cover-image to protect it from unauthorized receivers. This method does not consider the pixels' values and positions in the cover-image. Therefore, the smooth/flat regions in the cover-image will be contaminated and cause low visual quality after data hiding. Edge adaptive schemes have been investigated. For example, the edge-detecting filter is used in [6]. Mean and standard deviation and canny edge detection are used in [7]. Methods hiding data around the edge boundary of an object are proposed in [8]. However, when the cover-image is mostly smooth or without sharp edges, the payload is limited in these approaches.

In our proposed method, we use the level set method to determine the embedding positions of the secret-message. The level set method (LSM) was proposed by S. Osher and J. Sethian in 1988 [9]. LSM is a computational technique for tracking interface motion over time and has various applications including image processing [10], fluid dynamics and physical modeling. LSM involves propagating a continuous scalar variable. "Considering G(t) to be a moving closed curve in two dimensions**.** An Eulerian formulation for the motion of the interface is produced. The motion of the interface propagates along its normal direction with speed F, where F can depend on many factors, including the curvature, normal direction, shape, position of the front, or underlying fluid velocity. The interface G(t) can thus be represented as the

zero-height level set of a function ϕ". For a more detailed description of level set methods, the reader is referred to Sethian's published book [11].

Let's assume that the interface either moves "outward" (F > 0) or "inward" (F < 0) during the interface motion. The arrival time of the interface front at each grid point (T(x,y)) can be calculated and used to determine the propagating process of the front. This is the so-called fast marching method. In this method, all the arrival time values are composed of a function T(x,y) which renders a surface. This surface tells the position of the interface front at any actual time T. "This surface is called the arrival time surface because it gives the arrival time of the interface passing at each grid point" [12].

The equation for the arrival time function is called boundary value formulation, which is

$$|\nabla T|F=1,\ F=e^{-\alpha|\nabla G_\sigma * I(x,y,z)|} \quad (1)$$

T=0 on Γ, where Γ is the initial location of the interface. α is the fast marching method exponential coefficient which is set to 60 in our algorithm. T was discretized by the quadratic equation [12]:

$$\left[\begin{array}{l}\max(D_{i,j}^{-x}T,0)^2+\min(D_{i,j}^{+x}T,0)^2+\\ \max(D_{i,j}^{-y}T,0)^2+\min(D_{i,j}^{+y}T,0)^2\end{array}\right]^{1/2}={}^{1}\!/_{F_{i,j}} \quad (2)$$

D^+ and D^- represent forward and backward difference operators. Equation (2) is solved at each grid point in the propagating process, and the root with the largest value is chosen as the correct viscosity result.

The advantages of using the fast marching method for LSB image steganography are the following:

- The seeds initially selected can be encrypted as the stego-key to add additional security protection.
- The user can choose seeds in non-smooth/flat regions in the cover-image to avoid low visual quality data hiding. The fast marching method enables image segmentation [13].
- The algorithm is straightforward to implement. The same algorithm can be used for embedding and retrieving the secret-message.

III. Proposed Algorithm

A. *Finding the Embedding Positions by the Fast Marching Method*

Equation (2) is applied in our image stenography algorithm. T=0 is assigned to the user-selected seeds in the cover-image. The user can select the seeds in the non-smooth/flat regions to avoid higher detection rate, and the seeds' positions can be encrypted as the stego-key. By solving (2), the front position at any time T can be obtained. This equation can be solved for each pixel in the cover-image until two times of the number of the hidden bytes is reached (one byte is hidden over two pixels). These pixels can then be mapped in a queue in the increasing order of T. The pixel with the smallest T is called first to hide the secret-message. In this way, the embedding order of pixels in the cover-image is obtained. Fig. 1 shows the user-selected seeds in non-smooth/flat regions, and the seeds propagate by using the fast marching method.

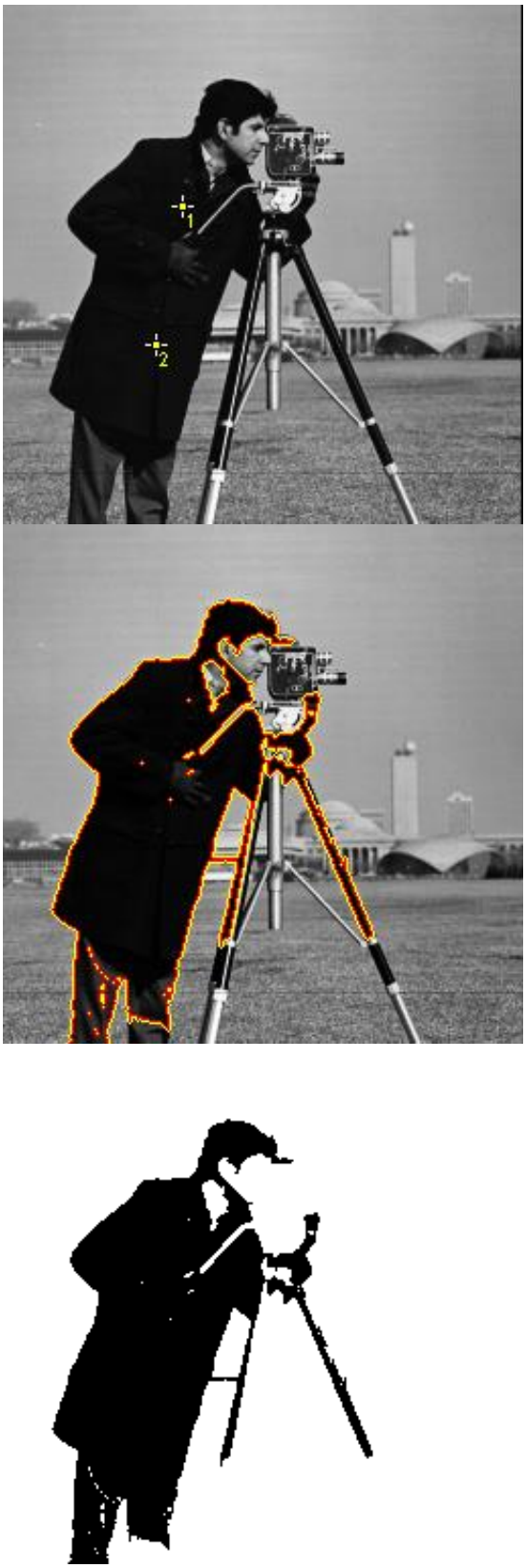

Fig. 1. The user-Selected Seeds Propagate by the Fast Marching Method.

The scheme of FMM has been briefly described above. A detailed process is shown in the following:

1) Initialize four vectors: far_away, alive, try and neighbor.

2) Assign max T (e.g., DBL_MAX in C++) for all pixels in the cover-image and set their status as far-away (push them into the far_away vector).

3) For each user-selected seed point:

a) set the smallest T (zero) value.

b) Remove it from the far_away vector and set the status as alive (push it into the alive vector).

c) Check its four adjacent points (up, down, left and right) in the cover-image and set their status as try (push them into the try vector). Calculate their T values by the following equation:

$$T(x,y)=1/\exp(-1 * 60 * grad\_mag[x][y])$$

grad_mag is the gradient magnitude value of the pixel. The cover-image first uses a Gaussian smoothing filter. Then the gradient magnitude values are computed in all color channels, and the values in the channel with the largest magnitude are picked [14].

4) For each point in the try vector:

a) Pick the point with the smallest T. Set the point as alive (push it into the alive vector) and remove it from the try vector.

b) Check its four adjacent points (up, down, left and right) in the cover-image. If the neighbor point is alive status, do nothing. If it is in the try vector, push it into the neighbor vector. If it is in far_away vector, push it into both the try and neighbor vectors and remove it from the far_away vector.

c) For each point (i, j) in the neighbor vector: Update its T. s1(a,b,c) and s2(a,b,c) are the functions to get the two roots of a quadratic function $\frac{-b\pm\sqrt{b^2-4ac}}{2a}$. There are 16 possible roots to consider for (2) and the largest root is picked as T:

```
double F = 1.0 / exp(-1 * 60 * grad_mag[i][j]);
root[0] = s1(2, -2 * (T[i][j-1] + T[i-1][j]), T[i][j-1]* T[i][j-1] + T[i-1][j] * T[i-1][j] - F * F);
root[1] = s2(2, -2 * (T[i][j-1] + T[i-1][j]), T[i][j-1]* T[i][j-1] + T[i-1][j] * T[i-1][j] - F * F);
root[2] = s1(2, -2 * (T[i][j-1] + T[i+1][j]), T[i][j-1]* T[i][j-1] + T[i+1][j] * T[i+1][j]-F * F);
root[3] = s2(2, -2 * (T[i][j-1] + T[i+1][j]), T[i][j-1]* T[i][j-1] + T[i+1][j] * T[i+1][j]-F * F);
root[4] = s1(2, -2 * (T[i][j+1]+T[i-1][j]), T[i][j+1] * T[i][j+1] +T[i-1][j] * T[i-1][j] -F * F);
root[5] = s2(2, -2 * (T[i][j+1]+T[i-1][j]), T[i][j+1] * T[i][j+1] +T[i-1][j] * T[i-1][j] -F * F);
root[6] = s1(2, -2 * (T[i][j+1]+T[i+1][j]), T[i][j+1]* T[i][j+1] + T[i+1][j] * T[i+1][j]-F * F);
root[7] = s2(2, -2 * (T[i][j+1]+T[i+1][j]), T[i][j+1]* T[i][j+1] + T[i+1][j] * T[i+1][j]-F * F);
root[8] = T[i][j-1]+F;
root[9] = T[i][j-1]-F;
root[10] = T[i][j+1]+F;
root[11] = T[i][j+1]-F;
root[12] = T[i-1][j]+F;
root[13] = T[i-1][j]-F;
root[14] = T[i+1][j]+F;
root[15] = T[i+1][j]-F;
```

d) Repeat step 4 until the number of points in the alive vector is greater than two times the number of hidden bytes. The pixel points in the alive vector are ordered by T ascendingly.

Fig. 2 shows an example of embedding orders with two user-selected seeds in a cover-image. T is calculated by using the fast marching method based on the image pixels' values and positions. The bands in the same color represent the pixels on the bands with the same T values. Pixels embed the secret-message in the increasing order of T.

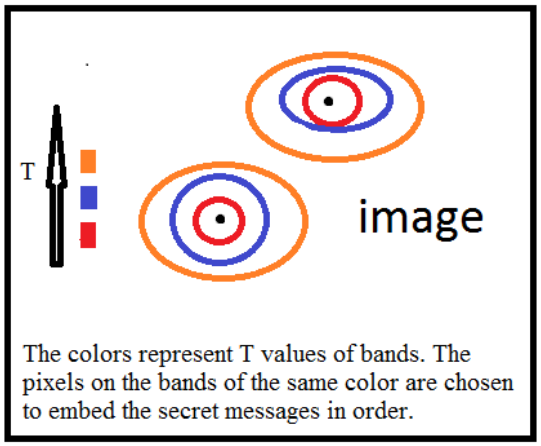

Fig. 2. Illustration of the Embedding Order with Two User-Selected Seeds.

B. Encoding the Secret-Message

After the embedding positions are obtained above, the algorithm starts the Least Significant Bit (LSB) encoding process. One byte of data is hidden into two adjacent points' R, G, B and Alpha channels in the alive vector. Any byte M from 0 to 255 can be extended to the form:

$$M = a_1 2^0 + b_1 2^1 + c_1 2^2 + d_1 2^3 + a_2 2^4 + b_2 2^5 + c_2 2^6 + d_2 2^7$$

If p^1 and p^2 are two adjacent elements in the alive vector of the cover-image and (p^1_r, p^1_g, p^1_b, p^1_a) and (p^2_r, p^2_g, p^2_b, p^2_a)are their RGB and Alpha values, the two new pixels' (p^{n1}, p^{n2}) values after M is embedded are:

$$p^{n1}_r = p^1_r - p^1_r \%2 + a_1$$

$$p^{n1}_g = p^1_g - p^1_g \%2 + b_1$$

$p_b^{n1}=p_b^1-p_b^1\%2+c_1$

$p_a^{n1}=p_a^1-p_a^1\%2+d_1$

$p_r^{n2}=p_r^2-p_r^2\%2+a_2$

$p_g^{n2}=p_g^2-p_g^2\%2+b_2$

$p_b^{n2}=p_b^2-p_b^2\%2+c_2$

$p_a^{n2}=p_a^2-p_a^2\%2+d_2$

C. Decoding the Secret-Message

Extracting the hidden data from the stego-image works similarly. The coordinates of the user-selected seed points can be encrypted as a shared stego-key. The extracting order is done by the fast marching method as the embedding process. If p^1 and p^2 are two adjacent elements in the alive vector of a stego-image and (p^1_r, p^1_g, p^1_b, p^1_a) and (p^2_r, p^2_g, p^2_b, p^2_a) are their RGB and Alpha values, the secret data can be constructed by:

$$M=(p_r^1\ \%2)2^0+(p_g^1\ \%2)\,2^1+(p_b^1\ \%2)2^2+(p_a^1\ \%2)2^3+$$

$$(p_r^2\ \%2)2^4+(p_g^2\ \%2)\,2^5+(p_b^2\ \%2)2^6+(p_a^2\ \%2)2^7$$

IV. Results and Analysis

The experimental results presented in this section compare the effectiveness of our proposed algorithm with existing methods. Several main factors affect an information hiding scheme: visual quality of the stego-images (HVS-human visual system) [15], embedding capacity, and error metrics such as PSNR.

Our experiment results show the proposed method achieves plausible HVS quality based on luminance similarity, structure correlation, edge similarity, and color similarity due to the nature of the fast marching method. It can have a larger payload capacity than methods such as [8].

We use the Mean Square Error (MSE) and the Peak Signal to Noise Ratio (PSNR) as the error metrics to evaluate stego-image quality. The MSE is computed by averaging the cumulative squared error between the original image and the stego-image, whereas PSNR represents a measure of the peak error. The following is the equation to compute MSE composed of d number of channels:

$$MSE=\frac{\sum_{m,n}[I_1(m,n)-I_2(m,n)]^2}{d\times m\times n}$$

The higher the value of PSNR, the closer is the stego-image to the cover-image. To compute the PSNR, the following equation is used:

$$PSNR=10\log_{10}\left(\frac{I\_MAX^2}{MSE}\right)$$

I_MAX is the largest possible variation in the input image data type. It is 255 in case of the simple single byte per pixel per channel.

Table I is the results of PSNR on original LSB, edge-based LSB [7] and our method. Our method has higher PSNR values.

TABLE I. Comparison of PSNR of LSB, EG_LSB and the Proposed Method

Method	Image Size	Hidden bits	PSNR
LSB	512*512	36584	51.12
EG_LSB	512*512	36584	54.22
Our Method	512*512	36584	65.67

Fig. 3 shows the cover-images Lena, Baboon and Pepper, the secret-message (image Baboon) and stego-images by using the proposed method. Table II is the comparison of PSNR of LSB with Four Neighbor method [4], Secret Key [5] and our method.

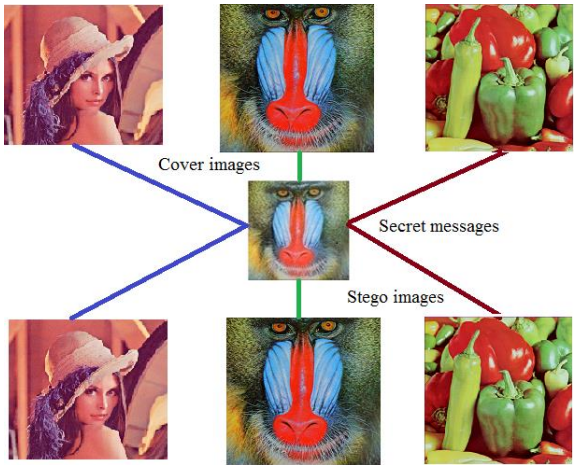

Fig. 3. Cover Images, Secret Messages and Stego Images by our Method.

TABLE II. Comparison of PSNR of LSB with Four Neighbor Method [4], Secret Key [5] and the Proposed Method

Cover image	Hidden bits	PSNR neighbor	PSNR secretKey	PSNR Our method
Lena	392208	41.15	53.76	55.48
Baboon	435223	36.52	53.75	55.03
Pepper	393567	41.03	53.78	55.44

V. Conclusion

In this paper, a novel approach for LSB image steganography by using the fast marching method is presented. The approach can avoid non-smooth/flat regions and the user-selected seeds can be used as the stego-key. The embedding and extracting positions are determined by the computational technique fast marching method based on image pixels' values and positions. The experiments show that the proposed method has plausible visual quality and desirable PSNR. Future work can include testing by using different kinds of steganalysis algorithms and extend the proposed method to other steganographic medias such as audio/video.

References

[1] M. Conway, "Code Wars: Steganography, Signals Intelligence, and Terrorism," Knowledge, Technology and Policy, Vols. 16, No. 2, no. Technology and Terrorism, p. 45~62, 2003.

[2] K. Bailey and K. Curran, "An Evaluation of Image Based Steganography," Multimedia Tools and Applications, vol. 30, no. 1, pp. 55-88, 2006.

[3] A. Pfitzmann and A. Westfeld, "Attacks on steganographic systems," in Proc. 3rd Int. Workshop on Information Hiding, 1999.

[4] M. Hossain, S. Haque and F. Sharmin, "Variable rate Steganography in gray scale digital images using neighborhood pixel information," in Proceedings of 2009 12th International Conference on Computer and Information Technology, Dhaka, 2009.

[5] S. Masud Karim, M. Saifur Rahman and M. Ismail Hossain, "A New Approach for LSB Based Image Steganography using Secret Key," in Proceedings of 14th International Conference on Computer and Information Technology, Dhaka, 2011.

[6] K. Hempstalk, "Hiding behind corners: Using edges in images for better steganography," in Proc. Computing Women's Congress, Hamilton, New Zealand, 2006.

[7] A. Chaturvedi and K. Doeger, "A Novel Approach for Data Hiding using LSB on Edges of a Gray Scale Cover Images," International Journal of Computer Applications, vol. 86, no. 7, 2014.

[8] M. Hussain and S. Haque, "Embedding data in edge boundaries with high PSNR," in 7th International Conference on Emerging Technologies, Islamabad, Pakistan, 2011.

[9] Osher and Sethian, "Fronts propagating with curvature dependent speed: algorithms based on Hamilton-Jacobi formulations.," Journal of Computational Physics, pp. 79:12-49, 1988.

[10] X. Huan, B. Murali and A. Ali, "Image restoration based on the fast marching method and block based sampling," Computer Vision and Image Understanding, vol. 114, no. 8, pp. 847-856, 2010.

[11] J. Sethian, Level Set Methods and Fast Marching Methods, Cambridge Univ. Press, 1999.

[12] J. Sethian, "A Fast Marching Level Set Method for Monotonically Advancing Fronts," in Proc. Natl. Acad. Sci., 1996.

[13] N. Forcadel, C. Le Guyader and C. Gout, "Generalized fast marching method: applications to image segmentation," Numerical Algorithms, vol. 48, p. 189, 2008.

[14] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Diego, CA, 2005.

[15] S. J. Thorpe, "Image Processing by the Human Visual System," in Advances in Computer Graphics, Berlin, Heidelberg, Springer, 1991, p. 309~341.