ISSN 2278 - 3091

International Journal of Advanced Trends in Computer Science and Engineering (IJATCSE), Vol.2 , No.5, Pages :64-69 (2013)
Special Issue of ICCECT 2013 - Held during September 20, 2013, Bangalore, India

Secret Fragment Visible Mosaic Image Using Genetic Algorithm

Nithya Francis[1], Naveen N[2]
[1]Dept of Applied Electronics & Instrumentation, Rajagiri School of Engineering & Technology, India, nithyaf@gmail.com
[2] Dept of Applied Electronics & Instrumentation, Rajagiri School of Engineering & Technology, India, Country, naveenn@rajagiritech.ac.in

Abstract *:* This paper presents an image hiding method using fragment visible mosaic image. In this method the secret image is divided into blocks or tiles and they are shuffled or reordered to become a target image in the mosaic form. In the existing method an image similarity measure, h-feature is defined using the color distribution in the pixels. The h-feature is used to select the most appropriate cover image for the secret image from an image database and also for the tile shuffling process. Since the tile re-ordering is done in a single iteration the performance is limited. So a genetic algorithm is used here in the tile shuffling by choosing PSNR as the match criterion to improve the quality of encrypted image.

Key words : Genetic Algorithm, h-feature, Mosaic image, PSNR.

INTRODUCTION

Mosaic is the art of creating images with an assemblage of small pieces of colored glass, stone, or other materials. This principle is utilized here in the area of image steganography. The new type of art image called secret-fragment-visible mosaic image contains small fragments of a given source image [2]. Observing such a type of mosaic image, one can see all the fragments of the source image, but the fragments are so tiny in size and so random in position that the observer cannot figure out what the source image looks like. Therefore, the source image may be said to be secretly embedded in the resulting mosaic image, though the fragment pieces are all visible to the observer. And this is the reason why the resulting mosaic image is named secret-fragment-visible. Because of this characteristic of the new mosaic image, it may be used as a carrier of a secret source image in the disguise of another, a target image of a different content. It is useful for the application of covert communication or secure keeping of secret images.

More specifically, a secret image is first divided into rectangular-shaped fragments, called tile images, which are fitted next into a target image selected from a database to create a mosaic image. The number of usable tile images for this operation is limited by the size of the secret image and that of the tile images. This is not the case in traditional mosaic image creation where available tile images for use essentially are unlimited in number because the tile images are not generated from the secret image and may be used repeatedly. Then, the information of tile-image fitting is embedded into some blocks of the mosaic image, which are selected randomly by a secret key. The random number generation is carried out using key based random permutation KBRP[8]. Accordingly, an observer possessing the key can reconstruct the secret image by retrieving the embedded information, while a hacker without the key cannot.

In the existing method the tile reordering is done in a single iteration. To enhance the quality of the encrypted image a method which uses a number of iterations to get a better reordering should be chosen. In this paper we use a genetic algorithm with population size 10, and take 10 generations. Thus a better PSNR is obtained.

Basic Idea of Secret Fragment Visible Mosaic Image

The formation of mosaic image which visually approximates the target image using the tiles or blocks from the secret image and the recovery of secret image from the encrypted mosaic image has the following basic steps

1. Construction of a color image database for use in selecting similar target images for given secret image
2. Creation of a secret-fragment-visible mosaic image using the tile images of a secret image and the selected similar target image as input
3. Recovery of the secret image from the created secret fragment- visible mosaic image

The first step includes selection of a wide variety of images and also calculation of some image similarity measures and histogram. The second step includes

1. Searching the database for a target image the most similar to the secret image
2. Fitting the tile images in the secret image into the blocks of the target image to create a mosaic image
3. Embedding the tile-image fitting information into the mosaic image for later secret image recovery

The third step corresponds to the decryption. It has the following steps

1. Retrieving the previously-embedded tile-image fitting information from the mosaic image
2. Reconstructing the secret image from the mosaic image using the retrieved information

ISSN 2278 - 3091
International Journal of Advanced Trends in Computer Science and Engineering (IJATCSE), Vol.2 , No.5, Pages :64-69 (2013)
Special Issue of ICCECT 2013 - Held during September 20, 2013, Bangalore, India

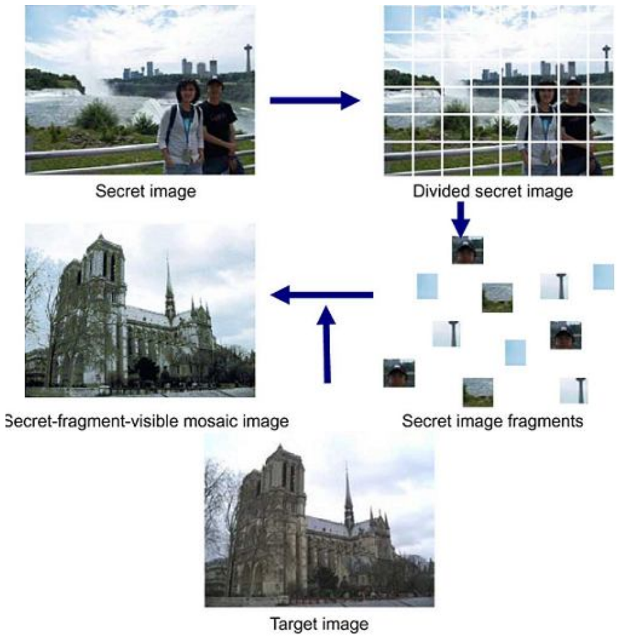

Fig 1: A figure showing the basic method of fragment visible mosaic image creation

SECRET-FRAGMENT-VISIBLE MOSAIC IMAGE CREATION

Database Construction

The target image database plays an important role in the mosaic image creation process. If a selected target image from the database is dissimilar to a given secret image, the created mosaic image will be distinct from the target one, resulting in a reduction of the information hiding effect. To generate a better result, the database should be as large as possible. Searching a database for a target image with the highest similarity to a given secret image is a problem of content-based image retrieval. In general, the content of an image may be described by features like shape, texture, color, etc. Due to the use of small tile images in the proposed method, which are the fragments of the secret image, it is found in this study that the most effective feature, which affects the overall visual appearance of the resulting mosaic image, is color. Therefore, we focus on extracting color distributions from images to define an appropriate image similarity measure for use in target image selection in this study.

One way for extracting the global characteristic of the color distribution of an image is the 1-D color histogram transformation technique proposed by Smith and Chang[3]. The technique re-quantizes first the color values *(r,g,b)* into fewer levels, say, N_r, N_g and N_b ones, respectively, resulting in the new colour values *(r', g', b')*. Then, it transforms the three new values into a single one by

$$f(r', g', b') = r' + N_r * g' + N_r * N_g * b' \qquad (1)$$

However, the use of this 1-D color value, originally proposed just for color indexing, is inappropriate for the study here where the human's visual feeling of image similarity is emphasized. Therefore a new color transformation function is defined as follows:

$$h(r', g', b') = b' + N_b * r' + N_b * N_r * g' \qquad (2)$$

The numbers of levels N_r, N_g and N_b are all set to 8. The largest weight N_b*N_r is assigned to the green channel value, *g'* and the smallest weight, the value 1, is assigned to the blue channel value, *b'*. This way of weight assignment is based on the fact that the human eye is the most sensitive to the green color and the least sensitive to the blue one, leading to a larger emphasis on the intensity of the resulting mosaic image[3]. The new color function h defined in equation 2 defines a 1D h-color scale.

To compute the similarity between a tile image in the secret image and a block in a target image for use in the tile-image fitting process during mosaic image creation, we propose a new feature for each image block (either a tile image or a target block), which is called h-feature, denoted as h_c, and computed by the following steps:

1. Compute the average of the RGB color values of all the pixels in image block as (r_c, g_c, b_c)
2. Compute the h-feature value h_c by equation 2 resulting in the following equation

$$h_c(r_c, g_c, b_c) = b_c + N_b * r_c + N_b * N_r * g_c \qquad (3)$$

We select a wide variety of images, all with a preselected size Z_c to form the database. Then divide each image, *D* in the database into target blocks of a preselected size Z_t, compute the h-feature value defined by equation 3 for each target block, generate accordingly an h-feature histogram, H_D of image, and finally save all the h-feature values of the target blocks and the histogram of images into the desired database *DB*.

Image Similarity Measure and Target Image Selection

Before generating the secret-fragment-visible mosaic image for a given secret image, *S* with the preselected size, Z_c , we have to choose from the database *DB* a target image which is the most similar to *S*. For this, first we divide *S* into blocks of the preselected size, Z_t, compute the h-feature values of all the resulting blocks by equation 3, and generate the h-feature histogram, H_S of *S*. Then, we define an image similarity value *m(S,D)* between *S* and each candidate target image *D* with h-feature histogram H_D in *DB* in the following way:

$$m(S, D) = \frac{1}{\sum_{h=0}^{511} |H_S(h) - H_D(h)|} \qquad (4)$$

Where $H_X(h)$ with $X = S$ or *D* is the number of image blocks in the bin of feature value *h*. The larger the value *m(S,D)* is, the more similar *D* and *S* are to each other. If the corresponding h-features in H_S and H_D are all identical, then *S* and *D* are regarded to be totally similar in the h-feature sense. After calculating the image similarity values of all the candidate target images in *DB* with respect to *S*, we select finally the image D_0 in *DB* with the largest similarity as the desired target image for *S* for use in mosaic image creation.

Fitting Tile images into Target Blocks

Given a secret image, after the most similar target image is selected, we have to find a tile image in *S* to fit into eachtarget block in D_0. This problem of fitting a limited number of tile images into a target *image* in an optimal way may be reduced to be a single-source shortest path problem. For this, we propose to use a greedy search algorithm to find suboptimal solutions which, though non optimal, are found feasible in this study for information hiding applications.

Also, we need a selection function for the greedy search algorithm to select a tile image *s* the most similar to each target block *d*. We use the previously mentioned concept of h-feature as the selection function. Specifically, we define the block similarity value *m(s,d)* between a tile image *s* with h-feature value h_s and a target block *d* with h-feature value h_d by

$$m(s, d) = \frac{1}{|h_s - h_d|} \quad (5)$$

This h-feature-based similarity measure takes into more consideration the relative intensity difference between the compared image blocks (the tile image and the target block), and helps creating a mosaic image with its content visually resembling the target image in a more global way.

Recovering the Secret Image

Another issue which should be dealt with in creating the mosaic image is how to embed the information of tile-image fitting so that the original secret image can be reconstructed from the created mosaic image. Each fitting of a tile image into a target block forms a mapping from *s* to *d*. The way we propose for dealing with the issue is to record these mappings into a sequence L_r, called the secret recovery sequence, and embed L_r into randomly-selected blocks in the created mosaic image using a technique of lossless least significant bit (LSB) replacement proposed by Coltuc and Chassery [9].

In more detail, to get the mappings, we start from the top leftmost target block d_1 in the selected target image D_0, and find for it the most similar tile image s_i in the secret image *S* in the sense of equation 3, and form the first mapping $s_i \rightarrow d_1$ to be included in L_r. Next, in a raster-scan order, we process the target block d_2 to the right of d_1 to find the most similar tile image s_j in the remaining tile images to form the second mapping $s_j \rightarrow d_2$ for L_r . Then, we do similarly to find the third mapping $s_k \rightarrow d_3$, and so on. In more detail, to get the mappings, we start from the top leftmost target block d_1 in the selected target image D_0 , and find for it the most similar tile image s_i in the secret image *S* in the sense of equation 3, and form the first mapping $s_i \rightarrow d_1$ to be included in L_r. Next, in a raster-scan order, we process the target block d_2 to the right of d_1 to find the most similar tile image s_j in the remaining tile images to form the second mapping $s_j \rightarrow d_2$ for L_r . Then, we do similarly to find the third mapping $s_k \rightarrow d_3$, and so on.

We continue this greedy search process until the last target block at the bottom-rightmost corner in the target image is processed. The resulting *Lr* may be regarded to include two block-index sequences,L_1 = *i, j, k....* and L_2 = *1,2,3...*with mappings $i \rightarrow 1$, $j \rightarrow 2$, $k \rightarrow 3$,... and so on. Since L_2 is a well-ordered sequence of 1,2,3.., we can ignore it and take L_r to include just L_1 to reduce the data volume of to be embedded.

Also, it is not difficult to figure out that if the width and height of a given secret image *S* are W_S and H_S, respectively, with Z_t being the previously-mentioned size of the tile images in *S*, then the number of tile images *N* in *S*, the number of bits N_X required to specify the index of a tile image, and the number of bits N_R required to represent the secret recovery sequence L_r , respectively, are as follows:

$$N = \frac{W_S * H_S}{Z_t} \quad (6)$$

$$N_X = \lfloor \log_2 N \rfloor + 1 \quad (7)$$

$$N_R = N * N_X \quad (8)$$

Thus we need to embed the sequence *i,j,k....* and the dimensions of the target image and the tiles into the preliminary mosaic image formed to get the final image. The numbers *i,j,k....* are embedded into some randomly selected tiles in the mosaic image using LSB replacement. This random number generation will be carried out using key based random permutation [8]. This method will generate a unique sequence for given length and an alphanumeric key. The final mosaic image can be decrypted only by knowing the sequence *i,j,k....* They can be retrieved from the mosaic image only with the knowledge of the secret key. The KBRP algorithm is explained in the next section. By making slight changes in some steps of the algorithm we can change the whole sequence to improve secrecy.

KEY BASED RANDOM PERMUTATION(KBRP)

A permutation, also called an arrangement number or order, is a rearrangement of the elements of an ordered list *S* into a one-to-one correspondence with *S* itself. The number of permutations on a set of *n* elements is given by *n*!

We can use any algorithm to generate a permutation based on a key such that a particular key should generate a unique sequence. For testing the mosaic image generation we have chosen the Key Based Random Permutation formulated by Shakir M. Hussain and Naim M. Ajlouni. It generates the random sequence needed in the encryption process[8]. KBRP is a method that can generate one permutation of size *N* out of *N!* permutations. This permutation is generated from certain key (alphanumeric string) by considering all the elements of this given key in the generation process. The permutation is stored in one dimensional array of size equal to the permutation size (*N*). The process involves three consecutive steps: init(), eliminate(), and fill().

First step, init(), is to initialize array of size N with elements from the given key, by taking the ASCII code of each element in the key and storing them in the array consecutively. To complete all elements of the array, we add elements to the array by adding two consecutive values of the array until all the elements of the array are set to values. Finally, all values are set to the range 1 to *N* by applying the mode operation.

The second step, eliminate(), is to get rid of repeated values by replacing them with value of zero and keep only one value out of these repeated values. Last step, fill(), is to replace all zero values with nonzero values in the range 1 to *N* which are not exist in the array. The resulted array now represents the permutation.

TILE SHUFFLING USING GENETIC ALGORITHM

In the earlier method tile shuffling is done by selecting the target blocks one by one in a raster scan order, and finding the most similar block from among the remaining blocks in the secret image. Since this procedure is performed in a single iteration, for the target blocks nearing the end in the scan order, the list of secret blocks from which match can be selected is very small. So the performance of this shuffling process is limited in this aspect. So, in this paper we introduce the use of a generic genetic algorithm to find the tile embedding sequence L_r. In the method h-feature is used to select the most appropriate target from the database. After the target selection the tile shuffling is done using genetic algorithm with population size 10 and maximum number of generation as 10. The mapping sequence, L_r previously mentioned is utilized in this method for initializing the population in the genetic algorithm. Let that sequence be named L_{rh}, since it is found with h-feature as the match criterion. While experimenting we found that for some images better mosaic result is obtained when we use h-feature as the match criterion. But in some other cases mosaic image generation is better if the match criterion is PSNR instead of h-feature. So another mapping L_{rp} is also found by similar method with PSNR as the match criterion. Now these two mappings L_{rh} and L_{rp} initialize the population of genetic algorithm, that means they are the candidate solutions. The genetic algorithm will refine these solutions through many generations. The Algorithm requires a measure to check whether the solution is good or bad. That is, it requires an objective function to measure the fitness of the solution. In this paper we have chosen PSNR as the measure for quality[7].

Genetic Algorithm(GA)

Genetic algorithms (GAs) are search methods based on principles of natural selection and genetics[4]. GAs encode the decision variables of a search problem into finite-length strings of alphabets of certain cardinality. The strings which are candidate solutions to the search problem are referred to as chromosomes, the alphabets are referred to as genes and the values of genes are called alleles.

To evolve good solutions and to implement natural selection, we need a measure for distinguishing good solutions from bad solutions. The measure could be an objective function that is a mathematical model or a computer simulation, or it can be a subjective function where humans choose better solutions over worse ones. In essence, the fitness measure must determine a candidate solutions relative fitness, which will subsequently be used by the GA to guide the evolution of good solutions.

Another important concept of GAs is the notion of population. Unlike traditional search methods, genetic algorithms rely on a population of candidate solutions. The population size, which is usually a user-specified parameter, is one of the important factors affecting the scalability and performance of genetic algorithms. For example, small population sizes might lead to premature convergence and yield substandard solutions. On the other hand, large population sizes lead to unnecessary expenditure of valuable computational time.

Once the problem is encoded in a chromosomal manner and a fitness measure for discriminating good solutions from bad ones has been chosen, we can start to evolve solutions to the search problem using the following steps

1. *Initialisation* - The initial population of candidate solutions is usually generated randomly across the search space. However, domain-specific knowledge or other information can be easily incorporated.
2. *Evaluation* - Once the population is initialized or an offspring population is created, the fitness values of the candidate solutions are evaluated.
3. Selection - Selection allocates more copies of those solutions with higher fitness values and thus imposes the survival-of-the-fittest mechanism on the candidate solutions. The main idea of selection is to prefer better solutions to worse ones, and many selection procedures have been proposed to accomplish this idea. In this paper we have used roulette-wheel selection.
4. *Recombination* - Recombination combines parts of two or more parental solutions to create new, possibly better solutions (i.e. offspring). There are many ways of accomplishing this and competent performance depends on a properly designed recombination mechanism. The offspring under recombination will not be identical to any particular parent and will instead combine parental traits in a novel manner
5. *Mutation* - While recombination operates on two or more parental chromosomes, mutation locally but randomly modifies a solution. Again, there are many variations of mutation, but it usually involves one or more changes being made to an individuals trait or traits. In other words, mutation performs a random walk in the vicinity of a candidate solution.
6. *Replacement* - The offspring population created by selection, recombination, and mutation replaces the original parental population.
7. Repeat steps 2-6 until a terminating condition is met.

Selection using Roulette wheel method

There are many methods used for the selection procedure. In this paper we use Roulette wheel method. It is a fitness proportionate selection. In Roulette wheel selection, each individual in the population is assigned a roulette wheel slot sized in proportion to its fitness. That is, in the biased roulette wheel, good solutions have a larger slot size than the less fit solutions. The roulette wheel is spun to obtain a reproduction candidate. The roulette wheel selection scheme can be implemented as follows:

1. Evaluate the fitness, fi , of each individual in the population.

2. Compute the probability (slot size), p_i , of selecting each member of the population:
   $p_i = f_i / \sum_{i=1}^{n} f_i$,where n is the population size.
3. Calculate the cumulative probability, q_i for each individual: $q_i = \sum_{j=1}^{i} p_j$
4. Generate a uniform random number, r$\epsilon(0, 1]$
5. If $r < q_1$ then select the first chromosome, x_1 else select the individual x_i such that $q_{i-1} < r \leq q_i$
6. Repeat steps 4-5 *n* times to create *n* candidates in the mating pool.

Recombination (Crossover) Operators.

After selection individuals from the mating pool are recombined (or crossed over) to create new, hopefully better, offspring. In the GA literature, many crossover methods have been designed. Many of the recombination operators are problem-specific but there are a few generic (problem independent) crossover operators.

In most recombination operators, two individuals are randomly selected and are recombined with a probability , called the crossover probability. That is, a uniform random number, *r*, is generated and if $r < p_c$, the two randomly selected individuals undergo recombination. Otherwise, that is, if $r > p_c$, the two offspring are simply copies of their parents.

In one point crossover a crossover site is selected at random over the string length, and the alleles on one side of the site are exchanged between the individuals.

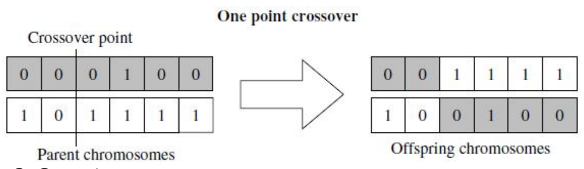

Fig 2: One point cross over

Mutation Operators

If we use a crossover operator, such as one-point crossover, we may get better and better chromosomes but the problem is, if the two parents (or worse, the entire population) has the same allele at a given gene then one point crossover will not change that. In other words, that gene will have the same allele forever. Mutation is designed to overcome this problem in order to add diversity to the population and ensure that it is possible to explore the entire search space.

One of the most common mutations is the bit-flip mutation. In bitwise mutation, each bit in a binary string is changed (a 0 is converted to 1, and vice versa) with a certain probability, p_m, known as the mutation probability. As mentioned earlier, mutation performs a random walk in the vicinity of the individual. Other mutation operators, such as problem-specific ones, can also be developed.

Replacement

Once the new offspring solutions are created using crossover and mutation, we need to introduce them into the parental population. There are many ways we can approach this. Bear in mind that the parent chromosomes have already been selected according to their fitness, so we are hoping that the children (which includes parents which did not undergo crossover) are among the fittest in the population and so we would hope that the population will gradually, on average, increase its fitness. Some of the most common replacement techniques are outlined below.

1. *Delete All:* This technique deletes all the members of the current population and replaces them with the same number of chromosomes that have just been created. This is probably the most common technique and will be the technique of choice for most people due to its relative ease of implementation. It is also parameter-free, which is not the case for some other methods.

2. *Steady-State:* This technique deletes n old members and replaces them with n new members. The number to delete and replace, n, at any one time is a parameter to this deletion technique. Another consideration for this technique is deciding which members to delete from the current population. Do you delete the worst individuals, pick them at random or delete the chromosomes that you used as parents? Again, this is a parameter to this technique.

3. *Steady-state-no-duplicates:* This is the same as the steady-state technique but the algorithm checks that no duplicate chromosomes are added to the population. This adds to the computational overhead but can mean that more of the search space is explored.

EXPERIMENTAL RESULTS

The database is created with 32 randomly selected images. The mosaic encryption was experimented using earlier h-feature based method and the genetic algorithm. The h-feature mapping obtained in the earlier method and a single iteration PSNR mapping are given as chromosomes or candidate solutions to the genetic algorithm. Algorithm was tested with different parameters as the fitness factor. The PSNR value and a dissimilarity factor derived from Pearson correlation ratio are used for evaluating the fitness of the solutions. The figures show some of the results obtained.

CONCLUSION

Secret Fragment Visible Mosaic Image can be used in covert communication and also for secure keeping of secret images. The secret image is divided into blocks or tiles and these tiles are reordered to form the mosaic image which visually looks like the target image. To get the most appropriate target image from a data base and to shuffle the tiles in the secret image into target image blocks a quantity called h-feature has been defined.

ISSN 2278 - 3091

International Journal of Advanced Trends in Computer Science and Engineering (IJATCSE), Vol.2 , No.5, Pages :64-69 (2013)
Special Issue of ICCECT 2013 - Held during September 20, 2013, Bangalore, India

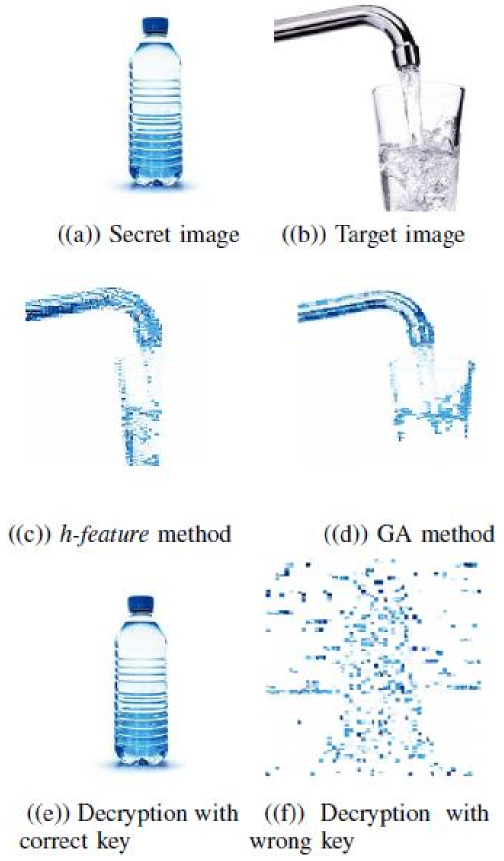

Fig 3: Comparison of encryption of bottle image using h-feature and GA methods

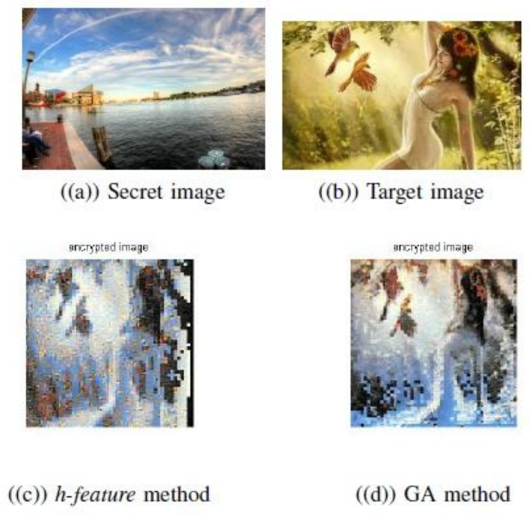

Fig 3: Comparison of encryption of harbor image using h-feature and GA methods

Table 1: Comparison of h-feature and GA methods

Bottle Image		
Method	Dissimilarity	PSNR
h-feature	38.6	13.4
GA	34.6	16.5
Harbor Image		
Method	Dissimilarity	PSNR
h-feature	168	8
GA	166	10.9

This h-feature is formulated by giving more consideration to the relative intensity difference, that is, humans' visual feeling of image similarity is emphasized. The tile mapping sequence is also embedded in randomly selected blocks. This random selection is based on a key and without this key the secret image cannot be retrieved. The existing method of tile shuffling is enhanced by utilizing a genetic algorithm to generate the tile mapping sequence. In the enhanced method h-feature mapping and a single iteration PSNR mapping are given as seeds for the genetic algorithm with population size 10 and maximum number of generations 10. PSNR between the mosaic image and the target image is selected as the fitness criterion of the mapping. We get a better PSNR than that with the earlier method of encryption. A dissimilarity factor is defined using correlation ratio for comparing the results. When genetic algorithm is used, the dissimilarity between the encrypted image and the target image is reduced. The future enhancement will be towards the formulation of a new fitness measure which gives emphasis to the visual plausibility of image for the genetic algorithm so that it can generate a mapping sequence that gives the best mosaic image. Future works may also be directed towards eliminating the use of an image database, thus allowing the users to select the target image freely.

REFERENCES

[1] Yu-Chee Tseng,Hsiang-Kuang Pan, "Data Hiding in 2 colour Images", *IEEE transactions on computers*,vol. 51, no.7,July 2002

[2] I-Jen Lai and Wen-Hsiang Tsai, "Secret-Fragment-Visible Mosaic ImageA New Computer Art and Its Application to Information Hiding"' *IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY*, VOL. 6, NO. 3 SEPTEMBER 2011.

[3] J. R. Smith and S. F. Chang, "Tools and techniques for color image retrieval", *ProcIS and T/SPIE*. vol. 2670, pp. 27, Feb. 1995

[4] SASTRY, GOLDBERG AND KENDALL, "Genetic Algorithms"

[5] Mehmet U. Celik, Gaurav Sharma, A. Murat Tekalp and Eli Saber," Reversible data hiding", 0-7803-7622-6/02/17.00 2002 IEEE.

[6] H. Joel Trussell, Eli Saber, and Michael Vrhel, "Color Image Processing [Basics and Special Issue Overview]", *IEEE SIGNAL PROCESSING MAGAZINE* [14] JANUARY 2005,1053-5888/05/20.002005IEEE

[7] SShen Wang, Bian Yang and Xiamu Niu, "A Secure Steganography Method based on Genetic Algorithm", *Journal of Information Hiding and Multimedia Signal Processing* Volume 1, Number 1, January 2010

[8] Shakir M. Hussain1 and Naim M. Ajlouni," Key Based Random Permutation (KBRP)", *Journal of Computer Science* 2 (5): 419-421, 2006

[9] D. Coltuc and J. M. Chassery, "Very fast watermarking by reversible contrast mapping", *IEEE Signal Process.* Lett., vol.14, no. 4, pp. 255258, Apr. 2007