Image hiding with an improved genetic algorithm and an optimal pixel adjustment process

Lin-Yu Tseng
Department of Computer Science and Engineering
Chung Hsing University
Taichung, Taiwan, R.O.C.
lytseng@cs.nchu.edu.tw

Yung-Kuan Chan
Department of Management Information Systems
Chung Hsing University
Taichung, Taiwan, R.O.C.
ykchan@nchu.edu.tw

Yu-An Ho
Department of Computer Science and Engineering
Chung Hsing University
Taichung, Taiwan, R.O.C.
yaho@cs.nchu.edu.tw

Yen-Ping Chu
Department of Computer Science and Engineering
Chung Hsing University
Taichung, Taiwan, R.O.C.
ypchu@dragon.nchu.edu.tw

Abstract

Image hiding techniques embed a secret image into a cover image. The fusion of the two, called a stego-image, fools grabbers who would not be conscious of the differences between the cover image and. the stego-image. A secret image can be transferred safely using this technique. In general, we would disarrange each pixel in the secret image and adjust all of them to form a suitable string of bits that could be embedded. Then these bits are embedded into the cover image in corresponding places, and this image would become a stego-image that hides secret data. This paper suggests a new image disarranging technique. It uses an improved genetic algorithm and an Optimal Pixel Adjustment Process, OPAP, to enhance the quality of a Stego-image. Experimental results show that a stego-image is indistinguishable from the cover-image. The stego-image can embed 4 bits per pixel, and the mean-square error of a stego-image is much lower than results for previous methods [1, 2, 4, 12].

1. Introduction

Using the Internet, secret data can be transmitted speedily. However receivers and senders use various encryption methods to prevent exposure to hackers [6-7, 9, 13, 14]. In order to protect transmitted data, important data is encrypted using encrypting and decrypting calculations. The process may use a series of ciphers that seem to be disordered nonsense to encrypt data before sending it [11]. At the receiving end, a decryption calculation is used. Through this method, encrypted data can be transmitted over the Internet. Even if hackers do obtain the encoding ciphers, if they cannot obtain the original secret data, they will never uncover the information that was hidden in the cipher.

The above-mentioned method has one weakness: encrypted ciphers attract hackers' attention. Therefore, if ciphers can be embedded into a general, natural image, hackers would be misdirected. This method is called data hiding. Image hiding techniques mainly deal with a secret image embedded within a cover image to get a Stego-image. This Stego-image conceals hidden data without advertising that it is hiding anything. One common method is called Least-significant-bit (LSB) [5, 8, 15-16]. LSB replaces a cover-image directly, after hiding a secret image within it. In general, bit-mapped images are commonly used. Every image is comprised of pixels, and each pixel indicates one color. If an image is shown in gray-scale, with a range of values from 0 to 255, lower values signify darkness, and higher values, lightness. Therefore, a gray-level image can be adjusted by adjusting the values. At least 8 bits are required to represent these values, and the binary system stores them from the most signification bit to the least signification bit $a_8 a_7 ... a_1$. LSB substitution replaces the least signification bit a_1 to make imperceptible changes that can't be recognized by human vision. For example, one pixel's gray-level may be 100. When it hides a 1, we modify the least signification bit to become pixel 101. This difference can't be recognized by human vision. In this way, we can hide another image.

Though LSB can easily embed secret data into an image, it would degrade image quality in a way that might alert hackers. Wang et al. [10] proposed a method called the moderately significant bit (MSB) which embedded secret data into a cover-image. Here the MSB means that the fourth bit of a gray pixel is a_4. When embedding, a genetic algorithm can be used to find the optimal substitution matrix for each pixel, and then perform a local pixel adjustment process (LPAP) to improve the quality of the stego-image. Chan et al.

978-0-7695-3382-7/08 $25.00 © 2008 IEEE
DOI 10.1109/ISDA.2008.235

[3] pointed out that the LPAP adjusts the fourth bit against LSB. Thus, the method cannot be applied for simple LSB substitution. This method uses a look-up table to adjust the image embedding. In addition to the fourth bit of LSB, other bits might be modified to achieve the best result.

Afterwards, Wang et al. [12] proposed a method which uses bijective mapping function to disarrange the secret data prior to embedding them into the cover image. Because of the numerous combinations after disarranging, this method goes a step further by utilizing a genetic algorithm to seek and find the optimal solution. Though this method has reduced calculating time, acquired bijective mapping function only approximates optimal solution. Chang et al. [1] utilized a dynamic programming strategy to determine the optimal solution, and even reduced the calculating time. In 2003, Thien et al., [2] proposed a 'high-hiding-capacity' method which can embed data via a digit by digit system in real time and achieve a much higher vision quality than LSB can, while avoiding resulting artificial edges of LSB. Against image hiding technique, Chan et al. [4] proposed an optimal pixel adjustment process, OPAP. The concept follows the method submitted by Wang et al. [10]. It handles the last 4 bits of the image and employs an optimal pixel adjustment process to lessen the related complicated calculating. Although the method that was submitted by Wang et al. [10] has improved the image quality substantially, in order to provide much greater cover image quality, this paper proposes a whole new image disarrange technique. We utilize an improved genetic algorithm and Optimal Pixel Adjustment Process, OPAP to enhance the quality of the cover image.

The remainder of the paper is organized as follows. The adjustment procedure of the optimal pixel will be introduced in the second section. The third section will introduce the hiding image technique that is proposed in this paper. The fourth section is the experimental results of the proposed method, and a contrast with other methods. The fifth section is the conclusion.

2. Optimal pixel adjustment process

This section will introduce OPAP as proposed by Chan et al. [4]. The main construct of OPAP follows the method that was submitted by Wang et al. [10]. It does optimal adjustment with the last k bits of an image, and uses low complicated calculating. The main calculation of OPAP is employed as algorithm *opap*:

Algorithm *opap*

Inputs: Transformed secret image S', Cover image C

Output: Stego-image C'

{

For $j \leftarrow 0$ to M-1

Do for $i \leftarrow 0$ to N-1

Do $D[j][i] = C[j][i] \bmod 2^k - S[j][i]$

If $(D[j][i] > 2^{k-1})$

Then if $(C[j][i] < 255 - 2^{k-1})$

Then $C'[j][i] = C[j][i] - D[j][i] + 2^k$

Else $C'[j][i] = C[j][i] - D[j][i]$

Else if $(D[j][i] < -2^{k-1})$

Then if $(C[j][i] > 2^{k-1})$

Then $C'[j][i] = C[j][i] - D[j][i] - 2^k$

Else $C'[j][i] = C[j][i] - D[j][i]$

Else $C'[j][i] = C[j][i] - D[j][i]$

}

Inputs are secret image S and cover image C; then secret image S is rearranged to become S' which is the same size as the cover image. The bit plane of S' is smaller than the k bit plane of C which is an embedded bit plane. M and N are the height and the width, and j and i are the coordinate of the image. $D[j][i]$ is the value that subtracts secret image S' from the place of LSB of C. If $C[j][i] = 100$, $S'[j][i] = 15$, k=4, then $D[j][i] = 100 \bmod 2^4 - 15 = -11$ is smaller than -2^3. As a result, we take 100-(-11)-16=95. This way, the difference will be smaller than 100+11=111 which is the direct embedding of k LSB bits from 100, and it could also reduce the value of the mean square error. In the end, it outputs adjusted stego-image C'.

3. The proposed method

This section introduces a secret image transform method. This method utilizes a genetic algorithm to determine the optimal transform seed to get the best quality cover image.

3.1. A secret image transform method

First, our method adjusts secret image S to be the same size as the cover-image. Suppose that the value of the pixel of gray-level is g-bit pixel value, the size of the cover-image is $M_c \times N_c$, and the size of the secret image $M_s \times N_s$; according to equation (1), it can then obtain the information on the lowest number of demand k bits of one pixel of the cover-image in order to embed it totally within the secret image. Therefore, it can transform the size of the secret image to equal

that of the cover image, and be shown as S'. Furthermore, each pixel is shown at least as *k* bits.

$$k = \left\lceil \frac{g \times M_s \times M_s}{M_c \times N_c} \right\rceil . \quad (1)$$

Then, take the acquired S' to perform the transform using a random seed. While transforming, at first it will result in a fixed size of transform block expressed by: $R = \{r[j][i] \mid 0 \le i, j \le n-1\}$, where $r[j][i] \in \{0,1,2,...,2^k - 1\}$. Moreover, according to the block size, S' will be segmented by $n \times n$ size, becoming $S'_m = \{q[j][i] \mid 0 \le i, j \le n-1\}$, here $q[j][i] \in \{0,1,2,...,2^k - 1\}$. *m* is the serial number of the segmented blocks. Then add each S'_m to R, and take the remainder of 2^k. As a result, it can get a disarranged secret image S'' which is shown as equation (2):

$$S''_m = (S'_m + R) \bmod 2^k . \quad (2)$$

Assume that *n* is 4, *k* is 4, and *m* is 1 (mean of the first block). There is a simple example as follows:

$$S'_1 = \begin{bmatrix} 3 & 2 & 10 & 2 \\ 5 & 14 & 0 & 2 \\ 3 & 2 & 3 & 1 \\ 7 & 8 & 9 & 8 \end{bmatrix}_{16}, \quad R = \begin{bmatrix} 3 & 7 & 7 & 3 \\ 6 & 0 & 13 & 5 \\ 1 & 2 & 2 & 6 \\ 8 & 4 & 11 & 12 \end{bmatrix}_{16},$$

$$S''_1 = (S'_1 + R) \bmod 2^4 = \begin{bmatrix} 6 & 9 & 1 & 5 \\ 11 & 14 & 13 & 7 \\ 4 & 4 & 5 & 7 \\ 15 & 12 & 4 & 4 \end{bmatrix}_{16} .$$

The acquired S''_1 is the remainder of 2^4 which is the result of adding S'_1 to R. Finally, embed S''_m to C to get the stego-image C'. We can use equation (3) peak signal-to-noise (PSNR) and equation (4) mean square error (MSE) to estimate the quality difference between C and C'. The definition of the function of PSNR is as follows:

$$\mathrm{PSNR} = 10 \log \frac{(255)^2}{MSE} dB \quad (3)$$

And the definition of *MSE* is as follow:

$$MSE = \left(\frac{1}{M \times N} \right) \sum_{j=1}^{M} \sum_{i=1}^{N} \left(C - C' \right)^2 \quad (4)$$

It can return to the S'_m only by subtracting R from S''_m. If there was a negative number in the resulting blocks, then add 2^k to return to S'_m.

3.2. Improved genetic algorithm

According to all possible results of *R* in Section 3.1, they can bring $(2^k)^{n \times n}$ kinds of changes. Therefore, calculating every change would take lots of time. The following introduce how to find out the approximate optimal solution by algorithm *GA*.

Algorithm *GA*

Inputs: Define the fitness function, number of generation size of population, crossover rate, mutation rate and selection rate.

Output: The best chromosome.

{

Generate the chromosomes of the first generation by random selection.

For $g \leftarrow 0$ to generation

Do Optimal Pixel Adjustment Process by chromosome

Evaluate the fitness value MSE by chromosome

If Satisfy the ending condition

Than Obtain the best solution

Break

Select the better chromosomes

Recombine new chromosomes by using crossover

Recombine new chromosomes by using mutation.

}

First, it has to input the secret image, transformed by the chromosomes that were built in the first generation, and then embed it into the cover image as well as calculate the value of MSE. In the second step, it has to execute a loop according the set numbers of generation. If it is satisfied with the set value of MSE while executing, it will then break from the loop and output the suitable chromosome. If it is not satisfied with the value of MSE, it will choose a better one from among these chromosomes. Furthermore, it will result a new generation of chromosomes through crossover and mutation, and then transfer them to the next generation for evaluation. When it is satisfied with the number of times of generation or the set value of MSE, it will stop reproducing and output the best transformed block.

The procedure of crossover can be handled by one point or two points. At first, transform the two dimension *R* transform into a one dimension block. The handling process is as follows:

$$\begin{bmatrix} a_0 & a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 & a_7 \\ a_8 & a_9 & a_{10} & a_{11} \\ a_{12} & a_{13} & a_{14} & a_{15} \end{bmatrix} \quad \begin{bmatrix} a_0 & a_1 & \dots & a_{15} \end{bmatrix}.$$

Then, take this one dimension block as a chromosome. If there were two chromosomes, G_1 and G_2 which are shown as Figure 1(a), the gene of G_1 is

shown by boldface while the gene of G_2 is in regular font. After a one point crossover, the result is shown as Figure 1(b). It results in new chromosomes G_1' and G_2'.

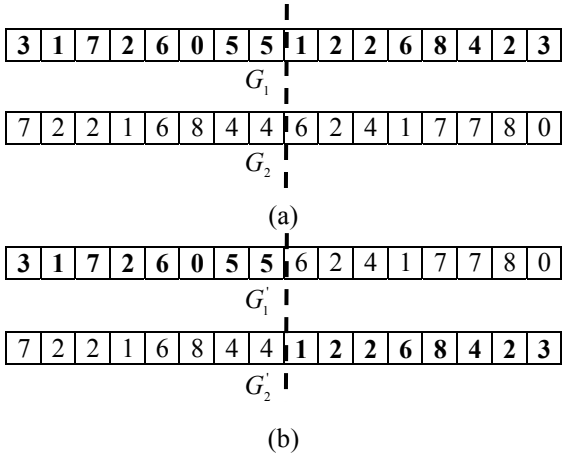


Figure 1. A simple example: after crossover, (a) G_1 and G_2 (b) become G_1' and G_2'.

Mutation mainly provides each gene with a probability. If the gene falls into the range of probability, then it will reproduce a new element at random. Figure 2 is the mutation results of G_1'. Boldface numbers are the reproduced genes which fall into a given probability.

Figure 2. The result is mutation of G_1'. Boldface numbers are the reproduced genes which fall into a given probability.

3.3. Discussion of set way of matrix

There are various kinds of design methods of transform block. One manner uses a $n \times n$ block R to handle a whole image. The second manner segments the image into four sub-images and individually trains each sub-image by a $(n/2)\times(n/2)$ block. Then, it will get the best four $(n/2)\times(n/2)$ blocks, with the total size equal to a $n \times n$ block. The third manner segments the image into 16 sub-images, and individually trains each sub-image by a $(n/4)\times(n/4)$ block. Then, it will get the best 16 $(n/4)\times(n/4)$ blocks. Here the total size is equal to a $n \times n$ block. The experimental result will show that the third manner spends much less time while obtaining the highest quality.

4. Experimental results

This section mainly discusses the quality of a hidden image by the method that is proposed in this paper, by experiment. The program is written in C++ Builder, and executed under an AMD Athlon 2600+ and 1G RAM WinXP system. The size of the experimental stego-image is 512×512 pixels gray-level, as shown as Figure 3(a) and 3(b). The secret image is 256×512 pixels gray-level images, shown as Figure 3(b), (c) and (d). Therefore, the data hiding rate is 4 bits of each pixel. Then, an 8×8 block R, four 4×4 block R, and 16 16×16 block R are taken as the transform block, and the secret image is disarranged by three methods: a whole image, 4 sub-images, and 16 sub-images. As a result, these 256 bits are secret keys.

First, we transform a secret image by one 8×8 block R. In the process of genetic algorithm, it will produce forty R, and treat them like chromosomes. If these forty R were handled by set 0.5 crossover rates and 0.3 mutation rates, it would reproduce another new forty chromosomes. Then, we pick out 40 chromosomes whose value of MSE are much lower among the 80 chromosomes, and take these 40 chromosomes as the next generation chromosomes. Therefore, the selection rate is 0.5. The last generation, which is tested 1000 times, will take the lowest value of MSE which is produced by R as transform block. After 1000 generations, we will know that the MSE still can be improved when it is transformed at the 751st generation. Therefore, the 1000 cycles of generation take 33 minutes and 49 seconds, and the value of MSE is 21.1132.

Then, the stego-image is segmented into four sub-images and each sub-image is transformed individually by a 4×4 block. In the process of genetic algorithm, it will produce forty R for each sub-image, take them as chromosomes and handled by set 0.5 crossover rates, 0.3 mutation rates and 0.5 selection rates. After 500 generations, it will take the lowest value of MSE which is produced by a 4×4 block as transform block. There will be four 4×4 blocks in the whole image, and the combined size will equal that of a 8×8 block. After 100 generations, we will know that the improved range of generation deceases. It is finished after 500 times generation, takes 17 minutes and 32 seconds, and the value of MSE is 20.7342.

Finally, the stego-image is segmented into sixteen sub-images, and each sub-image is individually transformed by one 2×2 block. In the process of genetic algorithm, it will be handled by set 0.5 crossover rates, 0.3 mutation rates and 0.5 selection rates. After 100 generations, it will take the lowest value of MSE which is produced by the 2×2 block as

the transform block. There will be sixteen 2×2 blocks in the whole image, and the combined size will be equal to that of the 8×8 block. After 20 generations, we will know that the improved range of generation greatly decreases. It is finished after 100 times generation, takes four minutes and five seconds, and the value of MSE is 20.3695.

By genetic algorithm, it can set the initial transform block at random from the enormous transform block, then use crossover, mutation and selection to produce new generation, and explore if there is a better transform block. The experiment took lots of test time, but in reality, it can set fewer numbers of generations to determine the better transform block.

Table 1 shows the results of the proposed methods: 8×8, four 4×4, and sixteen 2×2 of the cover image Lena and three secret images. It also lists the contrast result of simple LSB substitution, genetic algorithm [12], dynamic programming [1] and optimal pixel adjustment process [2, 4]. Table 2 shows the contrast between the proposed method and other methods of the cover image pepper and other three secret images. We understand that the proposed method can obtain the least value of MSE.

5. Conclusions

While transferring data over the Internet, we face many unknowns. To transfer data securely, a safety system is needed. This paper proposes an effective, simple image-hiding technique to improve the quality of stego-images. It can raise the quality of the stego-image, and hackers may ignore the stego-image with its embedded secret image.

The experimental result shows that the stego-image and cover image can not be distinguished. The hiding capacity can reach four bits in a pixel, and the mean-square error of the stego-image is much lower than for the images which were produced by previous methods. In the future, we hope that we can find an even better technique to hide more data in a cover image. Furthermore, we also have to make efforts in cover image modification. Less modifying or even no modifying should be the focus of future research.

6. Rerferences

[1] C. C. Chang, J. Y. Hsiao, and C. S. Chan, "Finding optimal least-significant-bit substitute in image hiding by dynamic programming strategy", *Pattern Recognition,* vol. 36, no. 7, 2003, pp. 1583-1595.

[2] C. C. Thien and J. C. Lin, "A simple and high-hiding capacity method for hiding digit-by-digit data in images based on modulus function", *Pattern Recognition*, vol. 36, no. 12, 2003, pp. 2875-2881.

[3] C. K. Chan and L.M. Cheng, "Improved hiding data in images by optimal moderately significant-bit replacement", *IEE Electronics Letters,* vol. 37, no. 16, 2001, pp. 1017-1018.

[4] C. K. Chan and L. M. Cheng, "Hiding data in image by simple LSB substitution", *Pattern Recognition,* vol. 37, no. 3, 2004, pp. 469-474.

[5] E. Adelson, "Digital signal encoding and decoding apparatus", *US Patent,* no. 4939515, 1990.

[6] J. Wang and L. Ji, "A region and data hiding based error concealment scheme for images", *IEEE Transactions on Consumer Electronics*, vol. 47, no. 2, 2001, pp. 257-262.

[7] K. L. Chung, C. H. Shen, and L. C. Chang, "A novel SVD- and VQ-based image hiding scheme", *Pattern Recognition Letters,* vol. 22, no. 9, 2001, pp. 1051-1058.

[8] L. F. Turner, "Digital data security system", *Patent IPN WO 89/08915,* 1989.

[9] L. M. Marvel, C. G. Boncelet, and C. T. Retter, "Spread spectrum image steganography", *IEEE Transactions on Image Processing*, vol. 8, no. 8, 1999, pp. 1075-1083.

[10] R. Z. Wang, C. F. Lin, and J. C. Lin, "Hiding data in images by optimal moderately significant-bit replacement", *IEE Electronics Letters,* vol. 36, no. 25, 2000, pp. 2069-2070.

[11]N. G. Bourbakis and C. Alexopoulos, "Picture data encryption using SCAN patterns", *Pattern Recognition*, vol. 25, no. 6, 1992, pp. 567--581.

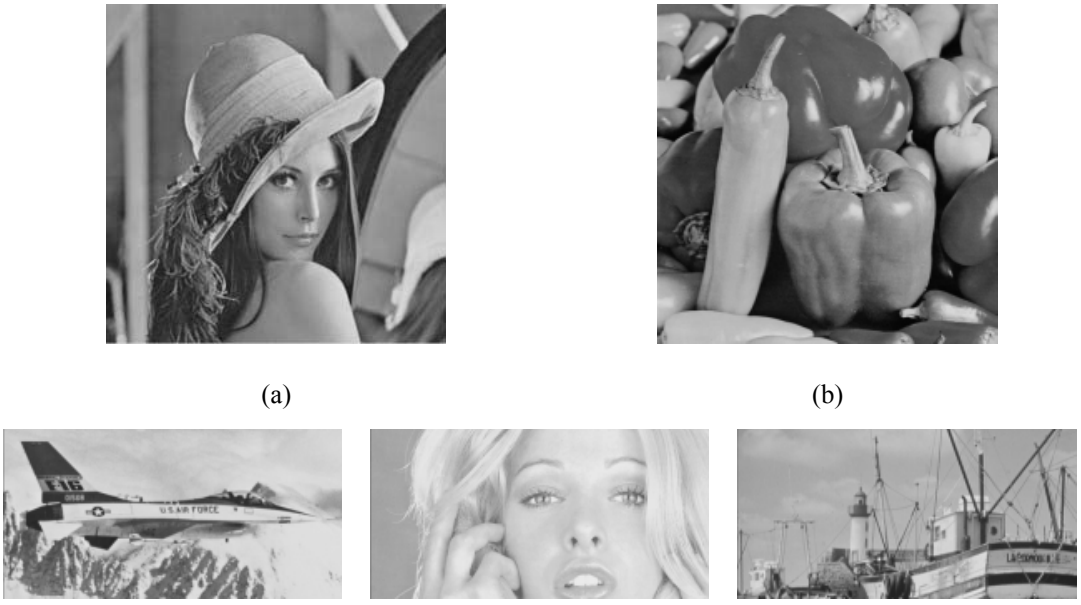
[12] R. Z. Wang, C. F. Lin, and J. C. Lin, "Image hiding by optimal LSB substitution and genetic algorithm", *Pattern Recognition*, vol. 34, no. 3, 2001, pp. 671-683.

[13] T. J. Chuang and J. C. Lin, "New approach to image encryption", *Journal of Electronic Imaging,* vol. 7, no. 2, April 1998, pp. 350-356.

[14] T. S. Chen, C. C. Chang, and M. S. Hwang, "A virtual image cryptosystem based upon vector quantization", *IEEE Transactions on Image Processing*, vol. 7, no. 10, 1998, pp. 1485-1488.

[15] W. Bender, D. Gruhl, N. Morimoto, and A. Lu, "Techniques for data hiding", *IBM Systems J.* vol. 35, no. 3-4, 1996, pp. 313-316.

[16] W. Bender, F.J. Paiz, W. Butera, S. Pogreb, D. Gruhl, and R. Hwang, "Applications for data hiding", *IBM Systems J.* vol. 39, no. 3-4, 2000, pp. 547-568.



(a) (b)

(c) (d) (e)

Figure 3. Testing images (a) and (b) are cover images: Lena and pepper. (c)(d)(e) are secret images Jet, Tiffany, and Boat.

Table 1. The MSE between the cover image Lena and the stego-image

Methods	Jet	Tiff	Boat
LSB	40.7825	42.7265	36.5536
Genetic Algorithm [12]	34.1974	34.7052	36.1665
Dynamic programming strategy [1]	33.3233	33.2183	34.8507
OPAP [2, 4]	21.1199	21.1874	21.4539
The proposed method by $R_{8\times8}$	21.1132	21.1234	21.1242
The proposed method by $R_{4\times4\times4}$	20.7342	20.8190	20.7647
The proposed method by $R_{16\times2\times2}$	20.3695	20.4328	20.4894

Table 2. The MSE between the cover image Pepper and the stego-image

Methods	Jet	Tiff	Boat
LSB	40.7747	42.7432	36.6526
Genetic Algorithm [12]	34.7647	34.5167	36.5978
Dynamic programming strategy [1]	33.2655	33.0889	34.7991
OPAP [2, 4]	21.2451	21.3875	21.4157
The proposed method by $R_{8\times8}$	21.2382	21.1550	21.2583
The proposed method by $R_{4\times4\times4}$	20.8734	20.8167	20.9062
The proposed method by $R_{16\times2\times2}$	20.4598	20.5622	20.6254