Available online at www.sciencedirect.com
SCIENCE DIRECT®
Pattern Recognition 38 (2005) 2466–2471
PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY
www.elsevier.com/locate/patcog

Fault-tolerant and progressive transmission of images

Shang-Kuan Chen, Ja-Chen Lin*

Department of Computer and Information Science, National Chiao Tung University, Hsinchu 30050, Taiwan, ROC

Received 1 July 2004; received in revised form 4 April 2005; accepted 4 April 2005

Abstract

A fault-tolerant progressive image transmission method is proposed. The advantages include the following: (1) Unlike most progressive methods, the image is divided into n parts with equal importance to avoid worrying about which part is lost or transmitted first. (2) If the image is a secret image, then the transmission can use n distinct channels (one shared result per channel), and intercepting up to $r_1 - 1$ channels by the enemy ($r_1 \leqslant \cdots \leqslant r_k \leqslant n$ are all pre-set constants) will not reveal any secret. Meanwhile, the disconnection up to $n - r_k$ channels will not affect the lossless recovery of the secret image.
© 2005 Pattern Recognition Society. Published by Elsevier Ltd. All rights reserved.

Keywords: Progressive; Image sharing; Image hiding; Quality-control

1. Introduction

Traditionally, when an image is transmitted in a progressive way, the information contained in the image is partitioned into several parts and the most significant part is transmitted first while the least significant part is transmitted last. If the most significant part is damaged or lost, the recovered image will be significantly degraded. Therefore, a "fault-tolerant" progressive image transmission system is more useful. In most researches [1–3] about progressive image transmission, the receiver can immediately stop transmitting an image if the received rough version of the image shows that the image being received (for example, a jet airplane) is not the one the receiver desired (for example, Lena). However, none of these methods is fault-tolerant. The goal of the current paper is therefore a fault-tolerant progressive image transmission method. This goal is achieved through the careful use of an image sharing technique [4] derived from Ref. [5,6].

Blakley [5] and Shamir [6] first proposed the idea of secret sharing, known as the (r, n) threshold scheme. In their (r, n) threshold scheme, the system consists of a dealer and n participants; the dealer distributes a secret number into n shares and each participant holds one share. Later, if r shares are received, then the secret number can be revealed. If less than r shares are received, then no information about the secret number can be revealed. This secret sharing scheme is fault-tolerant in the sense that $n-r$ shares can be lost during the reconstruction (because only r shares are needed).

Thien and Lin [4] applied the idea in Ref. [6] to share a secret image and generated n shadow images (the detail is described in Section 2). The size of each shadow image was only $1/r$ of the original secret image. The secret image can be recovered if r of the n shadow images were received. Due to the smaller size $(1/r)$ of the shadow images, the total transmission time needed to recover an image (by receiving r shadow images) would not increase.

In the scheme of Thien and Lin [4], if less than r shadow images were received, people could not get any information about the secret image. In the current paper, a progressive version of image sharing is proposed. The user can set several thresholds, namely, the k thresholds: $r_1 \leqslant r_2 \cdots \leqslant r_k = r$. If less than r_1 shared results are received, nothing can be

* Corresponding author. Fax: +886 3 572 1490.
E-mail address: gis88811@cis.nctu.edu.tw (J.-C. Lin).

0031-3203/$30.00 © 2005 Pattern Recognition Society. Published by Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2005.04.002

revealed. However, if r_1 shared results are received, a rough version of the original image can be revealed; then, for each $s > 1$, if r_s shared results are received and $r_s > r_s - 1$, the quality of the recovered image is better than the one using r_{s-1} shared results. Finally, if r_k shared results are received, the image can be recovered without any loss. Therefore, the absence of $n - r_k$ shared results cannot affect the lossless recovery.

Notably, since the content of each shared result of an original (secret) image always looks noisy, an attack from hackers is quite likely. Therefore, a data-hiding method [7] is utilized to hide the shared results in some host images to form stego images, which look ordinary instead of being noisy, to avoid attracting the hackers' attention [8–10].

Section 2 introduces the secret image sharing method proposed by Thien and Lin [4]. Section 3 describes the details of our scheme. Section 4 discusses the decoding phase. Section 5 provides the experimental result. Finally, Section 6 discusses the design to control quality.

2. Thien and Lin's secret image sharing method

Thien and Lin [4] proposed a method to share a secret image based on Shamir's [6] polynomial threshold scheme with a prime number p. The image is divided into several non-overlapping sectors, and each sector has r pixels. For each sector j, the r coefficients $a_0, a_1, \ldots, a_{r-1}$ of the corresponding polynomial

$$q_j(x) = a_0 + a_1 \times x + \cdots + a_{r-1} \times x^{r-1} (\mathrm{mod}\ p)$$

are assigned as the r gray values of the r pixels in the sector. The wth shadow image is the collection $\{q_j(w) | j = 1, 2, \ldots, \textit{original image size}/r\}$. Since each sector j, which has r pixels, contributes only one pixel $q_j(w)$ to the wth shadow image, the size of the wth shadow image is only $1/r$ of the secret image. This property holds for every $w \in \{1, 2, 3, \ldots, p-1\}$.

3. Proposed method

The scheme is illustrated in Fig. 1. First, a bit-plane scanning method rearranges the gray value information of the original image. Then, the rearranged data are shared. Finally, the shared results are hidden in some host images.

Step 1. Bit plane scanning to rearrange data: First, let the k threshold values $r_1, r_2, \ldots, r_k$ be assigned so that $r_1 \leqslant r_2 \leqslant \cdots \leqslant r_k = r$. The k thresholds denote the distinct number of shared results needed to recover the image with distinct quality levels. For example, a quite rough image can be recovered when r_1 shared results are received. Then, the image quality gradually improves when more shared results are received. Finally, the image can be recovered completely when r_k shared results are received.

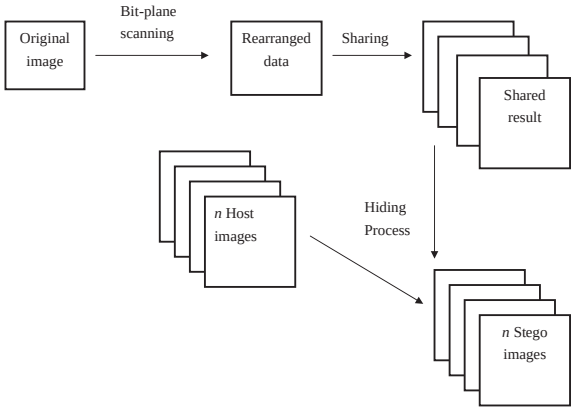

Fig. 1. The flow chart of our method (the encoding phase).

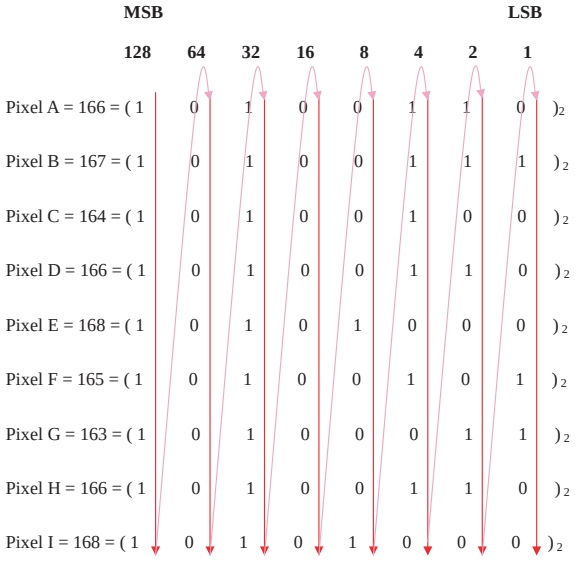

Fig. 2. The sequence of scanning the 8 bit planes. Note that $RSUM = r_1 + r_2 + \cdots + r_k$ pixels are scanned for each sector.

Below, we discuss how to scan the image and generate the rearranged data. To begin the process, the original image is divided into several non-overlapping sectors, and the sectors are processed one by one. Each sector always has *RSUM* pixels ($RSUM = r_1 + r_2 + \cdots + r_k$). Since each pixel has 8 bits in the gray-level image, each sector has $8(r_1 + r_2 + \cdots + r_k)$ bits. We then rearrange these bits to get another $r_1 + r_2 + \cdots + r_k$ values; each of them is still an 8-bit number ranging from 0 to 255. For example, if $k = 3$ and 2, 3, 4 as the three threshold values, then each sector contains $2 + 3 + 4 = 9$ pixels. Without the loss of generality, let the 9 pixels of the sector being discussed have the gray values 166, 167, 164, 166, 168, 165, 163, 166, and 168, respectively. By the rearranging process, shown in Fig. 2, the 9 transformed

values of the sector are 255, 128, 63, 224, 0, 143, 171, 76, and 140. The details are shown below. The 9 input pixels $\{A, B, C, D, E, F, G, H, I\}$ are

$$A = 166 = (1010\ 0110)_2 = (A_1A_2A_3A_4A_5A_6A_7A_8)_2,$$
$$B = 167 = (1010\ 0111)_2 = (B_1B_2B_3B_4B_5B_6B_7B_8)_2,$$
$$C = 164 = (1010\ 0100)_2 = (C_1C_2C_3C_4C_5C_6C_7C_8)_2,$$
$$D = 166 = (1010\ 0110)_2 = (D_1D_2D_3D_4D_5D_6D_7D_8)_2,$$
$$E = 168 = (1010\ 1000)_2 = (E_1E_2E_3E_4E_5E_6E_7E_8)_2,$$
$$F = 165 = (1010\ 0101)_2 = (F_1F_2F_3F_4F_5F_6F_7F_8)_2,$$
$$G = 163 = (1010\ 0011)_2 = (G_1G_2G_3G_4G_5G_6G_7G_8)_2,$$
$$H = 166 = (1010\ 0110)_2 = (H_1H_2H_3H_4H_5H_6H_7H_8)_2,$$
$$I = 168 = (1010\ 1000)_2 = (I_1I_2I_3I_4I_5I_6I_7I_8)_2.$$

Now we scan these $8 \times 9 = 72$ bits according to the order specified in Fig. 2; i.e. the 9 most significant bits (MSB) first $(A_1, B_1, \ldots, I_1)$; then the 9 second-most significant bits $(A_2, B_2, \ldots, I_2)$; then the 9 third-most significant bits $(A_3, B_3, \ldots, I_3)$, etc. Therefore we obtain a rearranged 72-bit data set $(A_1, B_1, \ldots, I_1, A_2, B_2, \ldots, I_2, A_3, B_3, \ldots, I_3, \ldots, A_8, B_8, \ldots, I_8)$. If we read these 72 bits (according to the above order), and explain them as 9 numbers (each one is an 8-bit number), then we can obtain the rearranged result for this sector, namely, the following 9 values:

$$(A_1B_1C_1D_1E_1F_1G_1H_1)_2 = (1111\ 1111)_2 = 255,$$
$$(I_1A_2B_2C_2D_2E_2F_2G_2)_2 = (1000\ 0000)_2 = 128,$$
$$(H_2I_2A_3B_3C_3D_4E_4F_3)_2 = (0011\ 1111)_2 = 63,$$
$$(G_3H_3I_3A_4B_4C_4D_4E_4)_2 = (1110\ 0000)_2 = 224,$$
$$(F_4G_4H_4I_4A_5B_5C_5D_5)_2 = (0000\ 0000)_2 = 0,$$
$$(E_5F_5G_5H_5I_5A_6B_6C_6)_2 = (1000\ 1111)_2 = 143,$$
$$(D_6E_6F_6G_6H_6I_6A_7B_7)_2 = (1010\ 1011)_2 = 171,$$
$$(C_7D_7E_7F_7G_7H_7I_7A_8)_2 = (0100\ 1100)_2 = 76,$$
$$(B_8C_8D_8E_8F_8G_8H_8I_8)_2 = (1000\ 1100)_2 = 140.$$

Step 2. Sharing: Recall that in our example above, we assumed that $r_1 = 2, r_2 = 3, r_3 = 4$, and therefore each sector has 9 pixels. In Step 1, we already transformed the 9 pixels $\{166, 167, 164, 166, 168, 165, 163, 166, 168\}$ of the current sector (say, Sector j) into 9 new values $\{255, 128, 63, 224, 0, 143, 171, 76, 140\}$. Now, the first $r_1 = 2$ transformed values $\{255, 128\}$ forms the first polynomial

$$f_j^{(1)}(x) = (255 + 128x) \bmod 257 \tag{1}$$

for the current sector. Then, the next $r_2 = 3$ transformed values $\{63, 224, 0\}$ forms the second polynomial

$$f_j^{(2)}(x) = (63 + 224x + 0x^2) \bmod 257 \tag{2}$$

for the current sector. Finally, the final $r_k{=}r_3{=}4$ transformed values $\{143, 171, 76, 140\}$ forms the final polynomial

$$f_j^3(x) = (143 + 171x + 76x^2 + 140x^3) \bmod 257 \tag{3}$$

for the current sector. For each participant $w \in \{1, 2, 3, \ldots, p-1\}$, the collection

$$\left\{ f_j^{(i)}(w) | j = 1, 2, \ldots, \frac{\textit{original image size}}{\textit{RSUM}}, \textit{ and } i = 1, 2, \ldots, k \right\} \tag{4}$$

is called the wth shared result $SR(w)$. Notably, since each sector has $RSUM = r_1 + r_2 + \cdots + r_k$ pixels, the total number of sectors that we have is $j_{MAX} = \textit{original image size}/RSUM$. In addition, each shared result $SR(w)$ receives only k numbers $f_j^{(1)}(w) \sim f_j^{(k)}(w)$ generated from sector j, which is an $RSUM$-pixel region of the original image. Therefore, the size of each shared result is $k/RSUM = k/(r_1 + r_2 + \cdots + r_k)$ of the original image. In the above example, this ratio is $\frac{3}{2+3+4} = \frac{1}{3}$.

(*Optional*) *Step* 3: *Hiding the shared results in some host images*: The contents of the shared results look noisy. Therefore, in the special case where the original image is an important secret image, the shared results should be hidden in some host images to form stego images which look ordinary (non-noisy) to avoid attracting an attacker's attention. The hiding algorithm that we use here is one we developed earlier (similar to the one used in Section 2.3 of Ref. [7]). Note that the size of each shared result is about $k/(r_1 + r_2 + \cdots + r_k)$ of the original image; therefore, the size of each stego image is about $2k/(r_1 + r_2 + \cdots + r_k)$ of the original image.

4. Decoding phase

The recovery of the original image includes three steps: extracting the shared results from stego images; recovering the rearranged values from the shared results; and restoring the pixel values from the rearranged values.

Step 1: *Extracting the shared results from stego images*: This step needs only simple operations such as division, addition, and multiplication. The procedure is similar to the one used in Section 2.3 of Ref. [7], and hence omitted.

Step 2: *Recovering the rearranged values from the shared results*: To illustrate this step, let us inspect the example given in Eqs. (1)–(3) where $k{=}3$ threshold values were used $(r_1 = 2, r_2 = 3, r_3 = 4)$. According to Eq. (4), the shared result held by participant w is

$$SR(w) = \bigcup_j \{f_j^{(1)}(w), f_j^{(2)}(w), f_j^{(3)}(w)\}, \tag{5}$$

where j ranges through all possible sectors contained in the original image. Now, in the decoding phase, assume that we receive two shared results, say, $SR(1)$ and $SR(4)$.

Then, since

$$SR(1) = \bigcup_j \left\{ f_j^{(1)}(1),\, f_j^{(2)}(1),\, f_j^{(3)}(1) \right\}$$

and

$$SR(4) = \bigcup_j \left\{ f_j^{(1)}(4),\, f_j^{(2)}(4),\, f_j^{(3)}(4) \right\},$$

we can reveal the values $f_j^{(1)}(1)$ and $f_j^{(1)}(4)$ for each sector j. Therefore, the coefficients 255 and 128 of the polynomial $f_j^{(1)}(x)$ defined in Eq. (1) can be determined (through the two points $(1, f_j^{(1)}(1))$ and $(4, f_j^{(1)}(4))$, a unique line is determined; and the equation of this unique interpolation polynomial of degree 1 can be found by using Lagrange's interpolation [see Section 3.2 of Ref. [4]]).

In the process of progressive transmission, assume that we receive one more shared result, say, besides $SR(1)$ and $SR(4)$, we also receive $SR(5)$. Then, since

$$SR(5) = \bigcup_j \{ f_j^{(1)}(5),\, f_j^{(2)}(5),\, f_j^{(3)}(5) \},$$

we can extract the values $f_j^{(2)}(1)$, $f_j^{(2)}(4)$, and $f_j^2(5)$. Again, by using Lagrange's interpolation, we can find the unique interpolation polynomial through the three points $(1, f_j^{(2)}(1))$, $(4, f_j^{(2)}(4))$, and $(5, f_j^{(2)}(5))$. Therefore, the 3 coefficients $\{63, 224, 0\}$ of $f_j^{(2)}(x)$ defined in (2) can be obtained. Therefore, $\{255, 128; 63, 224, 0\}$ are all known when we receive three shared results $SR(1)$, $SR(4)$, and $SR(5)$.

An analogous argument shows that we can know all 9 coefficients in Eqs. (1)–(3) if we receive 4 shared results, say, $SR(1)$, $SR(4)$, $SR(5)$, and $SR(8)$.

Step 3: *Restoring the pixel values from the rearranged values*: After obtaining the rearranged values in the previous step, the values can be transformed back to restore the pixels of the original image. For example, if $\{255, 128\}$ and $\{63, 224, 0\}$ are recovered in the previous step (assuming that three shared results are received), then $255 = (1111\ 1111)_2, 128 = (1000\ 0000)_2, 63 = (0011\ 1111)_2, 224 = (1110\ 0000)_2$, and $0 = (0000\ 0000)_2$ together form a sequence of 40 bits, i.e. (1111 1111 1000 0000 0011 1111 1110 0000 0000 0000). If we restore these 40 bits according to the scan order listed in Fig. 2, we can restore at least $\lfloor 40/9 \rfloor = 4$ of the most significant bits of the 9 pixels A–I. In fact, $40 - 9 \times 4 = 4$ implies that the first four pixels (A–D) can each recover one more bit. Therefore the $\lceil 40/9 \rceil = 5$ of the most significant bits of the pixels A, B, C, and D are revealed as (10100), (10100), (10100), and (10100), respectively; while the $\lfloor 40/9 \rfloor = 4$ of the most significant bits of the pixels E, F, G, H, and I are revealed as (1010), (1010), (1010), (1010), (1010), respectively (see Fig. 2).

5. Experiment

The experimental result is shown in Figs. 3–5 and Table 1. The input is the image Lena shown in Fig. 3, which is shared by our progressive scheme. In the experiment, we use $k = 4$ thresholds, which are $(r_1 = 2) < (r_2 = 3) < (r_3 = 4) < (r_4 = r_k = 5)$. We generate, say, $n = 6$ shares, and then the 6 shared results are hidden in six host images to generate 6 stego images. Fig. 4 shows the stego images. Their *PSNR*s range from 34.20 to 34.49. Notably, the size of each host image and each stego image in this experiment is $388 \times 388 = 2 \times [512 \times 512 \times \frac{4}{2+3+4+5}]$, for $512 \times 512 \times \frac{4}{2+3+4+5}$ (the original image size multiplied by $k/(r_1 + r_2 + \cdots + r_k)$) is the size of each shared result. The factor 2 is due to the fact that the stego image size is two times greater than the data (the shared result) hidden inside.

Fig. 5 shows the images recovered from various numbers of the stego images. Fig. 5(a) shows the recovered image when "any" two of the stego images in Fig. 4 are available. The recovered image has a poor quality because the *PSNR* is only 14.57 db. Fig. 5(b) shows the recovered image when any of the three stego images are available; the recovered image has a better quality (29.28 db). Fig. 5(c) shows the recovered image when any four of these stego images are available; the recovered image has an even better quality (48.46 db). Fig. 4 shows the recovered image when any five of the six stego images are available; the recovered image is lossless.

Fig. 3. The 512×512 original image.

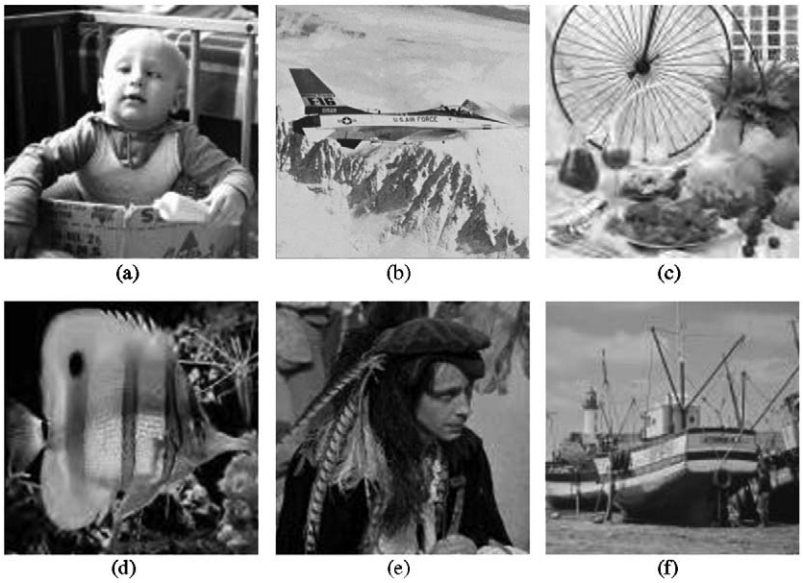

Fig. 4. The six 388 × 388 stego images (the *PSNR*s range from 34.20 to 34.49).

Fig. 5. The recovered images revealed from various numbers of stego images: (a) from any 2 stego images ($PSNR = 14.57$); (b) from any 3 stego images ($PSNR = 29.28$); (c) from any 4 stego images ($PSNR = 48.46$); (d) from any 5 stego images (lossless).

Table 1
*PSNR*s for the three kinds of thresholds-setting that are all composed of 3, 4, and 5 shared results

Number of stego images	Thresholds-setting		
	3, 4, 5 (12 pixels/sector)	3, 4, 5, 5, 5 (22 pixels/sector)	3, 3, 3, 4, 5 (18 pixel/sector)
3	20.01	13.15	31.11
4	42.70	20.00	46.38
5	Lossless	Lossless	Lossless

6. Quality control design

It is possible to control the quality of the image recovered from a small number of shared results. In some commercial applications (for example, a cable system whose image quality depends on the amount of money paid by the viewer), if a viewer has, say, only 3 shared results, the copyright owner may require that the unveiled quality should be poor. In some other applications (for example, the hot line between two police stations), the requirement might be that only 3 shared results can still provide good-quality recovery. To control quality, we may repeat some threshold values in the design. For example, consider a design where the threshold values are $\{3, 4, 5\}$. If we use only three threshold values $\{r_1 = 3, r_2 = 4, r_3 = 5\}$, we get the results shown in the middle column of Table 1; i.e. 20.01 and 42.7 db when we receive 3 and 4 shared results, respectively. (There are $3 + 4 + 5 = 12$ pixels per sector in this case; for we construct 3 polynomials for each sector, and these 3 polynomials have 3, 4, and 5 coefficients, respectively.) In some business applications, their systems may want to reveal only poor quality images when customers purchase less than 5 shared results. The thresholds-setting

$\{r_1 = 3, r_2 = 4, r_3 = 5, r_4 = 5, r_5 = 5\}$ is then a solution for this, because only about $\frac{3+4}{3+4+5+5+5} \times 8 = \frac{7}{22} \times 8 \approx 2.5$ of the 8 bits are revealed for each pixel when 4 shared results are received. (There are $3 + 4 + 5 + 5 + 5 = 22$ pixels per sector in this thresholds-setting case; for we build 5 polynomials whose numbers of coefficients are 3, 4, 5, 5, and 5, respectively. Receiving 4 shared results can recover only the first two polynomials [because $3 \leqslant 4$ and $4 \leqslant 4$], and therefore recover only $3 + 4 = 7$ of the 22 coefficients of the polynomials.) As shown in Table 1, the image quality is quite poor (13.15 or 20.00 db) when the image is recovered using 3 or 4 shared results. Finally, when fast revealing of the good-quality image is required, e.g. between police stations, the thresholds-setting $\{r_1 = 3, r_2 = 3, r_3 = 3, r_4 = 4, r_5 = 5\}$ can achieve the goal, because $\frac{3+3+3}{3+3+3+4+5} \times 8 = \frac{1}{2} \times 8 = 4$ of the 8 bits are revealed for each pixel when only 3 shared results are received. (There are $3+3+3+4+5=18$ pixels per sector in this thresholds-setting case; for we build 5 polynomials whose numbers of coefficients are 3, 3, 3, 4, and 5, respectively. Receiving 3 shared results can recover only the first three polynomials, and therefore recover $3+3+3=9$ of the 18 coefficients). As shown in Table 1, the image quality is acceptable (31.11 db) when the image is recovered by 3 shared results. Moreover, the image quality is good (46.38 db) when the image is recovered by 4 shared results.

7. Conclusion

There are several characteristics in the proposed progressive image sharing scheme, including (1) the scheme is fault tolerant (allowing $n-r$ stego images to be lost or damaged); (2) the shared results are equally important, so there is no need to worry about which part is lost or transmitted first; (3) the scheme is secure (less than r_1 shared results cannot reveal any information about the image); and (4) quality-control design is possible (as explained in Section 6).

Acknowledgements

The authors would like to thank the reviewer for valuable comments. This work was supported by the National Science Council, Republic of China, under grant NSC92-2213-E-009-032.

References

[1] K.L. Chung, S.Y. Tseng, New progressive image transmission based on quadtree and sharing approach with resolution control, Pattern Recognition Lett. 22 (2001) 1545–1555.
[2] A. Benazza-Benyahia, J.-C. Pesquet, A unifying framework for lossless and progressive image coding, Pattern Recognition 35 (2002) 627–638.
[3] C.C. Chang, J.J. Jau, T.S. Chen, A fast reconstruction method for transmitting image progressively, IEEE Trans. Consumer Electron. 44 (4) (1998) 1225–1233.
[4] C.C. Thien, J.C. Lin, Secret image sharing, Comput. Graphics 26 (2002) 765–770.
[5] G.R. Blakley, Safeguarding cryptographic keys, Proceedings AFIPS 1979 National Computer Conference, vol. 48, New York, USA, June 1979, pp. 313–317.
[6] A. Shamir, How to share a secret, Commun. ACM 22 (11) (1979) 612–613.
[7] Y.S. Wu, C.C. Thien, J.C. Lin, Sharing and hiding secret images with size constraint, Pattern Recognition 37 (7) (2004) 1377–1385.
[8] N. Bourbakis, A. Dollas, SCAN-based compression-encryption-hiding for video on demand, IEEE Multimedia Mag. 10 (2003) 79–87.
[9] F.A.P. Petitcolas, R.J. Anderson, M.G. Kuhn, Information hiding—a survey, Proc. IEEE 87 (7) (1999) 1062–1078.
[10] W. Bender, W. Butera, D. Gruhl, R. Hwang, F.J. Paiz, S. Pogreb, Applications for data hiding, IBM System J. 39 (3–4) (2000) 547–568.

About the Author—SHANG-KUAN CHEN was born in 1972 in Taiwan, Republic of China. He received his B.S. degree in applied mathematics in 1994 from Fu Jen Catholic University, Taiwan. In 1998, he received his M.S. degree in applied mathematics from National Chiao Tung University, Taiwan. Since 1999, he has been studying for a Ph.D. degree in the Computer and Information Science Department of National Chiao Tung University. His research interests include visual cryptography, data hiding, and interconnecting network.

About the Author—JA-CHEN LIN was born in 1955 in Taiwan, Republic of China. He received his B.S. degree in computer science in 1977 and M.S. degree in Applied Mathematics in 1979, both from National Chiao Tung University, Taiwan. In 1988, he received his Ph.D. degree in mathematics from Purdue University, USA. During 1981–1982, he was an instructor at National Chiao Tung University. From 1984 to 1988, he was a graduate instructor at Purdue University. He joined the Department of Computer and Information Science at National Chiao Tung University in August 1988, and became a professor there. His research interests include pattern recognition and image processing. Dr. Lin is a member of the Phi-Tau-Phi Scholastic Honor Society.