Chapter 4
A New Digital Pointillistic Image Creation Method for Information Hiding by A Palette Color Coding Technique

4.1 Overview of Proposed Method

The idea of the proposed method comes from the paintings of Georges Seurat who was a French leading painter in the Neo-impressionist movement of the late 19th century. He placed pure colors side by side by applying tiny strokes, dots, and dashes to the surface of the canvas. When seen from a distance, the tiny strokes blend together in the eye and are perceived as secondary colors. This technique has become known as Pointillism. In this chapter, we try to imitate the style of the pointillistic painter via a computer, and we name this kind of image *digital pointillistic image*.

In order to achieve the purpose of information hiding, we try to embed data into certain features of the digital pointillistic image. We propose a new pointillistic image creation method and a method for information hiding in digital pointillistic images by a palette color coding technique. By this method, we can embed secret data in a digital pointillistic image during the image creation process.

4.2 Proposed Digital Pointillistic Image Creation Method

4.2.1 Idea of Proposed Method

In a sense, the gimmick of creating pointillism is similar to the process of creating a digital image via a computer. A computer draws a digital image using colored pixels. However, because the pixels are too small, computer monitors will not give us a picture like one that a neo-impressionist paints.

A flowchart of the proposed digital pointillistic image creation process is shown in Figure 4.1. In order to imitate the pointillistic style while creating a digital image, we distort an input image and enlarge the color pixels of it to become "dotlike." Besides, we also overlap the enlarged color dots of the created digital pointillistic image. In principle, pixels are drawn onto a digital image serially, starting from the top-left pixel, and moving horizontally on a row-by-row basis. However, while performing the proposed digital pointillistic image creation process described in Section 4.2.3, we should disarrange the drawing order of the *enlarged-and-overlapped-color-dots* (denoted as *EOCD*s) by applying a random array creation process discussed in Section 4.2.3 first. Besides, if we want to create the digital pointillistic image to have a watercolor look, we can apply the digital watercolor image creation process proposed in Section 4.2.2 to deal with the original digital image before performing the digital pointillistic image creation process.

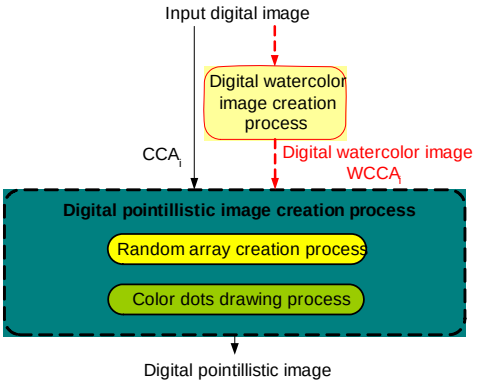


Figure 4.1 A flowchart of the digital pointillistic image creation process.

4.2.2 Proposed Digital Watercolor Image Creation Process

In principle, the digital watercolor image creation process proposed in this study and described in this section is an auxiliary for the digital pointillistic image creation process. That is, we can transform an original digital image into a digital watercolor image before creating a digital pointillistic image, and this is helpful to let the output digital pointillistic image becomes more like a real painting. Of course, this is not an essential process, and we can either apply it or not before performing the digital pointillistic image creation process.

The operations of the proposed method include quantization and filtering as shown in Figure 4.2. We quantize an input digital image into nine bits per pixel, that is, three bits per channel of R, G, or B of each single pixel. Then, we apply a voting filter to deal with the quantized image. For each pixel in the quantized image, we accumulate the number of the colors of the pixels *within a square* (the square size being 5×5), and the square is centered at the currently-processed pixel. We find out the color, say denoted as C_{new}, which has the maximum accumulation value, and set the color of that pixel to be C_{new}.

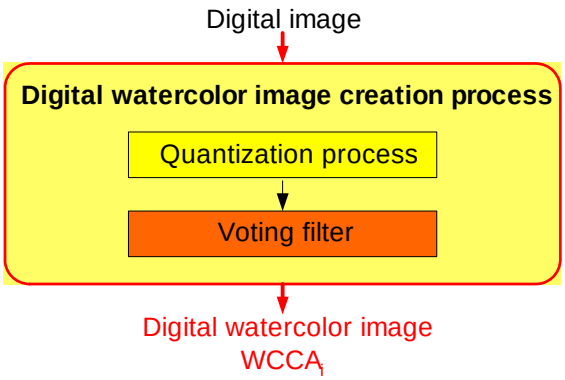


Figure 4.2 Digital watercolor image creation process.

4.2.3 Proposed Pointillistic Image Creation Process

As mentioned in Section 4.2.1, we have to enlarge and overlap the color pixels of an input digital image to form a digital pointillistic image. As shown in Figure 4.3, we set the value of *DAD* (meaning the distance between the centers of two adjacent *EOCDs*) as "5," and set the value of *DOD* (meaning the diameter of each $EOCD_i$) as "8."

Before drawing the *EOCDs* in the digital pointillistic image, we create a *random array* (denoted as *RA*) first by applying a random array creation process illustrated in Figure 4.4. The *RA* is an array with the drawing order of the *EOCDs*, and the detailed process is described as an algorithm below.

Algorithm 4.1: Scheme of digital pointillistic image creation process.

Input: A digital image.

Output: A digital pointillistic image.

Steps:

Step 1 Figure out how many *EOCDs* will be drawn on a digital pointillistic image, and denote the number as *NOD*.

Step 2 Create an *RA* utilizing the *NOD* derived by Step 1 and a random number generator.

Step 3 Decide a drawing order by applying the *RA* derived by Step 2, and draw the *EOCDs* on their proper locations with their specific colors.

In Step 1, we figure out the width (denoted as *ImageWidth*) and the height (denoted as *ImageHeight*) of the input digital image first, and then derive the value of

the *NOD* by applying Formula (4.1) below:

$$NOD = \frac{imageWidth}{DAD} \times \frac{imageHeight}{DAD} \tag{4.1}$$

In Step 2, we create an empty array (denoted as *UA*) first, with the size of the *UA* being equal to the value of *NOD*, and set the value of each element of the array as "0" to "$NOD-1$" in order. Then, we utilize a random number generator, and set the "generation range" of that from "0" to "$NOD-1$." Besides, we create two other empty arrays, namely, *ID*_0 and *ID*_1 with sizes equal to the value of *NOD*, too. We set the value of element ID_0_i as the $(2x-1)$-th value generated from the random number generator, and the value of ID_1_i as the $(2x)$-th value generated from the random number generator, where the parameters $i = 0 \sim NOD-1$ and $x = 1 \sim NOD$. Then, we swap the element in UA_{ID_0i} and the element in UA_{ID_1i} in order, and denote the swapped *UA* as *random array RA*. A flowchart of the random array creation process is illustrated in Figure 4.4.

In Step 3, first, by utilizing the values of *DAD* and *DOD*, we can figure out the coordinates of the center of each $EOCD_i$ in order. Second, we get the RGB values of the pixels at these coordinates of an input digital image, and then store the RGB values into a *Center-Color-Array* (denoted as *CCA*) in order. Furthermore, if we apply the digital watercolor image creation process proposed in Section 4.2.2 before applying the digital pointillistic image creation process to deal with the input digital image, we will derive a digital watercolor image. Then, we get the RGB value of the pixel at the center of each $EOCD_i$ of the digital watercolor image, and then store the RGB values into a *digital watercolor image's Center-Color-Array* (denoted as *WCCA*).

Finally, we apply the drawing order stored in the *RA* to draw each $EOCD_i$ with the RGB value obtained from the CCA_i or $WCCA_i$ in a digital pointillistic image.

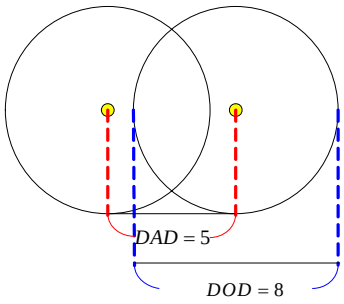


Figure 4.3 An illustration of *DAD* and *DOD*.

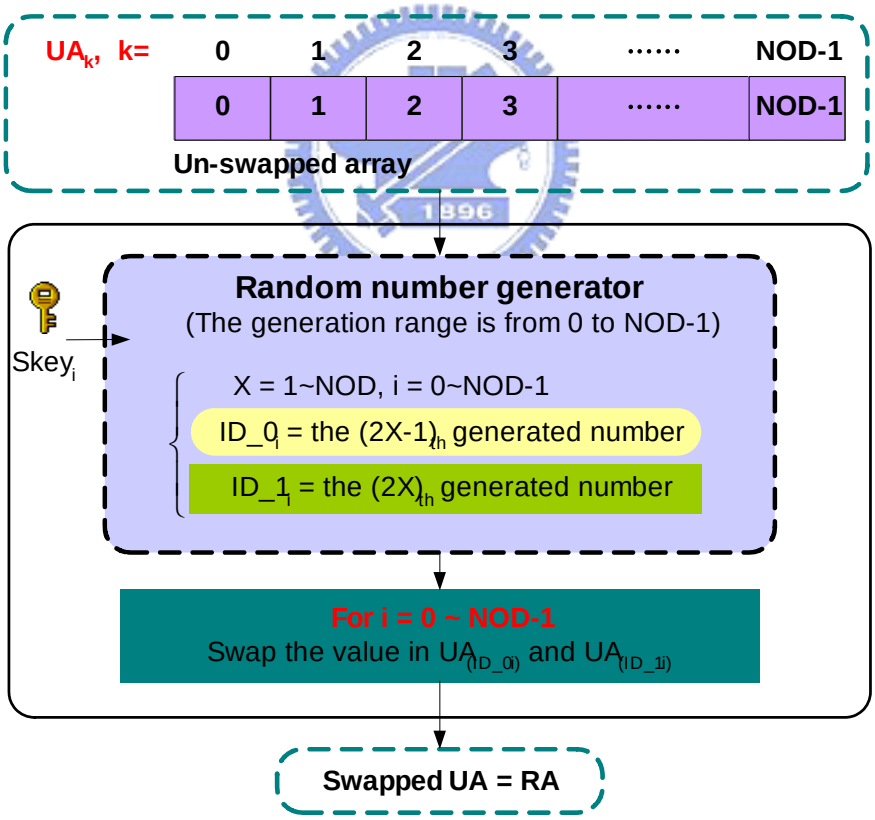


Figure 4.4 A flowchart of the random array creation process.

4.2.4 Experimental Results and Discussions

Figure 4.5(a) is an original image. Figure 4.5(b) is the final digital pointillistic image created form Figure 4.5(a) by applying *directly* the digital pointillistic image creation process proposed in Section 4.2.3. Figure 4.5(c) is the final digital watercolor image created form Figure 4.5(a) by applying the digital watercolor image creation process proposed in Section 4.2.2. Figure 4.5(d) is the final digital pointillistic image created from the digital watercolor image shown in Figure 4.5(c).

Figure 4.6, Figure 4.7, and Figure 4.8 show some other experimental results of this section.

(a)

Figure 4.5 Experimental results. (a) An original image. (b) A digital pointillistic image of (a). (c) A digital watercolor image of (a). (d) A digital pointillistic image of (c).

(b)

Figure 4.5 Experimental results. (a) An original image. (b) A digital pointillistic image of (a). (c) A digital watercolor image of (a). (d) A digital pointillistic image of (c). (continued).

(c)

Figure 4.5 Experimental results. (a) An original image. (b) A digital pointillistic image of (a). (c) A digital watercolor image of (a). (d) A digital pointillistic image of (c). (continued).

(d)

Figure 4.5 Experimental results. (a) An original image. (b) A digital pointillistic image of (a). (c) A digital watercolor image of (a). (d) A digital pointillistic image of (c). (continued).

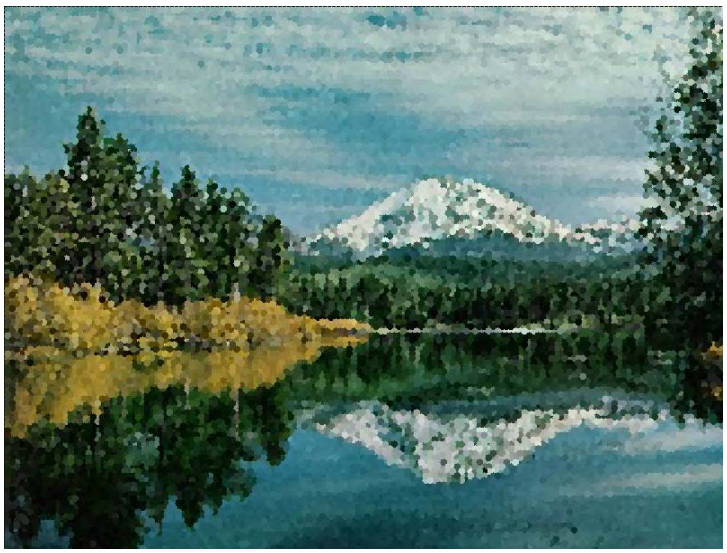
(b)

(c)

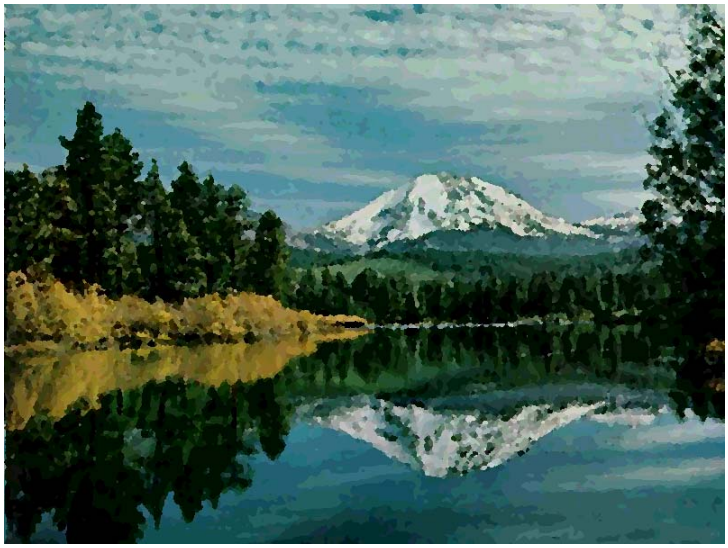
(a)

(d)

Figure 4.6 Experimental results. (a) An original image. (b) A digital pointillistic image of (a). (c) A digital watercolor image of (a). (d) A digital pointillistic image of (c).

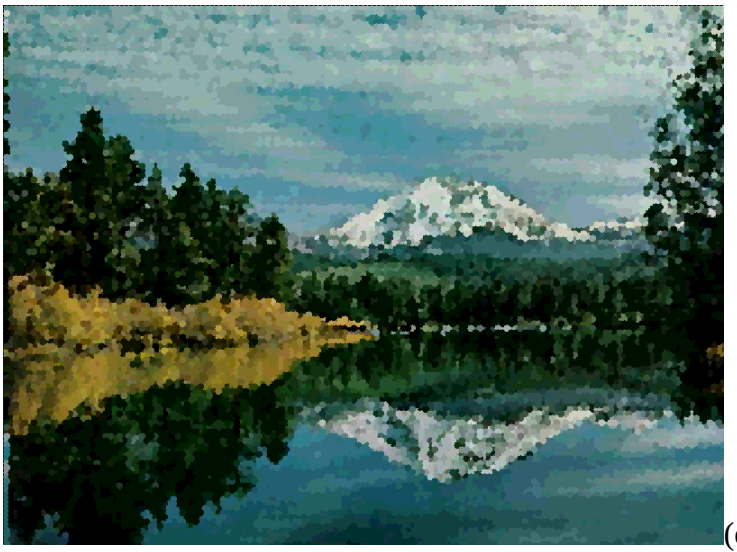


(b)



(c)

(a)



(d)

Figure 4.7 Experimental results. (a) An original image. (b) A digital pointillistic image of (a). (c) A digital watercolor image of (a). (d) A digital pointillistic image of (c).

(b)

(c)

(a)

(d)

Figure 4.8 Experimental results. (a) An original image. (b) A digital pointillistic image of (a). (c) A digital watercolor image of (a). (d) A digital pointillistic image of (c).

4.3 Proposed Data Hiding in Pointillistic Images by A Palette Color Coding Technique

4.3.1 Core Concept

The main concept of proposed Information hiding in digital pointillistic images by a palette color coding technique is to utilize the variations of the RGB values of *enlarged-and-overlapped-color-dots* (denoted as *EOCDs*) of a digital pointillistic image to implement the information hiding process. For this aim, we derive a *palette image* (or a *palette watercolor image*) by applying a palette color coding technique, which will be proposed in Section 4.3.3 to deal with an input digital image (or a digital watercolor image) before applying the digital pointillistic image creation process. Then, we get the RGB values of the pixel at the center of each $EOCD_i$ of the *palette image* (or the *palette watercolor image*), and store the RGB values into a *palette* of the *CCA* (denoted as *PCCA*) or a *palette* of the *WCCA* (denoted as *PWCCA*). Note that the *CCA* means the *Center-Color-Array* and that the *WCCA* means the *digital watercolor image's Center-Color-Array* as mentioned in Section 4.2.3. A flowchart of the *PCCA* and the *PWCCA* deriving processes is shown in Figure 4.9.

A flowchart illustrating the above core concept is shown in Figure 4.9. If we want to embed a bit code, $Data_i$, into $PCCA_i$ (or $PWCCA_i$), as shown in Figure 4.9(a), by applying the data combination and disarrangement process proposed in Section 4.3.2 with utilizing the input secret key (denoted as $Skey_i$), we derive a *Disordered* $Data_i$ (denoted as $DData_i$), and we can then know whether we need to modify the RGB values of $PCCA_i$ (or $PWCCA_i$) or not by checking if the bit value of the $DData_i$

is equal to zero or not. If the answer is "Yes," we will modify the RGB values of $PCCA_i$ (or $PWCCA_i$), and then derive a *data embedded Center-Color-Array* (denoted as $ECCA_i$). We apply the proposed digital pointillistic image creation process with the RBG values stored in the *ECCA* to create a data-embedded digital pointillistic image.

While performing the data extraction process proposed in Section 4.3.4, we apply the palette color coding technique proposed in Section 4.3.3 first, to deal with a data-embedded digital pointillistic image, and derive a *palette data-embedded digital pointillistic image's Center-Color-Array* (denoted as *PECCA*). Referring to Figure 4.10(b), for each $EOCD_i$ of a digital pointillistic image, we check whether the $ECCA_i$ is equal to the $PECCA_i$ or not. By the checking result, we can realize whether the RGB values of this $EOCD_i$ have been modified or not. If the $ECCA_i$ is equal to the $PECCA_i$, we view this $EOCD_i$ as a non-modified one. Otherwise, we will view this $EOCD_i$ as a modified one. Eventually, we derive a $DData_i$, and by applying the data recovering process discussed in Section 4.3.4, we can recover a $Data_i$ from the $DData_i$. The details of the data extraction process will be described in Section 4.3.4.

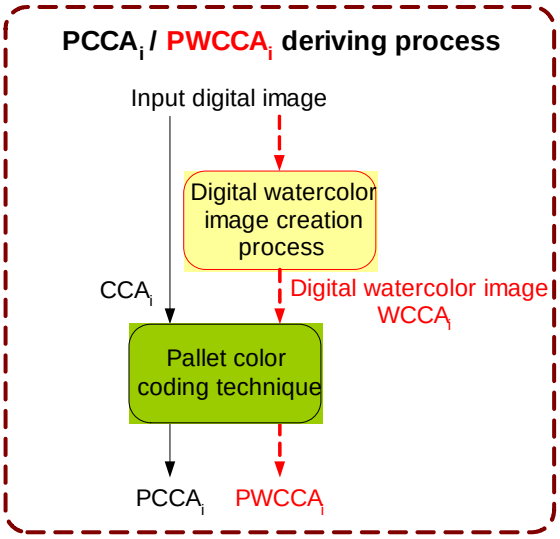
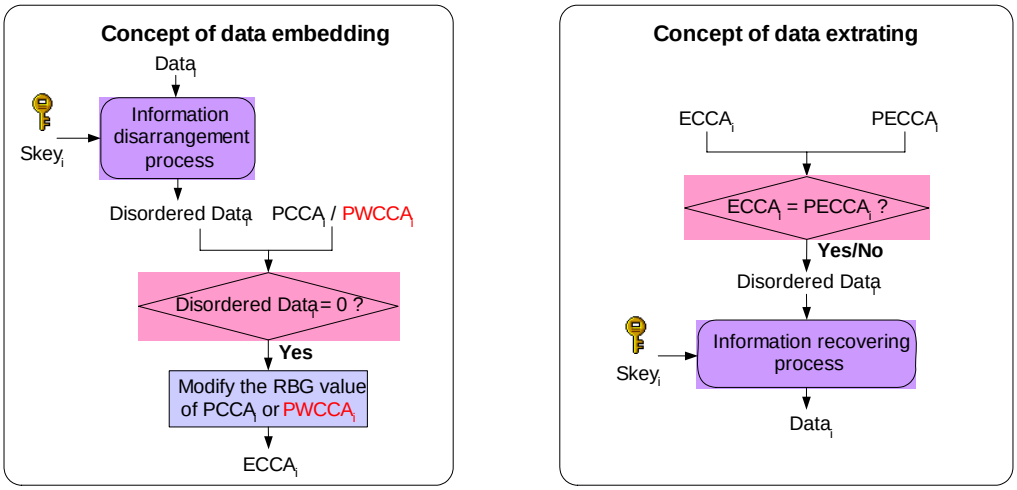


Figure 4.9 A flowchart of a PCCA and a PWCCA deriving processes. (We can follow the black arrows to derive the $PCCA_i$ or follow the red arrows to derive the $PWCCA_i$).



(a) (b)

Figure 4.10 The core concept of information hiding in digital pointillistic images by a palette color coding technique.

4.3.2 Data Combination and Data Disarrangement Process

As mentioned in Chapter 3, we embed data into three features of a digital puzzle image, but in this chapter, we can only embed data into one feature, i.e., the RGB values of each $EOCD_i$, of a digital pointillistic image. However, the capacity of the embedded data is equal to the *NOD* of a digital pointillistic image, that is, as long as the digital pointillistic image is not too small, the capacity will be quite large. So, we try to combine two kinds of secret information and embed them in the digital pointillistic image in the mean time.

Besides, the secret information might be extracted by people who know this algorithm, so it is necessary to utilize a secret key to disorder the secret information before embedding it.

The details of the data combination and disarrangement process are described as an algorithm below.

Algorithm 4.2: Data combination and disarrangement process.

Input: A bit sequence of a secret message (denoted as Mes_i), a bit sequence of a watermark (denoted as $Water_i$), and a bit sequence of a secret key (denoted as $SKey_i$).

Output: A disordered bit sequence of the combination of the inputs Mes_i and $Water_i$ (denoted as $DData_i$).

Steps:

Step 1 Create an empty array (denoted as *DData*), with the size of the *DData* being equal to the value of *NOD*.

Step 2 Derive an data array (denoted as *Data*) by appending an *ending pattern* of sixteen successive 0s at the end of Mes_i, that is, set $Mes_i = 0$ if $n-16 < i \le n$, and connect $Water_i$ after the ending pattern, and then, append the ending pattern again at the end of $Water_i$.

Step 3 Arrange the elements of *Data* into the *DData* in order.

Step 4 Randomly set the value of each redundant element of the *DData* (at the rear of the $Data_i$) as 0 or 1.

Step 5 Swap the elements of $DData_i$ by applying the random array creation process described in Section 4.2.3 utilizing the input $Skey_i$ (to perform the random number generation operation), and then derive a swapped $DData_i$.

In Step 1, by utilizing the ending pattern, we can determine where Mes_i and $Water_i$ end in a sequence of extracted bits after performing the proposed data extraction process. Besides, we can find out where $Water_i$ starts in the sequence of extracted bits by utilizing the ending pattern, too. The details of how to utilize the ending pattern to separate $Data_i$ into Mes_i and $Water_i$ will be discussed in Section 4.3.5.

4.3.3 Data Hiding Process

A flowchart of the data hiding process is shown in Figure 4.11. First, we derive a *Disordered Data*$_i$ (denoted as *DData*$_i$) from an input *Mes*$_i$ and an input *Water*$_i$, and derive a *PCCA* (or a *PWCCA*) from an input digital image (or a digital watercolor image). Then, we derive the *data embedded PCCA*$_i$ *or PWCCA*$_i$ (denoted as *ECCA*$_i$). Finally, we apply the proposed digital pointillistic image creation process with the RBG values stored in the *ECCA* to create a data-embedded digital pointillistic image.

Algorithm 4.3: Data hiding in digital pointillistic images by a palette color coding technique.

Input: A digital image (or a digital watercolor image); a *Mes*$_i$, a *Water*$_i$, and an *SKey*$_i$.

Output: A data embedded digital pointillistic image.

Steps:

Step 1 Derive a *Disordered Data*$_i$ (denoted as *DData*$_i$) from the inputs *Mes*$_i$ and *Water*$_i$ by applying the data combination and disarrangement process proposed in Section 4.3.2 utilizing the input *SKey*$_i$.

Step 2 Derive a *PCCA* (or a *PWCCA*) from the input digital image (or the digital watercolor image) by performing the palette color coding technique described as an algorithm below.

Step 3 Derive an *ECCA* by checking *DData*$_i$. If *DData*$_i$ is equal to zero, modify the RGB values of the *PCCA* (or the *PWCCA*) by the method illustrated in Figure 4.12.

Step 4 Apply the proposed digital pointillistic image creation process proposed in Section 4.2.3 with the RBG values stored in the *ECCA* to create a data embedded digital pointillistic image.

Of course, because of the data embedding process, some RGB values of the *ECCA* are different from those of the *CCA* or the *WCCA*. However, in our opinion, this will not destroy the aesthetic of the digital pointillistic image.

Before performing the proposed palette color coding technique, we have to establish a *color table*, denoted as *CT*, which is an array of RGB values. All of the elements in *CT* are decided by the user, that is, the user can choose his/her favorite colors and arrange them into *CT*. After applying the proposed palette color coding technique to process a digital image, the colors of the output digital image (denoted as *palette image*) will all be in *CT*.

Algorithm 4.4: Palette color coding technique.

Input: A *digital image's Center-Color-Array* (denoted as *CCA*).

Output: A *Palette CCA* (denoted as *PCCA*).

Steps:

Step 1 Establish a *CT*.

Step 2 For each CCA_i, compute the distance between CCA_i and CT_i.

Step 3 Figure out the CCA_i with the minimum distance, and replace the RGB values of the minimum-distance CCA_i with those of CT_i. Then, rename CCA_i as $PCCA_i$.

In Step 2, if (R_A, G_A, B_A) represents the RGB values of CCA_i, and (R_B, G_B, B_B) represents the RGB values of CT_i, then we compute the distance between them by $(R_B - R_A)^2 + (G_B - G_A)^2 + (B_B - B_A)^2$. Besides, if an input array is $WCCA_i$ or $ECCA_i$, then the output array will be $PWCCA_i$ or $PECCA_i$.

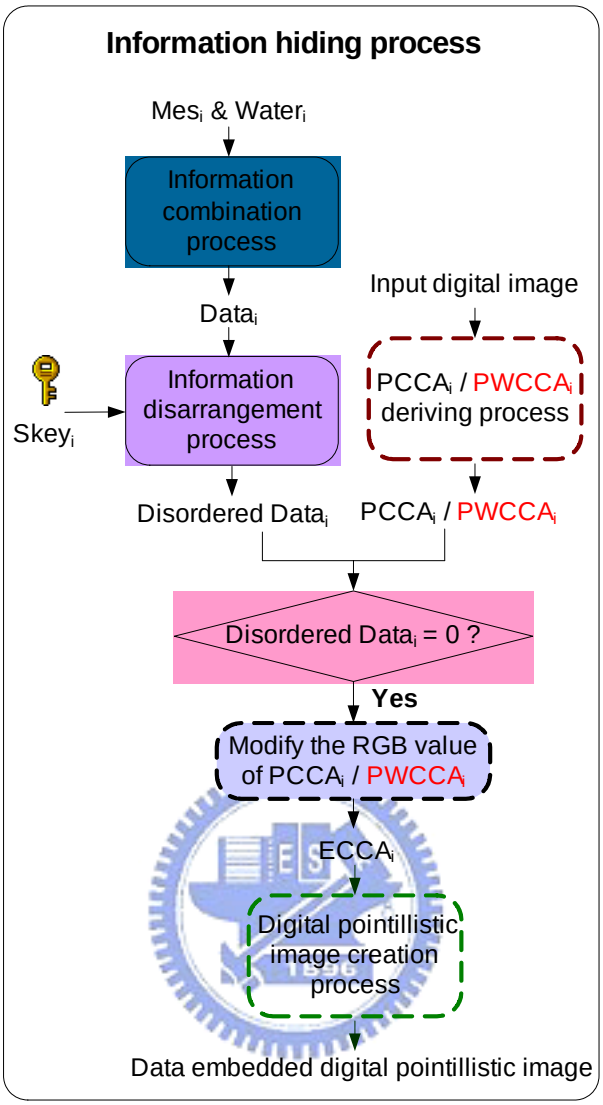


Figure 4.11 A flowchart of the information hiding in digital pointillistic images by a palette color technique.

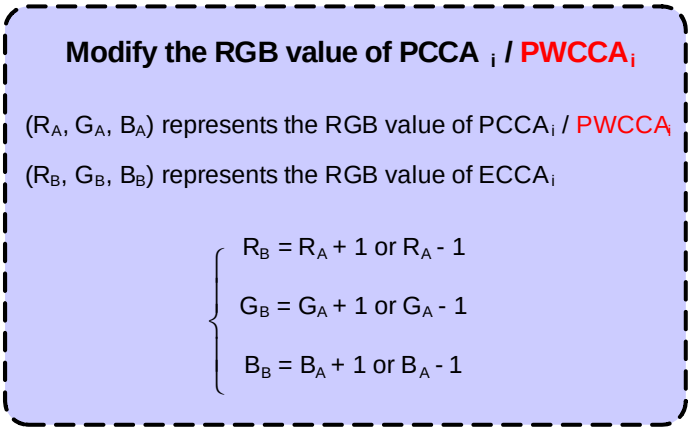


Figure 4.12 A diagrammatic explanation of how to modify the RGB value of PCCA$_i$ /PWCCA$_i$ while the Disordered Data$_i$ = 0.

4.3.4 Data Extraction Process

A flowchart of the data extraction process is shown in Figure 4.13, and the procedure is described as an algorithm below.

Algorithm 4.5: Data extracting from digital pointillistic images by a palette color technique.

Input: A data embedded digital pointillistic image, and an $SKey_i$.

Output: A Mes_i, and a $Water_i$.

Steps:

Step 1 Utilize the values of *DAD* and *DOD* defined in Section 4.2.3 to figure out the *data-embedded Center-Color-Array* (denoted as *ECCA*) of the input digital pointillistic image.

Step 2 Derive $PECCA_i$ from $ECCA_i$ by performing the palette color coding technique proposed in Section 4.3.3.

Step 3 Compare each $ECCA_i$ with each $PECCA_i$, and check if $ECCA_i$ is equal to $PECCA_i$ or not to derive a $DData_i$ in the following way.

3.1 Set $DData_i = 1$ if the answer is "yes."

3.2 Set $DData_i = 0$ if the answer is "no."

Step 4 Derive the embedded Mes_i and $Water_i$ from the $DData_i$ by applying the data recovering and separation process proposed in Section 4.3.5.

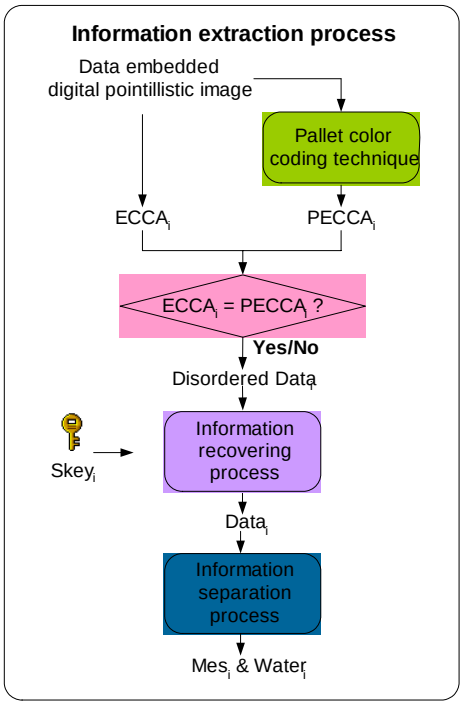


Figure 4.13 A flowchart of data extracting from digital pointillistic images by a palette color technique.

4.3.5 Data Recovering and Data Separation Process

The data recovering and separation process is an inverse version of the data combination and disarrangement process proposed in Section 4.3.2. It is described as an algorithm below.

Algorithm 4.6: Information recovering and separation process.

Input: A $DData_i$, and an $SKey_i$.

Output: A bit sequence of a secret message (Mes_i), and a bit sequence of a watermark ($Water_i$).

Steps:

Step 1 Swap the elements of $DData_i$ to derive a $Data_i$ by applying the "inverse" random array creation process illustrated in Figure 4.14, and start up the

random number generator with using $Skey_i$.

Step 2 Extract Mes_i from $Data_i$ in the following way.

2.1 Search $Data_i$ for the first *ending pattern* (16 successive 0s), and truncate the redundant bits at the rear of Mes_i.

Step 3 Extract $Water_i$ from $Data_i$ in the following way.

3.1 Search $Data_i$ for the first *ending pattern* (16 successive 0s), and get the index (denoted as w) of the first bit after the *ending pattern*. The element of $Data_w$ is taken as the first bit of $Water_i$.

3.2 Search $Data_i$ for the second *ending pattern* (16 successive 0s), and truncate the redundant bits at the rear of $Water_i$.

In Step 1, different from the processing steps of the random array creation process illustrated in Figure 4.4, we swap the values in $DData_{ID\_0i}$ and $DData_{ID\_1i}$, for $i = (NOD-1) \sim 0$ but not for $i = 0 \sim (NOD-1)$.

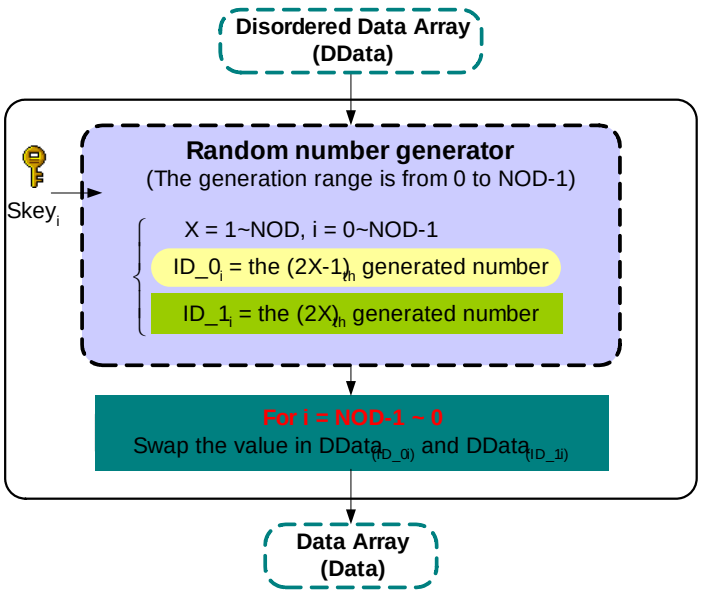
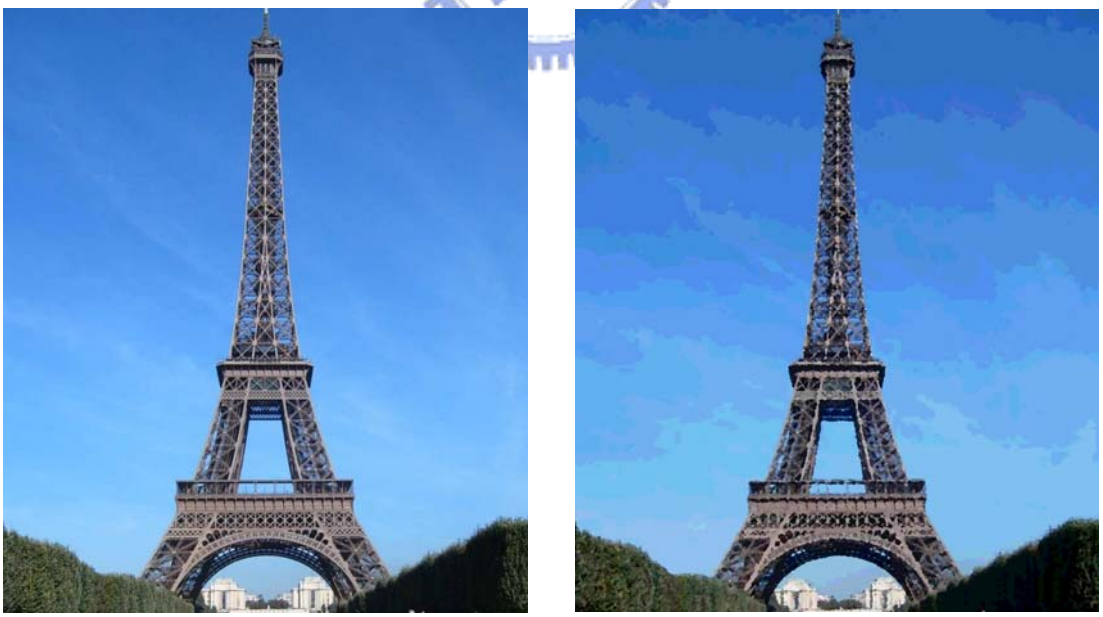


Figure 4.14 A flowchart of the data recovering process by utilizing the inverse random array creation process.

4.3.6 Experimental Results and Discussions

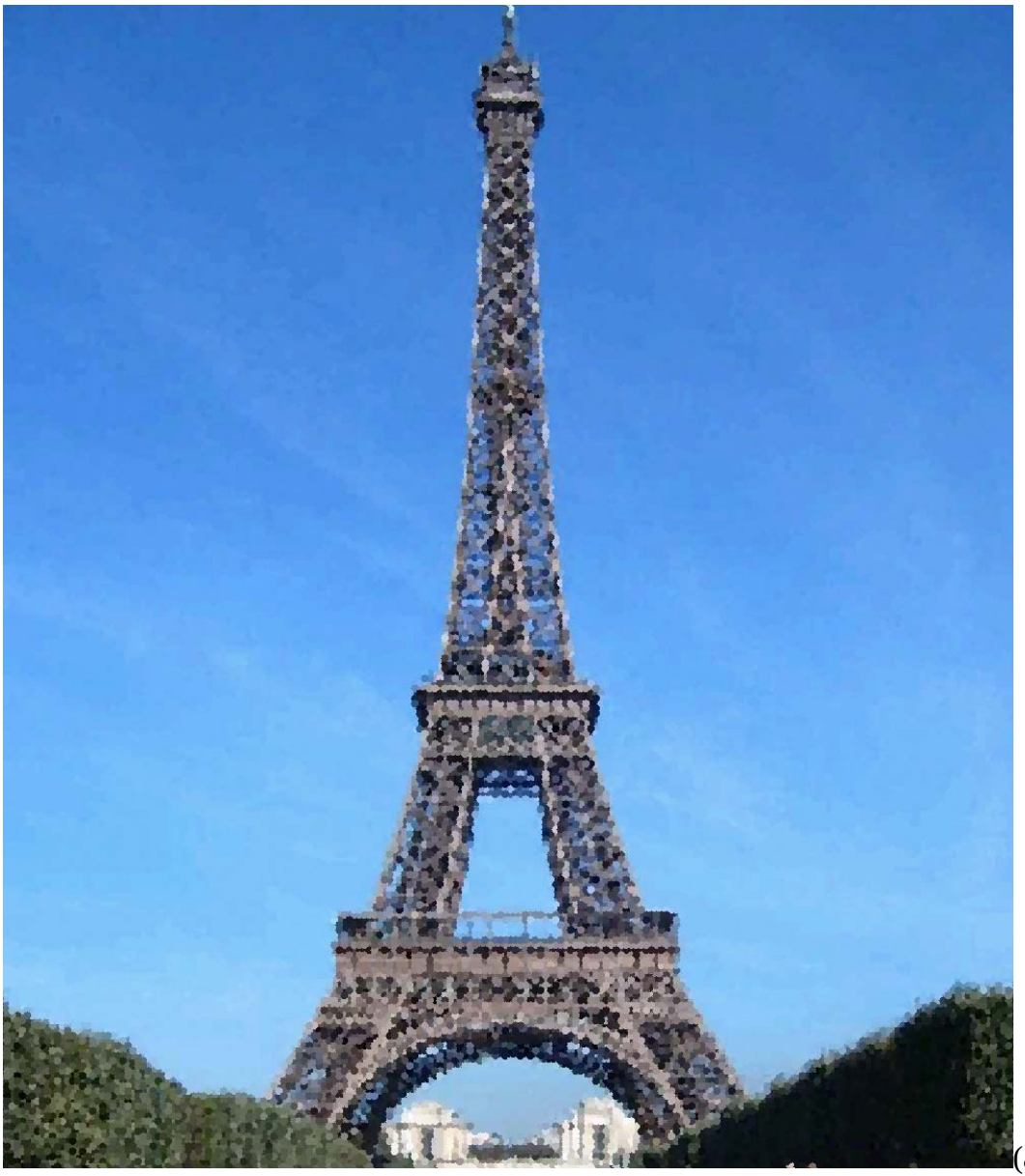
Figure 4.16 and Figure 4.17 show some experimental results of secret hiding in a digital pointillistic image. Figure 4.16(a) is a digital pointillistic image with the secret message, “阿祥，有一句話我一直很想告訴你，那就是我喜歡你!!,” and the watermark, Figure 4.16(c), embedded. Figure 4.16(b) is the secret message extracted from Figure 4.16(a) with a correct key. Figure 4.16(d) is the watermark extracted from Figure 4.16(a) with a correct key. Figure 4.16(e) shows the extraction result of Figure 4.16(a) with a wrong key.

The only one difference between Figure 4.16 and Figure 4.17 is: before performing the data hiding process proposed is Section 4.3, we have performed the proposed digital watercolor image creation process proposed in Section 4.2.2 first to deal with the Figure 4.15(a).

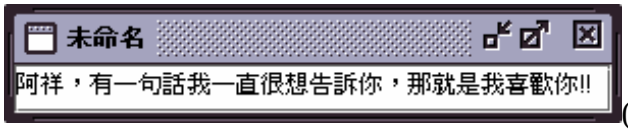


(a) (b)

Figure 4.15 An experimental result of a digital watercolor image (b) created from an original image (a).



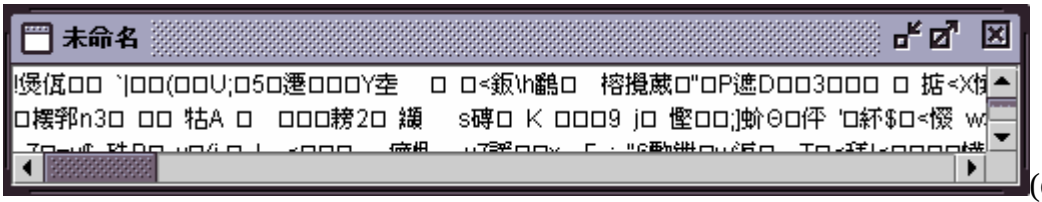
(a)



(b)

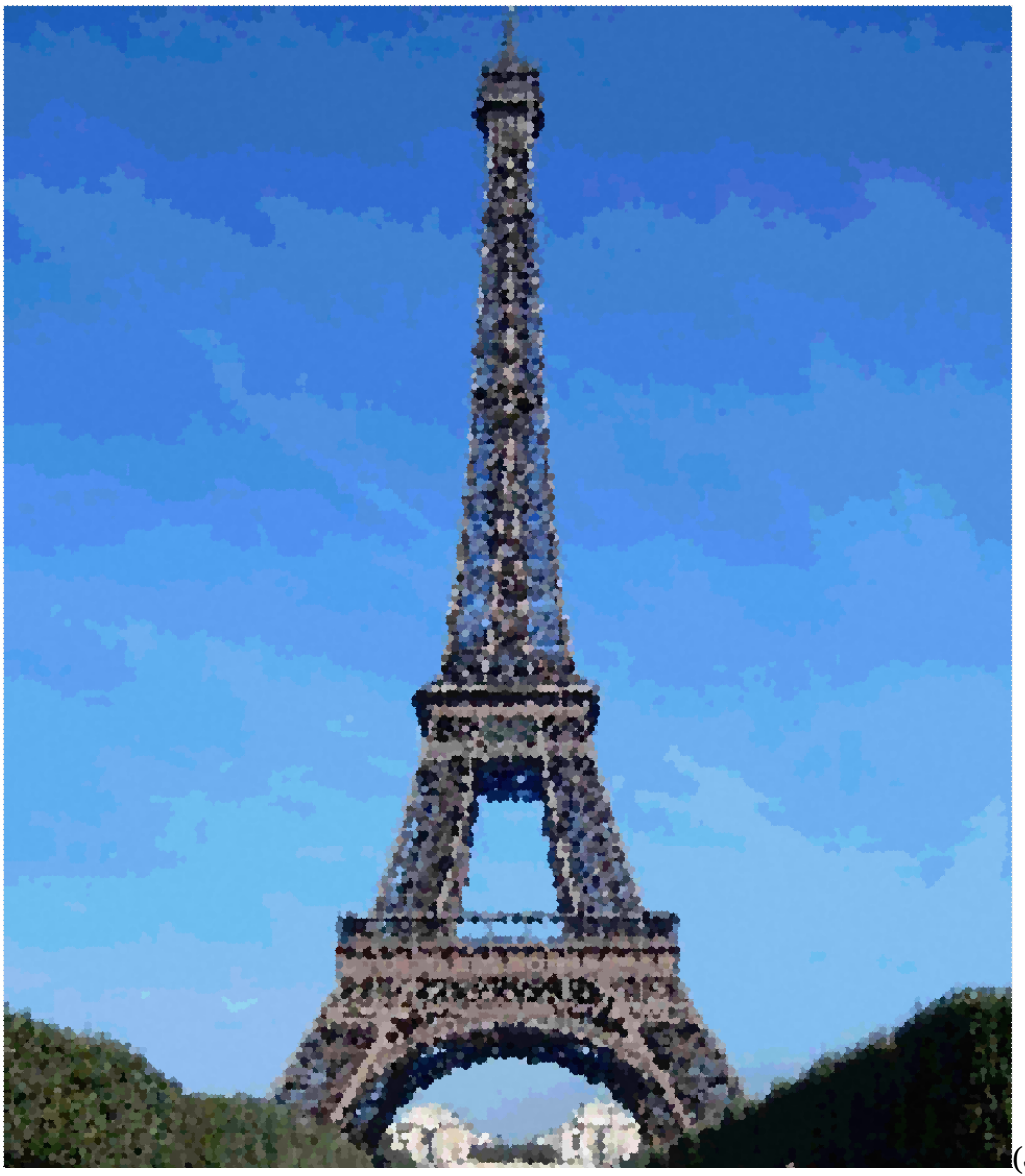
(c)

(d)

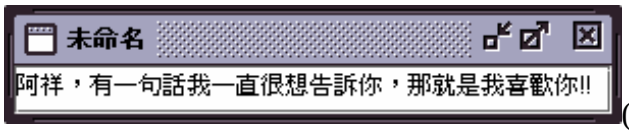


(e)

Figure 4.16 Experimental results of secret hiding in a pointillistic image. (a) A pointillistic image with secret message and the watermark, (c), embedded. (b) The secret message extracted from (a) with a correct key. (d) A watermark extracted from (a) with a correct key. (e) The extraction result of (a) with a wrong key.



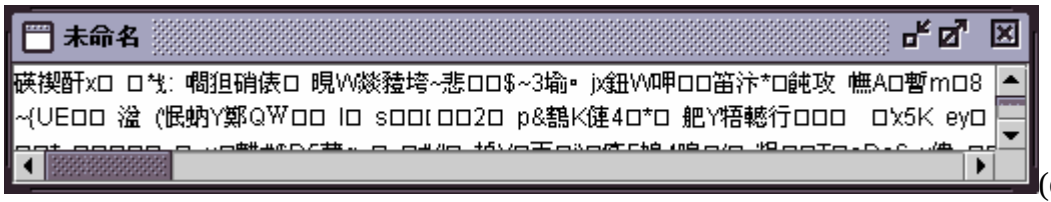
(a)



(b)

(c)

(d)



(e)

Figure 4.17 Experimental results of secret hiding in a pointillistic image. (Transform an original image into a digital watercolor image first) (a) A pointillistic image with secret message and the watermark, (c), embedded. (b) The secret message extracted from (a) with a correct key. (d) A watermark extracted from (a). (e) The extraction result of (a) with a wrong key.

Figure 4.18 shows the performing time of information embedding in and extracting from digital pointillistic images.

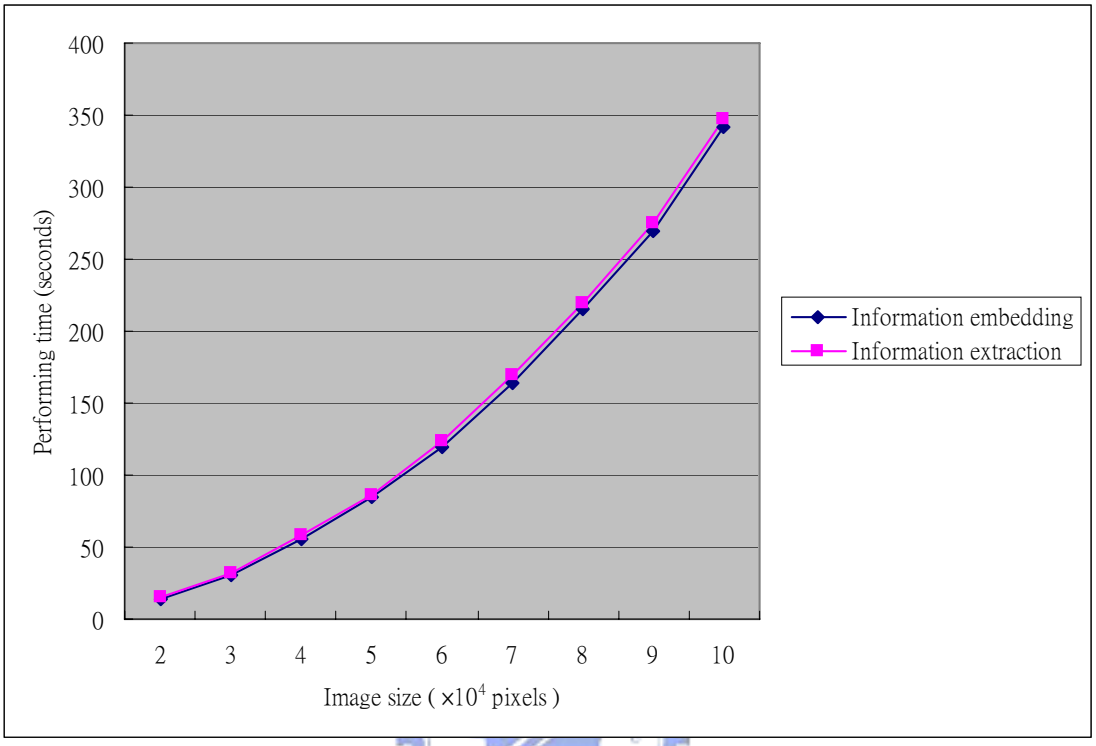


Figure 4.18 The performing time of experimental results.