

Chapter 2

A New Digital Puzzle Image Creation Method by Boundary Shape Parameterization for Information Hiding

2.1 Overview of Proposed Method

In this chapter, we propose a method to create a digital puzzle image automatically with an image as input. In order to implement the concept of information hiding in the digital puzzle image, we use a technique of boundary shape parameterization as a base of the proposed creation method. In Section 2.2, the proposed digital puzzle image creation method for the information hiding purpose will be described in detail. Furthermore, users may have their own choices (i.e., the size of secret data, the number of people who participate in the secret sharing, and their own favors) to decide the puzzle pieces size of the digital puzzle image.

More specifically, we decompose the puzzle image into a number of puzzle pieces, and divide them into several groups randomly. The number of groups is equal to the number of secret sharing participants. The proposed puzzle image decomposition method is described in Section 2.2.2.2. Naturally, we must find out a method that can recover the puzzle image automatically when all of the participants return their own parts of the puzzle pieces. Because this automatic digital puzzle image reconstruction process is based on searching the puzzle pieces for correct shapes, we should try the best to increase the possible number of distinct shapes. The

digital puzzle image reconstruction process will be described in Section 2.3. Finally, some experimental results will be presented in Section 2.4.

2.2 Proposed Digital Puzzle Image Creation Process

2.2.1 Ideas of Creation Process

2.2.1.1 Properties of Orientation, Size, and Angle of Puzzle Pieces

The orientation, size, and angle of the puzzle piece are three major parameters of the proposed digital puzzle image creation process. We denote the orientation, size and angle of a puzzle piece as OT , SZ , and AG , respectively, in the sequel. The properties of these three features are illustrated in Figure 2.1. Figure 2.1(a), (b), (c), and (d) illustrate the four possible “orientations” of a puzzle piece. Each picture is composed of a triangle, a big circle, and two small circles. The notation $BigR$ represents the radius of the big circle in the middle of the picture. We utilize $BigR$ to represent the feature of size, which is the area of the big circle. The notation θ represents the angle of the triangle. We utilize θ to represent the feature of angle.

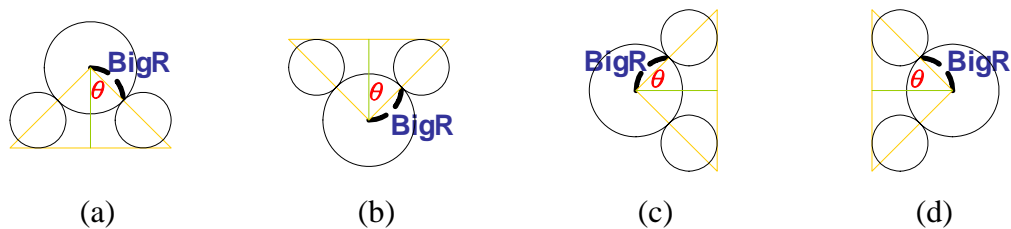


Figure 2.1 Properties of orientation, size, and angle of digital puzzle pieces.

2.2.1.2 Composition of Parameterized Boundary Shape Segments

In order to embed information in a digital puzzle Image, we propose a new digital puzzle image creation method by boundary shape parameterization. After performing this method, a digital puzzle image will result, which is composed of digital puzzle pieces. Each digital puzzle piece, as illustrated in Figure 2.2, is composed of “boundary shape segments.”

Each digital puzzle piece has two vertical sides and two horizontal sides. We choose one *OT* from Figure 2.1(c) or Figure 2.1(d) to compose a vertical side, and choose one *OT* from Figure 2.1(a) or Figure 2.1(b) to compose a horizontal side. Take Figure 2.2, which represents the sketch of a digital puzzle piece, as an example, the south, north, west, and east sides of it are respectively composed of the *OTs* chosen from Figure 2.1(a) , (b), (c), and (d) in order. Detailed discussions of digital puzzle image creation processes will be stated in Section 2.2.2.

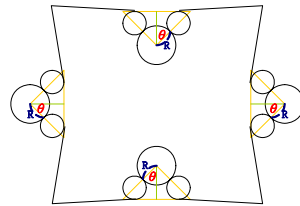


Figure 2.2 Sketch of a digital puzzle pieces.

2.2.1.3 Concepts of Information Hiding by Boundary Shape Parameterization

Since the boundary shape of the digital puzzle piece has been parameterized and is composed of several segments, we can embed the information in a digital puzzle

image by modifying the shapes of the puzzle pieces. That is, the concept of the information hiding process is based on modifying the shape of each puzzle piece in a digital puzzle image. Therefore, when we want to extract the embedded information from a digital puzzle image, we can detect the shapes of the puzzle pieces of the digital puzzle image and then derive the embedded information by analyzing the features of the detected shapes.

In this study, we sequentially embed three kinds of data into the three parameters which were introduced in Section 2.2.1.1. First, we embed a secret message by modification of the parameter of orientation using the feature of *OT*. Second, we embed a watermark by modifying the parameter of size represented by *SZ*. Finally, we generate authentication signals and embed them by modifying the parameter of angle represented by *AG*. By embedding the given data, a user can achieve the purpose of covert communication, copyright protection, or image authentication. All the details of the procedures for embedding the three types of features and the corresponding information extraction processes will be described in Chapter 3.

2.2.2 Details of Digital Puzzle Image Creation Process

2.2.2.1 Scheme of Image Creation Process

The concept of the digital puzzle image creation process is to create a digital puzzle image by composing parameterized boundary shapes. Before performing the creation process, every pixel with the RGB value of (0, 0, 0) in the input digital image is replaced with the RGB value of (2, 2, 2). Because of this pre-processing, there will be no pixel with the RGB value of (0, 0, 0) except the boundary lines of each digital

puzzle piece in the output digital puzzle image. This pre-processing is necessary in order to make the digital puzzle image decomposition process smooth, and the details will be described in Section 2.2.2.2.

In the sequel, we denote the numbers of puzzle pieces of a digital puzzle image in the horizontal and vertical directions as $HParts$ and $VParts$, respectively, and the width and height of the input image as $ImageWidth$ and $ImageHeight$, respectively. Referring to the illustration shown in Figure 2.3, we derive the value of $HParts$ by dividing $ImageWidth$ by the digital puzzle piece size denoted as PPS , which is a value selected by a user, and derive the value of $VParts$ by dividing $ImageHeight$ by PPS . Referring to Figure 2.4 and Figure 2.5, we denote the lines of a digital puzzle image in the horizontal and vertical directions as $Xaxis_j$ and $Yaxis_j$, respectively, and the “puzzle piece starting points” on $Xaxis_j$ and $Yaxis_j$ as HS_i and VS_i , respectively. The index j of $Xaxis_j$ and $Yaxis_j$ indicates the processing order of the horizontal and the vertical lines of a digital puzzle image, respectively, and the index i of HS_i and VS_i indicates the processing order of HS_i and VS_i , respectively.

Before introducing the puzzle creation process, we define some parameters first. They are $UpDown$, $BigR$, and θ . The value of the parameter $UpDown$ is either “1” or “-1.” The value of the parameter $BigR$ is either “BR” or “SR”, and the values of BR and SR are derived by Formula (2.1) and (2.2) below, respectively:

$$BR = \frac{PPS \times 4.2}{27} + 1; \quad (2.1)$$

$$SR = \frac{PPS \times 4.2}{27}. \quad (2.2)$$

The value of θ is either “40°” or “48°.” We will assign random values to these parameters during the digital puzzle image creation process.

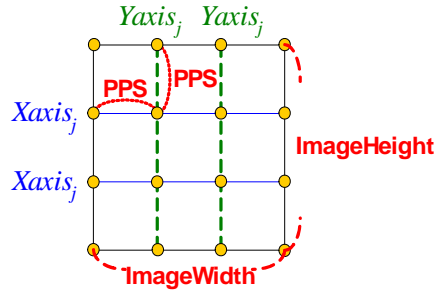


Figure 2.3 Utilizing the input parameter, PPS , to decide the locations of each $Xaxis_j$ and each $Yaxis_j$ of a digital puzzle image.

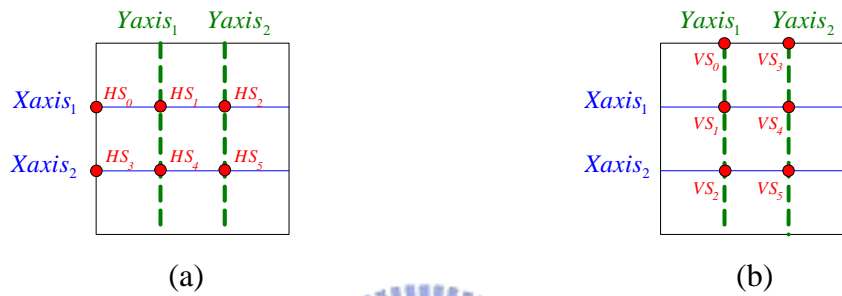


Figure 2.4 The processing order of $Xaxis_j$, $Yaxis_j$, HS_i , and VS_i .

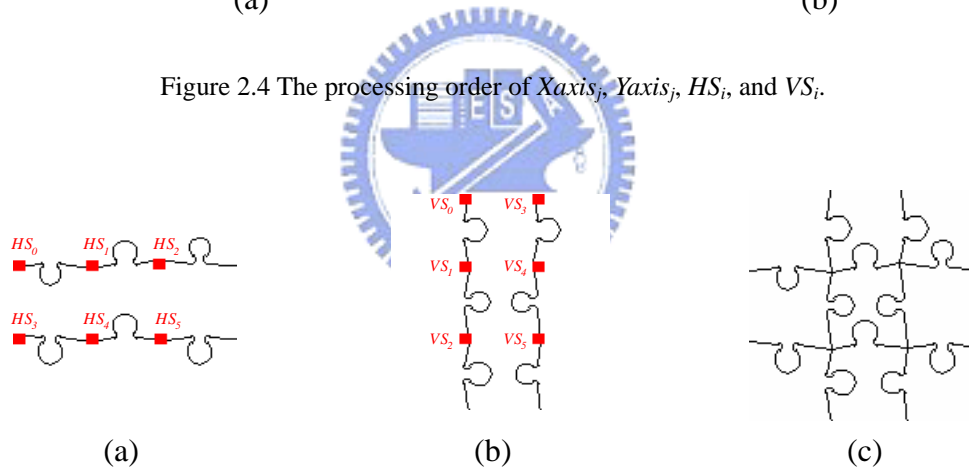


Figure 2.5 An illustration of utilizing the locations of HS_i and VS_i to draw curves. (a) An illustration of drawing each curve from each HS_i . (b) An illustration of drawing each curve from each VS_i . (c) An illustration of combining the horizontal curves in (a) with the vertical curves in (b).

Algorithm 2.1 : Digital puzzle image creation process for information hiding by boundary shape parameterization.

Input: A digital image I and a digital puzzle piece size denoted as PPS .

Output: A digital puzzle image.

Steps:

Step 1 Utilize the value PPS to decide the locations of each $Xaxis_j$ and each $Yaxis_j$ of I .

A diagrammatic explanation is illustrated in Figure 2.3.

Step 2 Deal with the X -axis parts of I .

2.1 Find out each HS_i of each $X-axis_j$ in order, and a diagrammatic explanation is illustrated in Figure 2.4(a). If we denote the value of HS_i as i , and the value of $Xaxis_j$ as j , we can derive the coordinates of HS_i by applying Formula (2.3) below:

$$HS_i(x, y): \left\{ \begin{array}{l} x = (HS_i \bmod HParts) \times PPS \\ y = Xaxis_j \times PPS \end{array} \right\}. \quad (2.3)$$

2.2 Find the coordinates of points D , E , M , and A illustrated in Figure 2.6 by applying Formula (2.4) to (2.7), respectively, below:

$$D(x_1, y_1): \left\{ \begin{array}{l} x_1 = x + \frac{1}{3} \times PPS \\ y_1 = y \end{array} \right\}; \quad (2.4)$$

$$E(x_2, y_2): \left\{ \begin{array}{l} x_2 = x + \frac{2}{3} \times PPS \\ y_2 = y \end{array} \right\}; \quad (2.5)$$

$$M(x_3, y_3): \left\{ \begin{array}{l} x_3 = x + \frac{1}{2} \times PPS \\ y_3 = y \end{array} \right\}; \quad (2.6)$$

$$A(x_4, y_4): \left\{ \begin{array}{l} x_4 = x + \frac{1}{2} \times PPS \\ y_4 = y + UpDown \times \left(\frac{PPS/6}{\tan \theta} \right) \end{array} \right\}. \quad (2.7)$$

2.3 Find the distances of \overline{AD} , \overline{AE} , and \overline{CE} illustrated in Figure 2.6 by

applying Formula (2.8) to (2.10), respectively, below:

$$\overline{AD} = \sqrt{(x_1 - x_4)^2 + (y_1 - y_4)^2}; \quad (2.8)$$

$$\overline{AE} = \sqrt{(x_2 - x_4)^2 + (y_2 - y_4)^2}; \quad (2.9)$$

$$\overline{CE} = \overline{AE} - \text{Big}R. \quad (2.10)$$

2.4 Find the coordinates of points B and C illustrated in Figure 2.6 by applying Formula (2.11) and (2.12), respectively, below:

$$B(x_5, y_5): \left\{ \begin{array}{l} x_5 = \frac{x_4 - \text{Big}R \times (x_4 - x_1)}{\overline{AD}} \\ y_5 = \frac{\text{Big}R \times \text{UpDown} \times (-1) \times (y_1 - y_4)}{\overline{AD}} + y_4 \end{array} \right\}; \quad (2.11)$$

$$C(x_6, y_6): \left\{ \begin{array}{l} x_6 = \frac{\text{Big}R \times (x_2 - x_4)}{\overline{AD} + x_4} \\ y_6 = \frac{\text{Big}R \times \text{UpDown} \times (-1) \times (y_2 - y_4)}{\overline{AD}} + y_4 \end{array} \right\}. \quad (2.12)$$

2.5 Find the value of $\text{Small}R$, which denotes the radius of the small circle shown in Figure 2.6 by applying Formula (2.13) below:

$$\text{Small}R = \frac{\overline{CE} \times \cos \theta}{1 + \cos \theta}. \quad (2.13)$$

2.6 Find the coordinates of points H and I illustrated in Figure 2.6 by applying Formula (2.14) and (2.15), respectively, below:

$$H(x_7, y_7): \left\{ \begin{array}{l} x_7 = x_4 - \frac{(x_4 - x_5) \times (\text{Big}R + \text{Small}R)}{\text{Big}R} \\ y_7 = y_4 + \frac{(\text{Big}R + \text{Small}R) \times \text{UpDown} \times (-1) \times (y_5 - y_4)}{\text{Big}R} \end{array} \right\}; \quad (2.14)$$

$$I(x_8, y_8) : \left\{ \begin{array}{l} x_8 = x_4 + \frac{(x_6 - x_4) \times (BigR + SmallR)}{BigR} \\ y_8 = y_4 + \frac{(BigR + SmallR) \times UpDown \times (-1) \times (y_6 - y_4)}{BigR} \end{array} \right\}. \quad (2.15)$$

2.7 Find the coordinates of points J and K illustrated in Figure 2.6 by applying Formula (2.16) and (2.17), respectively, below:

$$J(x_9, y_9) : \left\{ \begin{array}{l} x_9 = x_7 \\ y_9 = y_7 - UpDown \times SmallR \end{array} \right\}; \quad (2.16)$$

$$K(x_{10}, y_{10}) : \left\{ \begin{array}{l} x_{10} = x_8 \\ y_{10} = y_8 - UpDown \times SmallR \end{array} \right\}. \quad (2.17)$$

2.8 Draw \overline{BC} in the clockwise direction.

2.9 Draw \overline{BJ} in the clockwise direction, and draw \overline{CK} in the counter-clockwise direction.



Step 3 Find out each VS_i of each $Yaxis_j$ in order, and a diagrammatic explanation is illustrated in Figure 2.4(b). Deal with the $Y-axis$ parts of I in order with steps similar to those of Step 2.

Step 4 Utilize the four sides, and the coordinates of each $Xaxis_j$ and each $Yaxis_j$ of I to find the coordinates of the intersection points denoted as IP_i , which are the yellow points illustrated in Figure 2.3. Randomly shift the four kinds of distances in the horizontal and vertical directions, respectively, of each $IP_i(x_i, y_i)$. By applying Formula (2.18), compute the coordinates of the intersection points $Cross_i(x_c, y_c)$ as follows:

$$Cross_i(x_c, y_c) : \left\{ \begin{array}{l} x_c = x_i \text{ or } (x_i \pm \frac{PPS}{18} \text{ or } \pm \frac{PPS}{16}) \\ y_c = y_i \text{ or } (y_i \pm \frac{PPS}{18} \text{ or } \pm \frac{PPS}{16}) \end{array} \right\}. \quad (2.18)$$

Step 5 Connect points J and K illustrated in Figure 2.6 to their closest intersection point $Cross_i$ to accomplish the creation of I finally.

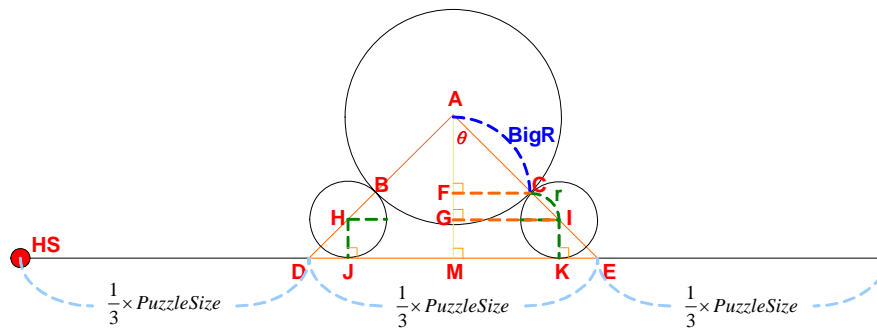


Figure 2.6 An illustration of a horizontal side of a digital puzzle piece.

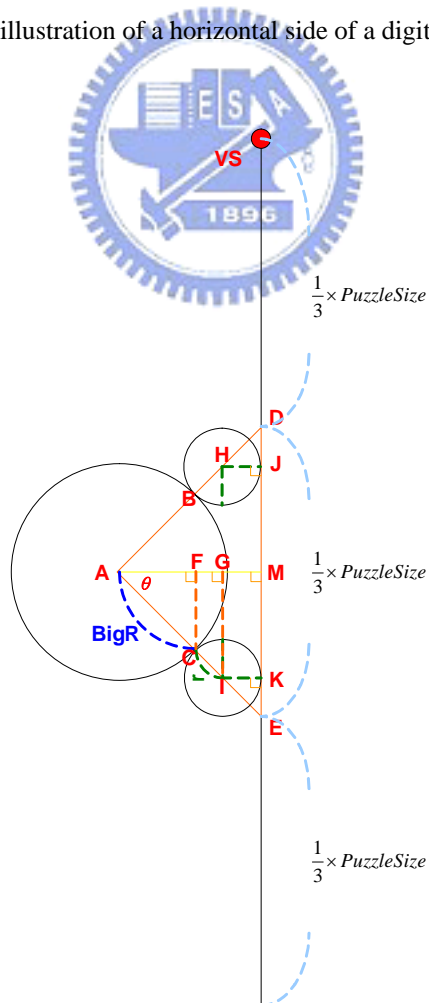


Figure 2.7 An illustration of a vertical side of a digital puzzle piece.

2.2.2.2 Use of Region Growing for Digital Puzzle Image Decomposition

We use region growing to decompose a digital puzzle image. Before performing the region growing process, we choose proper locations to place the seeds (denoted as SD_i), which are the starting points of every puzzle piece for region growing. Each yellow point shown in Figure 2.8, which is the center of each puzzle piece, represents the location of a seed SD_i of a digital puzzle image. An example of the decomposed digital puzzle piece image (denoted as PPI) is illustrated in Figure 2.9. A flowchart of the proposed digital puzzle image decomposition process is shown in Figure 2.10.

Algorithm 2.2 : Digital puzzle image decomposition process for information sharing.

Input: A digital puzzle image denoted as I , a puzzle piece size (denoted as PPS), and a number (denoted as $ShareNum$) of information sharing participants.

Output: Digital puzzle piece files.

Steps:

- Step 1 Sprinkle the SDs in I .
- Step 2 Create an “empty” PPI for each SD_i , and let the width and height of the PPI_i be twice as big as those of PPS .
- Step 3 Apply the region growing technique on each SD_i in the following way.
 - 3.1 Find out the RGB value and the coordinates of each “grown pixel” in I .
 - 3.2 “Draw” on each PPI_i those pixels with the same RGB values around each “grown pixel” in I .
 - 3.3 Terminate the region growing process when a pixel with the RGB value of (0, 0, 0) is detected.

Step 4 Randomly divide the $PPIs$ into several groups, with the number of groups being equal to the number $ShareNum$.

In Step 3, the proposed region growing technique is based on checking the RGB values of the pixels in the digital puzzle image I . As mentioned in Section 2.2.2.1, every pixel with the RGB value of (0, 0, 0) in the input image is replaced with the RGB value of (2, 2, 2) before performing the digital puzzle image creation process. Therefore, there will be no pixel with the RGB value of (0, 0, 0) except at the boundary lines of each digital puzzle piece in I .

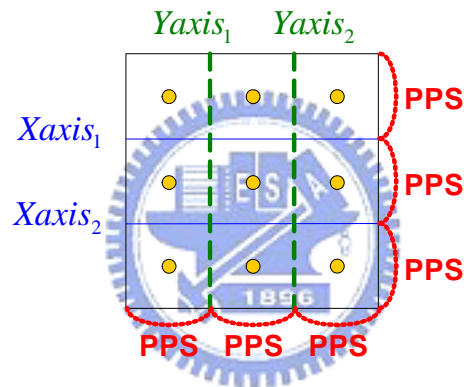


Figure 2.8 The yellow points represent locations of seeds for region growing in a digital puzzle image.

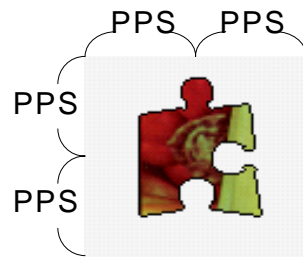


Figure 2.9 An illustration of a decomposed digital puzzle piece image. The width and height of the digital puzzle piece (denoted as PPI) are twice as big as those of the puzzle piece size (denoted as PPS) input by users.

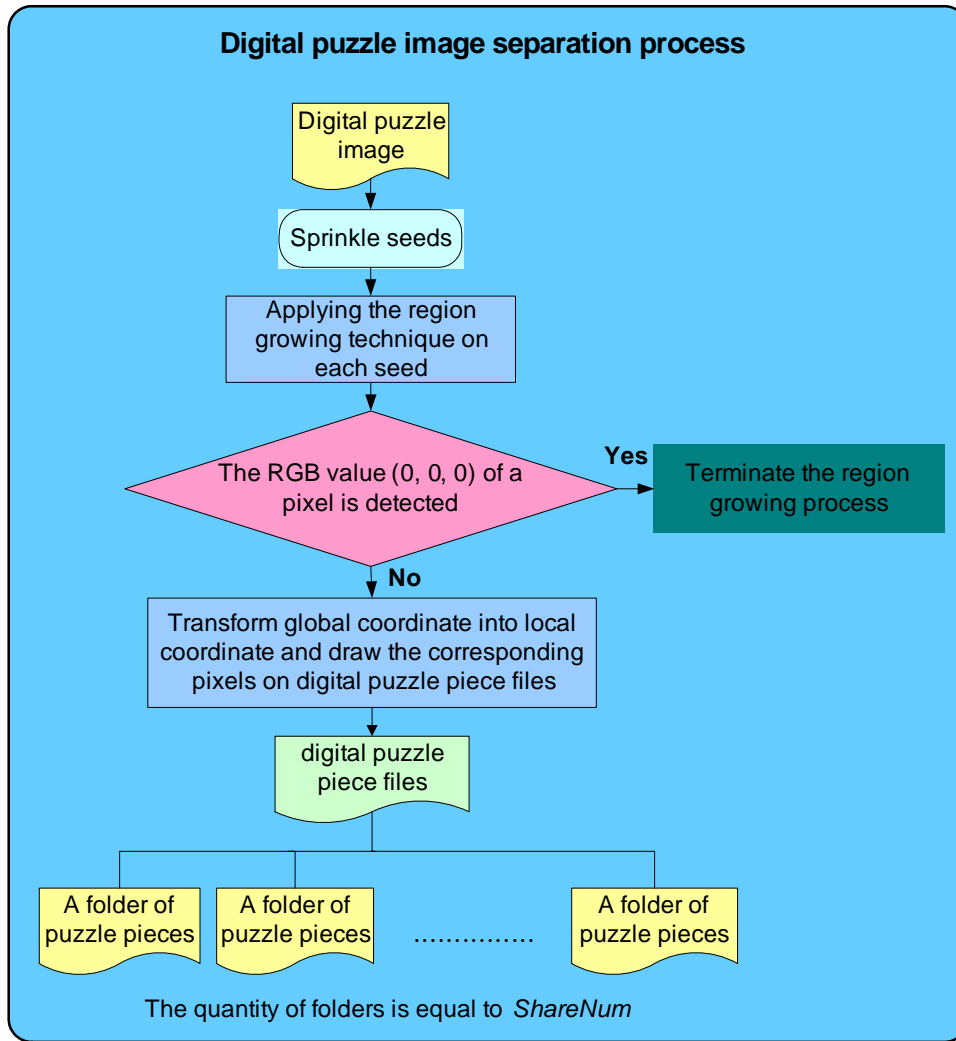


Figure 2.10 The flowchart of the proposed digital puzzle image decomposition process.

2.3 Proposed Digital Puzzle image Reconstruction Process

2.3.1 Idea of Reconstruction Process

As mentioned in Section 2.1, the concept of the proposed automatic digital puzzle image reconstruction process is based on searching the puzzle pieces for correct shapes. We scan the four sides of a digital puzzle piece to check if it is an

indent, an *outdent*, or a *flat* side. We define an *outdent* as a region that touches the convex hull of the piece and has a “neck” that interlocks with a neighboring piece. Therefore, the digital puzzle piece like that shown in Figure 2.11(a) will be classified as having two *indent* sides, one *outdent* side, and one *flat* side.

Also we derive a *puzzle piece region map* (denoted as *PM*) from its corresponding digital puzzle piece file. *PM* indicates whether a pixel is inside the boundary of a digital puzzle piece or not, and a diagrammatic explanation is illustrated in Figure 2.11(b). A *black pixel* (denoted as *BP*) of *PM* represents that the pixel is inside the boundary of its corresponding digital puzzle piece, and a *white pixel* (denoted as *WP*) represents that the pixel is outside the boundary of its corresponding digital puzzle piece. Each of the *PM* has four sides, and we denote them as *N*, *E*, *W*, and *S*, respectively.

A digital puzzle piece has four *primary neighbors* (denoted as *north*, *east*, *west*, and *south*). Pieces interlock with their primary neighbors by *tabs*, consisting of an *indent* in one piece mating with an *outdent* in its neighbor.



Figure 2.11 An example of a decomposed puzzle piece. (a) *N*, *E*, *W*, and *S* represent four sides of a digital puzzle piece. (b) The puzzle piece region map (*PM*) derived from (a).

2.3.2 Detail of Reconstruction Process

2.3.2.1 Digital Puzzle Pieces Orientation Detection Process

By performing this process of orientation detection, we derive the *attributes* of the four sides of each digital puzzle piece. That is, we can figure out if the attribute of a side is *indent*, *outdent*, or *flat*. We assign a value “-1” to an indent side, a value “1” to an outdent side, and a value “0” to a flat side, respectively. We define the value “-1,” “1,” and “0” as the *orientation value* (denoted as *OV*) of the side of a *PM*.

Algorithm 2. 3: Puzzle pieces orientation detection process.

Input: Digital puzzle pieces and their corresponding *PMs*.

Output: The orientation values (denoted as *OVs*) of the four sides (denoted as *N*, *E*, *W*, and *S*, respectively) of each *PM_i*.

Steps:

Step 1 Find the width of the *PM* (denoted as *PMW*).

Step 2 Perform a raster scan of each *PM_i*. Referring to Figure 2.12, let point *o* represent the center of the *PM*, and scan the image from point *o* to point *a*, from point *o* to point *c*, from point *o* to point *b*, and from point *o* to point *d*, respectively.

Step 3 Referring to Figure 2.12, calculate the number *BP* of black pixels in each scanning line (\overline{oa} , \overline{ob} , \overline{oc} , and \overline{od}), respectively, and denote the value of *BP* as the orientation pixel number (denoted as *OPN*).

Step 4 Find the *OVs* of *N*, *E*, *W*, and *S*, respectively, of every input *PM_i* in the following way.

4.1 If $OPN < \frac{PMW}{4}$, set *OV* as -1.

4.2 If $\frac{PMW}{4} - 2 < OPN < \frac{PMW}{4} + 2$, set *OV* as 0.

4.3 If $OPN > \frac{PMW}{4}$, set *OV* as 1.

In addition, the image height of a PM (denoted as PMH) is equal to the image width of a PM (denoted as PMW), and the value of each of \overline{oa} , \overline{ob} , \overline{oc} , and \overline{od} is equal to $\frac{PMW}{2}$. A flowchart of Step 4 is shown in Figure 2.13.

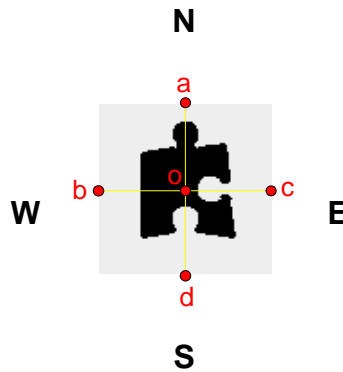


Figure 2.12 The yellow lines represent the scanning ranges of a puzzle piece.

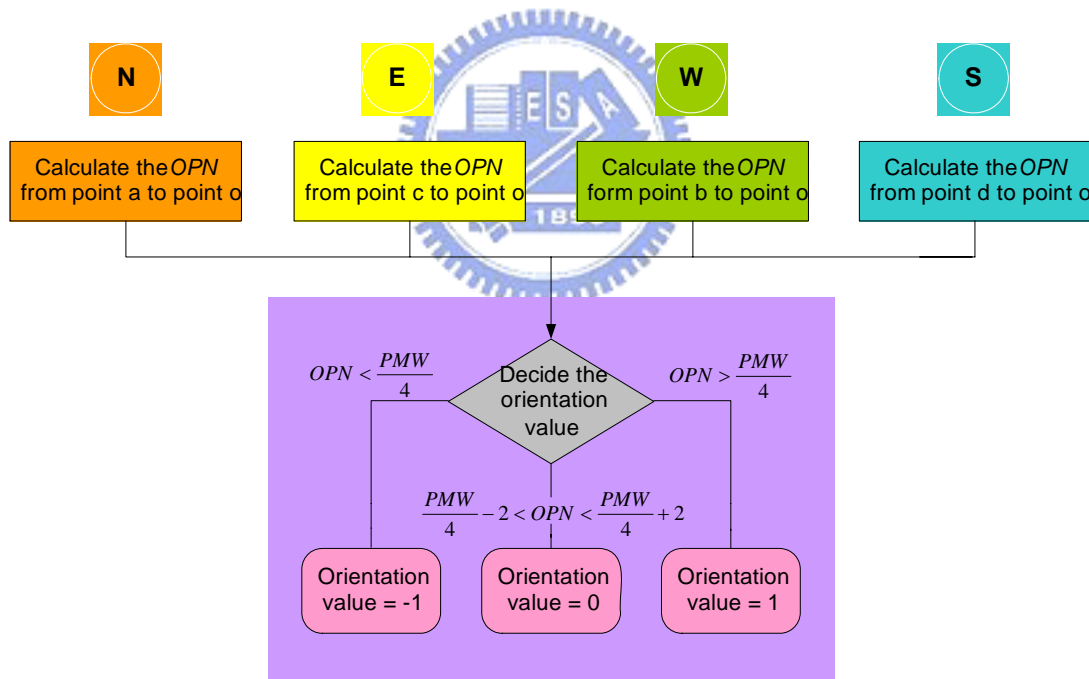


Figure 2.13 The flowchart of the orientation value detection process, where the value of the BPs are denoted as OPN .

2.3.2.2 Digital Puzzle Piece Overlapping and Flawing Detection Process

After performing the orientation detection process described in Section 2.3.2.1, we only roughly mate the digital puzzle pieces by accessing the orientation values of them. We will encounter further two kinds of problems while we try to mate the digital puzzle pieces, namely, the problems of *overlapping* and *flawing of puzzle pieces*. A diagrammatic explanation is shown in Figure 2.14.

Figure 2.15 shows the four directions of the scanning regions of a digital puzzle piece, and Figure 2.16 shows the four primary neighbors of a digital puzzle piece illustrated in Figure 2.15. The scanning regions are framed by the blue rectangles, and we denote the pixels within the scanning regions in the top, bottom, left, and right rectangles as $PMPup_i$, $PMPdown_i$, $PMPleft_i$, and $PMPright_i$, respectively. Figure 2.17 illustrates the situation of composing the digital puzzle pieces as shown in Figure 2.15 (a) and Figure 2.16(a), (b), (c), and (d).

If we want to make sure whether a pair of complementary pieces really match or not, we should perform additionally a digital puzzle piece overlapping detection process to figure out a *composition Boolean value* (denoted as CBV), which indicates whether two scanned PMs really match or not. We define CBV to be “0” when the two scanned PMs really match each other; and CBV to be “1” when we encounter a puzzle piece *overlapping* or *flawing* problems while mating the two PMs .

Take the digital puzzle pieces as shown in Figure 2.15(a) and Figure 2.16(a) as an example. If the detection results of $PMPup_i$ and $PMPdown_i$ are both BPs , it means that there is an “overlapping pixel” in the digital puzzle image. If the results of $PMPup_i$ and $PMPdown_i$ are both WPs , it means that it is a “flawing pixel” of the digital puzzle image. A flowchart of checking the CBV between the digital puzzle pieces shown in Figure 2.15(a) and Figure 2.16(a) is illustrated in Figure 2.18, and the detailed process is described as an algorithm below.

Algorithm 2.4: The digital puzzle pieces overlapping and flawing detection process.

Input: Digital puzzle pieces and their corresponding *PMs*.

Output: *Composition Boolean value* (denoted as *CBV*).

Steps:

Step 1 Detect a digital puzzle piece as shown in Figure 2.15 and its four primary neighbors as shown in Figure 2.16(a), (b), (c), and (d) by performing a raster scan on their corresponding *PMs*.

Step 2 Decide the *CBV* by calculating the number of overlapping and flawing pixels.

2.1 Perform a raster scan of the corresponding *PMs* of Figure 2.15(a) and Figure 2.16(a), that is, detect $PMPup_i$ of Figure 2.15(a) and $PMPdown_i$ of Figure 2.16(a), respectively, to check whether they are *black pixels* (denoted as *BPs*) or *white pixels* (denoted as *WPs*).

2.2 Compare the detected result of each $PMPup_i$ of Figure 2.15(a) with the detected result of each $PMPdown_i$ of Figure 2.16(a).

2.3 Calculate the total number of the flawing and the overlapping pixels.

2.4 Denote the *CBV* of the two detected puzzle pieces as “0” if the total number of pixels derived in Step 2.4 is smaller than two.

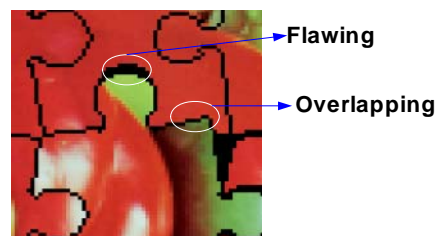


Figure 2.14 An illustration of flawing and overlapping problems of the digital puzzle image reconstruction process.

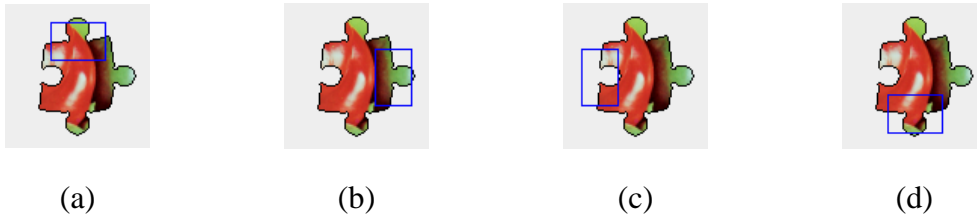


Figure 2.15 The scanning regions of four sides of a digital puzzle piece.



Figure 2.16 The scanning regions of the four primary neighbors of the digital puzzle piece as shown in Figure 2.15. (a) The north neighbor. (b) The east neighbor. (c) The left neighbor. (d) The south neighbor.

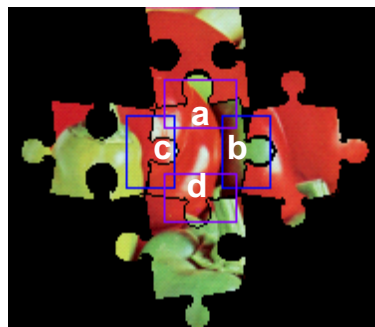


Figure 2.17 The situation of combining the digital puzzle pieces shown in Figure 2.15 and Figure 2.16.

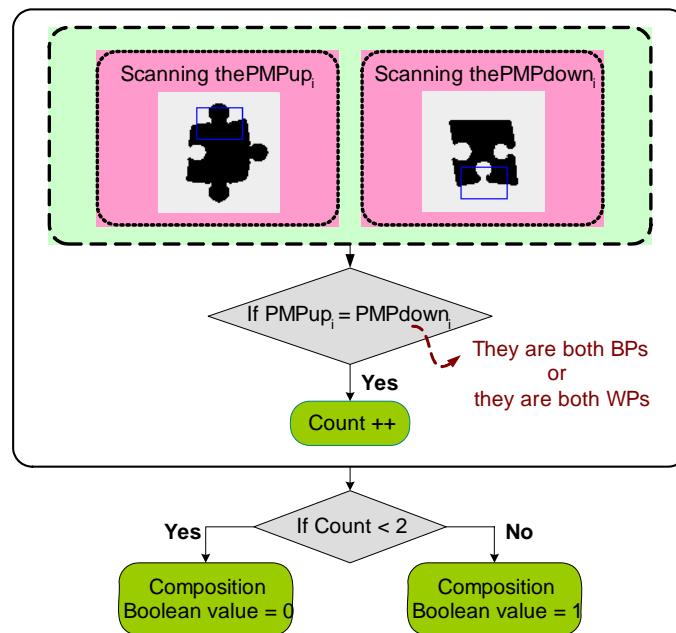


Figure 2.18 A flowchart of checking the CBV of Figure 2.15(a) and Figure 2.16(a).

2.3.2.3 Digital Puzzle Pieces Searching and Reconstruction

Process

In the process of digital puzzle piece search and reconstruction process, we apply the digital puzzle piece orientation detection process and the digital puzzle piece overlapping detection process described in Section 2.3.2.1 and Section 2.3.2.2. We also implement the proposed reconstruction process by applying a depth-first search (*DFS*) algorithm.

Digital puzzle pieces should be located within their proper grid of the digital puzzle image in a zig-zag order, as illustrated in Figure 2.19. We randomly pick a *PM* from a *pool* to check if it can mate the latest located *PM* or not. The *pool* is a temporary place for the non-located *PMs* to dwell in.

A flowchart of an example of applying the following algorithm is illustrated in Figure 2.22 in Section 2.4.



Algorithm 2.5: Digital Puzzle pieces searching and reconstruction process.

Input: The folders of digital puzzle pieces and their corresponding *PMs*.

Output: A reconstructed digital puzzle image.

Steps:

Step 1 Read in the *PMs*, and find the original *HParts* and *VParts* of the digital puzzle image to be reconstructed (denoted as *I'*) in the following way.

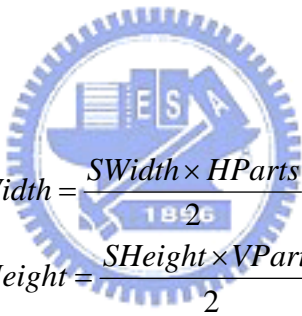
1.1 Check *N*, *E*, *W*, and *S* of each *PM_i*.

1.2 Find the *PMs* whose orientation value (denoted as *OV*) of *N* is “0” (which means *N* is a flat side), and calculate the number of this kind of *PMs*.

- 1.3 Find the *PMs* whose *OV* of *W* is “0” (which means *W* is a flat side), and calculate the number of this kind of *PMs*.
 - 1.4 Take the numbers of flat sides of *N* and *W* as the values of *HParts* and *VParts* of the digital puzzle image *I'*.
- Step 2 Find the exact digital puzzle piece which should be placed on the most-west-north position (denoted as *Position₁*) of *I'*, where the location of *Position_i* is illustrated in Figure 2.19.
- 2.1 Check *N*, *E*, *W*, and *S* of each *PM_i*.
 - 2.2 Find the exact *PM* whose *OVs* of *N* and *W* are both “0”, and place this *PM* on *Position₁*.
- Step 3 Utilize an index to point toward *Position_i*, where the last puzzle piece is located. At this moment, the index is pointed to *Position₁*.
- Step 4 Deal with the first row of the *I'* in the following way.
- 4.1 Assign the value “1” to *i*.
 - 4.2 Push proper *PMs* into a stack in *Position_{1+i}*.
 - 4.2.1 Randomly pick a *PM* from a pool, and make sure that its *OV* of *N* is “0.”
 - 4.2.2 Push the lastly picked *PM* into a stack in *Position_{1+i}*, if the sum of the *OV* of *E* of the lastly located *PM* and the *OV* of *W* of the lastly picked *PM* is “0” and their *CBVs* are “0.”
 - 4.2.3 Terminate this step if all of the *PMs* in the pool are checked.
 - 4.3 Pop out a *PM* from the stack in *Position_{1+i}*, and place it on *Position_{1+i}*. Make the index to point to *Position_{1+i}* now.
 - 4.4 Update *i* to be “*i*+1,” and repeat Steps from 4.2 to 4.3. When the stack is empty, tread back the pointer and update *i* to be “*i*-1.”

Step 5 Deal with all the other rows of I' in a way similar to Step 4, but with the position of the lastly located PM and its primary neighbors being different.

In Step 1, we perform the digital puzzle image decomposition process described in Section 2.2.2.2, utilize the input value of the digital puzzle piece size (denoted as PPS) to decompose a digital puzzle image, and let the image width and height of a decomposed digital puzzle piece image be twice as big as the input value of PPS . Therefore, we can find out the image width and height of a recovered digital puzzle image by checking the input digital puzzle piece files and applying Formula (2.19) below, where we denote the width and height of the digital puzzle image as “ $Width$ ” and “ $Height$,” and denote the width and height of a digital puzzle piece file as “ $SWidth$ ” and “ $SHeight$.”



$$\left\{ \begin{array}{l} Width = \frac{SWidth \times HParts}{2}; \\ Height = \frac{SHeight \times VParts}{2}. \end{array} \right. \quad (2.19)$$

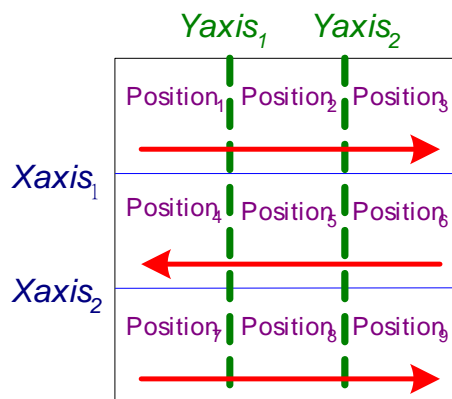


Figure 2.19 The i of $Position_i$ indicates the order of locating the digital puzzle pieces.

2.4 Experimental Results and Discussions

Figure 2.20 is an experimental result of applying the digital puzzle image creation process described in Section 2.2.2.1 to an input digital image. Figure 2.20(a) is the input image and Figure 2.20(b) is the final result. We can find that the digital puzzle pieces in Figure 2.20(b) have many kinds of shapes. In principle, according to the proposed creation algorithm, the probability for two *frame puzzle pieces* with the same shape to appear is $1/(3 \times 2 \times 2 \times 2 \times 3)^3$, and the probability for two *interior puzzle pieces* with the same shape to appear is $1/(3 \times 2 \times 2 \times 2 \times 3)^4$. The reason is that the shape of each side of this kind of digital puzzle piece is decided by the parameters *UpDown*, *BigR*, θ , and the two coordinates of *Cross*.

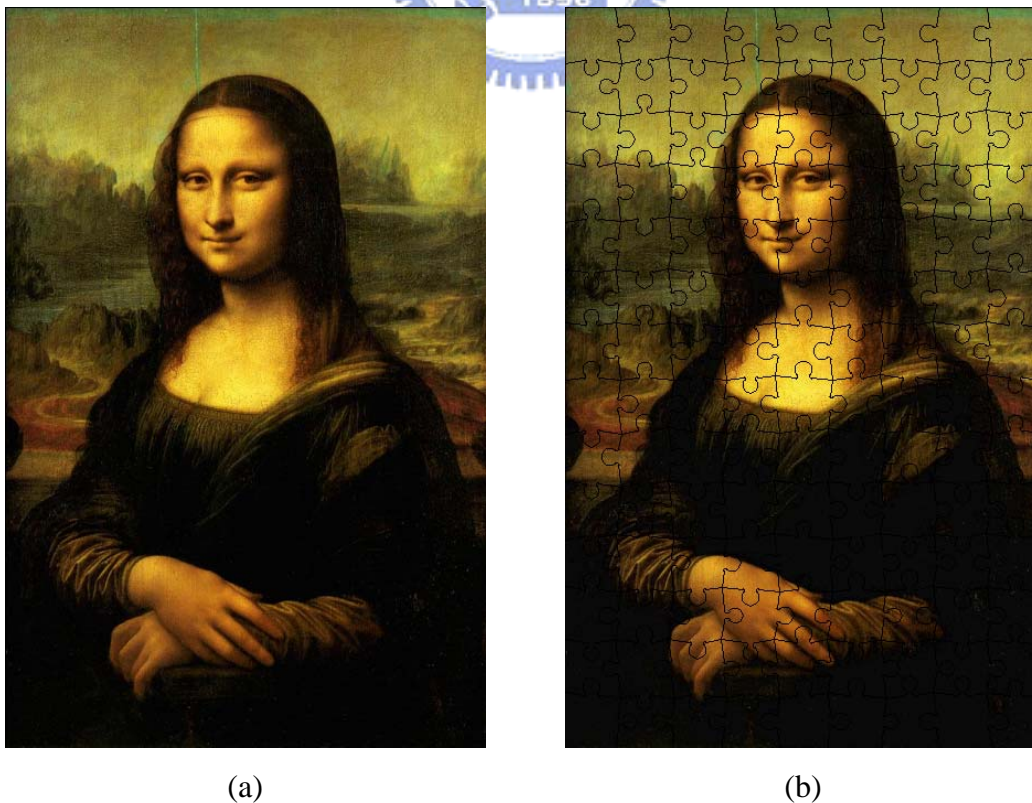


Figure 2.20 Experimental results. (a) An input digital image. (b) A digital puzzle image of (a).

Figure 2.21 is an experimental result of applying the digital puzzle image decomposition process described in Section 2.2.2.2 to a digital puzzle image as shown in Figure 2.20(b). If there are four information-sharing participants, Figure 2.20(b) will be randomly divided into four folders of digital puzzle pieces as shown in Figure 2.21(a) through (d). Each participant of information sharing can get one folder of them.

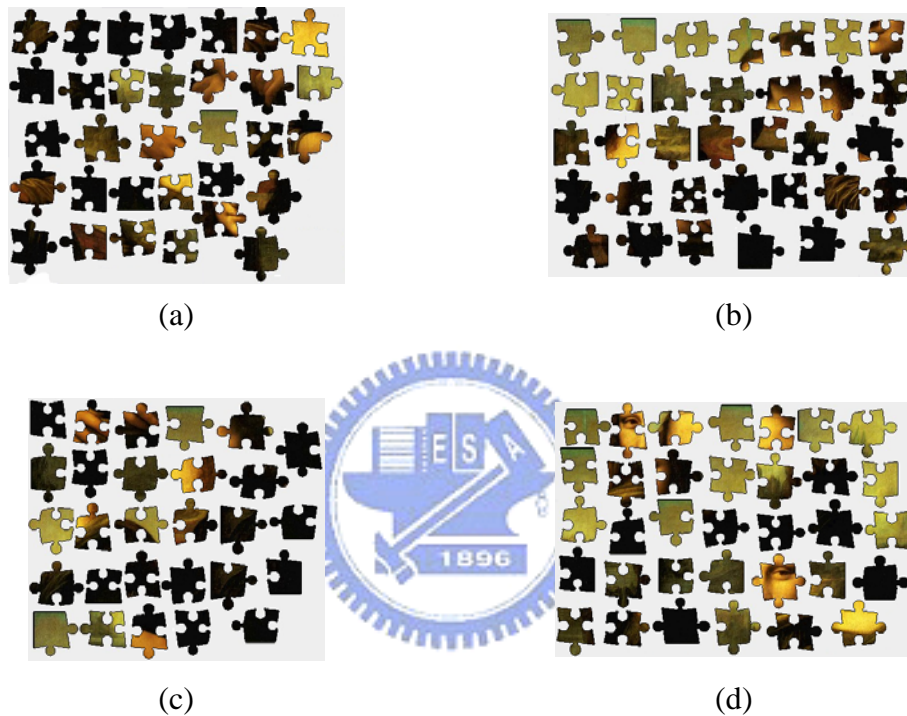


Figure 2.21 (a), (b), (c), and (d) represent four folders with digital puzzle pieces in it, and each participant can get one of them.

By applying the digital puzzle image reconstruction process described in Section 2.3 to deal with the decomposed digital puzzle pieces as shown in Figure 2.21, we can get a recovered digital puzzle image as shown in Figure 2.22(k), and this image is exactly the same as Figure 2.20(b).

After detecting all of the puzzle pieces as shown in Figure 2.21, we can figure out that the original digital puzzle image is a “14×9” puzzle, and start to reconstruct it. In Figure 2.22(a), we show the found most-west-north puzzle piece. In Figure 2.22(b), we deal with the first row of the digital puzzle image. In Figure 2.22(c), we

show the found most-east-north puzzle piece. In Figure 2.22(d), we show the found first puzzle piece of the even row. In Figure 2.22(e), we deal with the even row. In Figure 2.22(f), we show the found last puzzle piece of the even row. In Figure 2.22(g), we show the found first digital puzzle piece of the odd row. In Figure 2.22(h), we deal with the odd row. In Figure 2.22(i), we show the found last puzzle piece of the odd row. In Figure 2.22(j), we repeat the steps for Figure 2.22(d) through (i). Finally, we can reconstruct the digital puzzle image successfully as shown in Figure 2.22(k).

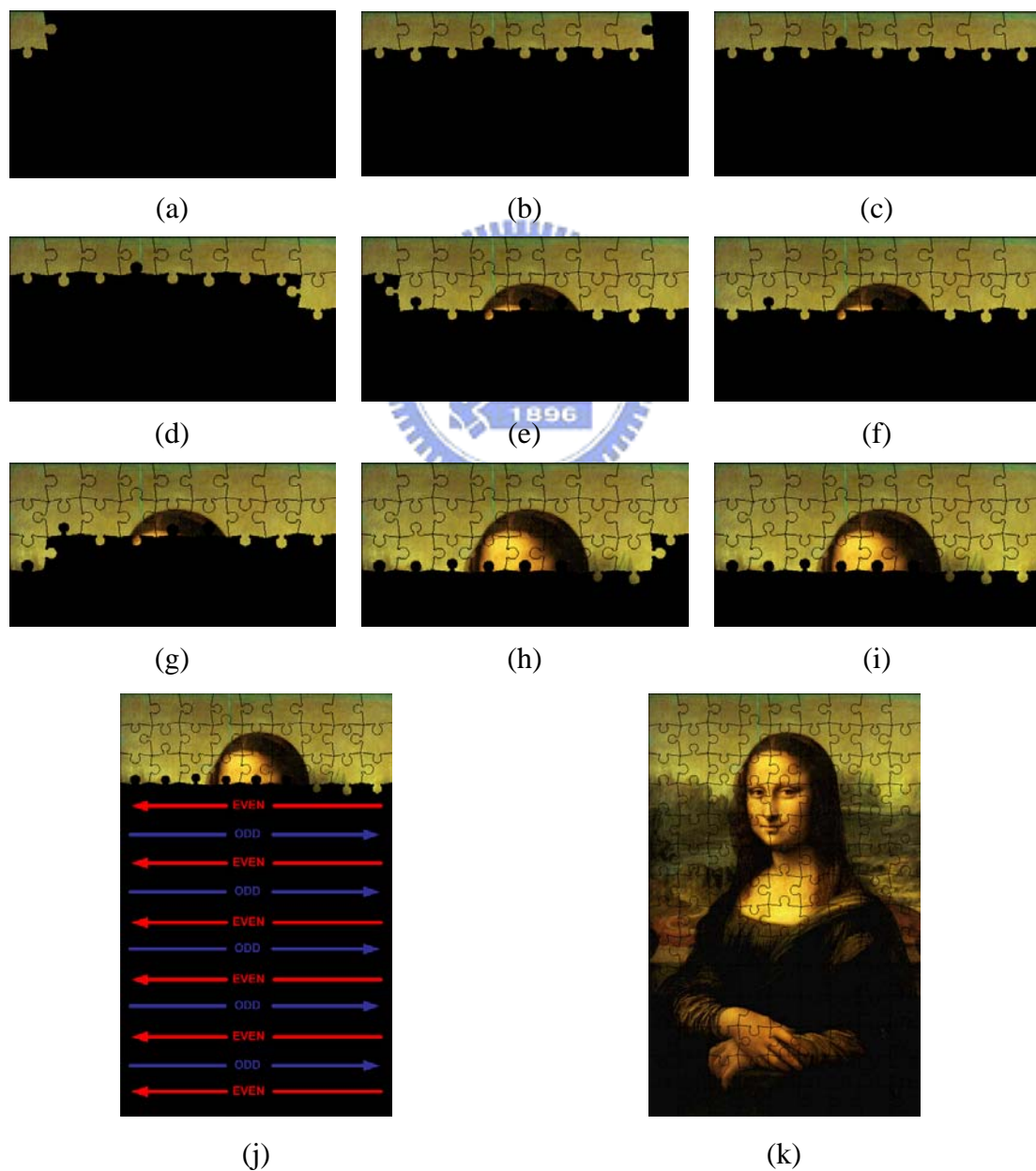


Figure 2.22 The results of the intermediate steps of digital puzzle image reconstruction process.