國立交通大學

多媒體工程研究所

碩 士 論 文

半自動化拼圖系統

Semi-Automatic Puzzle Solver

研 究 生：林厚邑

指導教授：陳玲慧　教授

中 華 民 國 一百零一 年 七 月

半自動化拼圖系統
Semi-Automatic Puzzle Solver

研 究 生：林厚邑　　Student：Hou-I Lin

指導教授：陳玲慧　　Advisor：Ling-Hwei Chen

國 立 交 通 大 學

多媒體工程研究所

碩 士 論 文

A Thesis

Submitted to Institute of Multimedia Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

July 2012

Hsinchu, Taiwan, Republic of China

中華民國一百零一年七月

半自動化拼圖系統

研究生：林厚邑　　指導老師：陳玲慧 博士

國立交通大學多媒體工程研究所

摘要

本論文中，我們提出一個全新的半自動化拼圖系統，這個系統針對任何型態的拼圖進行還原，並且不受該拼圖的特性所限制，例如拼圖碎片的形狀、相鄰拼圖碎片的數目…等等。我們的系統利用拼圖碎片的形狀和顏色資訊來還原拼圖碎片，大致上，可以分成四個步驟。第一，從輸入影像中擷取出拼圖碎片。第二，將每一個拼圖碎片的形狀和顏色資訊擷取出來，並且利用這些資訊來建立形狀特徵序列 (shape feature string)和顏色特徵序列 (color feature string)。第三，針對每兩片拼圖碎片，利用形狀特徵序列和顏色特徵序列來找出能將這兩片拼圖碎片接合在一起的可能邊界，並計算這個邊界的相似度。第四、根據上個步驟求出來的相似度，將兩片具有最高相似度的拼圖碎片接合在一起，我們不斷地重複此步驟，直到該拼圖被完整還原。而在第四步驟當中，我們提供了一個使用者決策功能，讓使用者決定，每一次被挑選出來的拼圖碎片是否確實可以被接合在一起。藉由這個功能，我們就能完整的將拼圖碎片還原成原始拼圖。

我們使用真實世界的拼圖和人為產生的拼圖對我們的系統進行測試。實驗結果證明我們的系統可以有效的將任何型態的拼圖還原成原始圖片。

Semi-Automatic Puzzle Solver

Student: Hou-I Lin Advisor: Dr. Ling-Hwei Chen

Institute of Multimedia Engineering

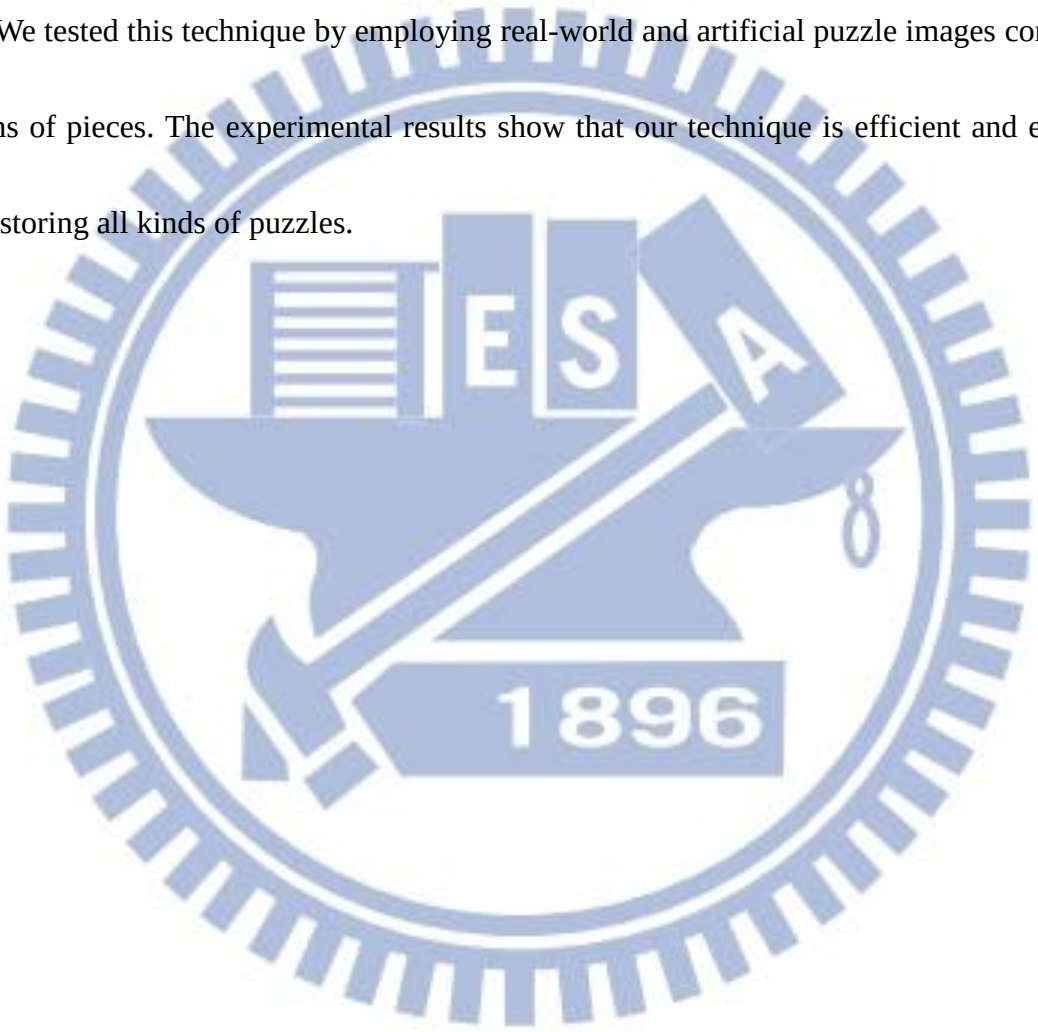
National Chiao Tung University

Abstract

This thesis proposes a new technique on puzzle solving. The novelty of the proposed technique is that it provides a semi-automatic puzzle solution which can reconstruct all kinds of puzzles without any initial restriction about the shapes of pieces, the number of neighbor pieces, etc. This technique uses both shape and color information to cope with the puzzle solving problem and consists of four steps. First, puzzle pieces are extracted from the input image. Second, shape and color features are extracted from each puzzle piece to form the shape and color feature strings. Third, the possible common boundary of each pair of puzzle pieces is found and its similarity measure is estimated by using their shape and color feature strings. Fourth, based on the similarity measure of each pair of pieces, a recurrent procedure is applied; it merges puzzle pieces with the largest similarity measure in pair until the original puzzle image is reformed. In the merging step, we provide a simple user interactive method, which allows a user to decide if each pair of pieces selected by the solver is adjacent or not.

Our solver will select another pair of puzzle pieces which may be adjacent as soon as the user decides previous selected pieces are not adjacent pieces. This method makes our solver be able to restore puzzle pieces completely. And a user can easily reconstruct a puzzle through our puzzle solver.

We tested this technique by employing real-world and artificial puzzle images containing dozens of pieces. The experimental results show that our technique is efficient and effective for restoring all kinds of puzzles.



誌　謝

這篇論文的完成，首先要感謝指導教授 陳玲慧博士。在這兩年碩士生涯中，老師在生活上和課業上給予許多的關心與指導，讓我在這短短的兩年當中，不僅得到學業上的知識，還獲得更多做人處事的道理。除此之外，要感謝口試委員 李建興教授、 石昭玲教授以及 李坤龍博士於口試中給予的指導與建議，使得整篇論文更趨完善。

接著，要感謝實驗室各位一起生活、一起奮鬥的夥伴們，博士班的學長姐，龍哥、超哥、Jimmy、阿和、阿三、盈如、芳如，還有畢業的學長們，阿熹和駿哥，以及同屆的同學，教主和子杰，學弟阿嘉、阿昌、科科，因為有大家的陪伴，讓我兩年的碩士生涯過的充實又豐富。

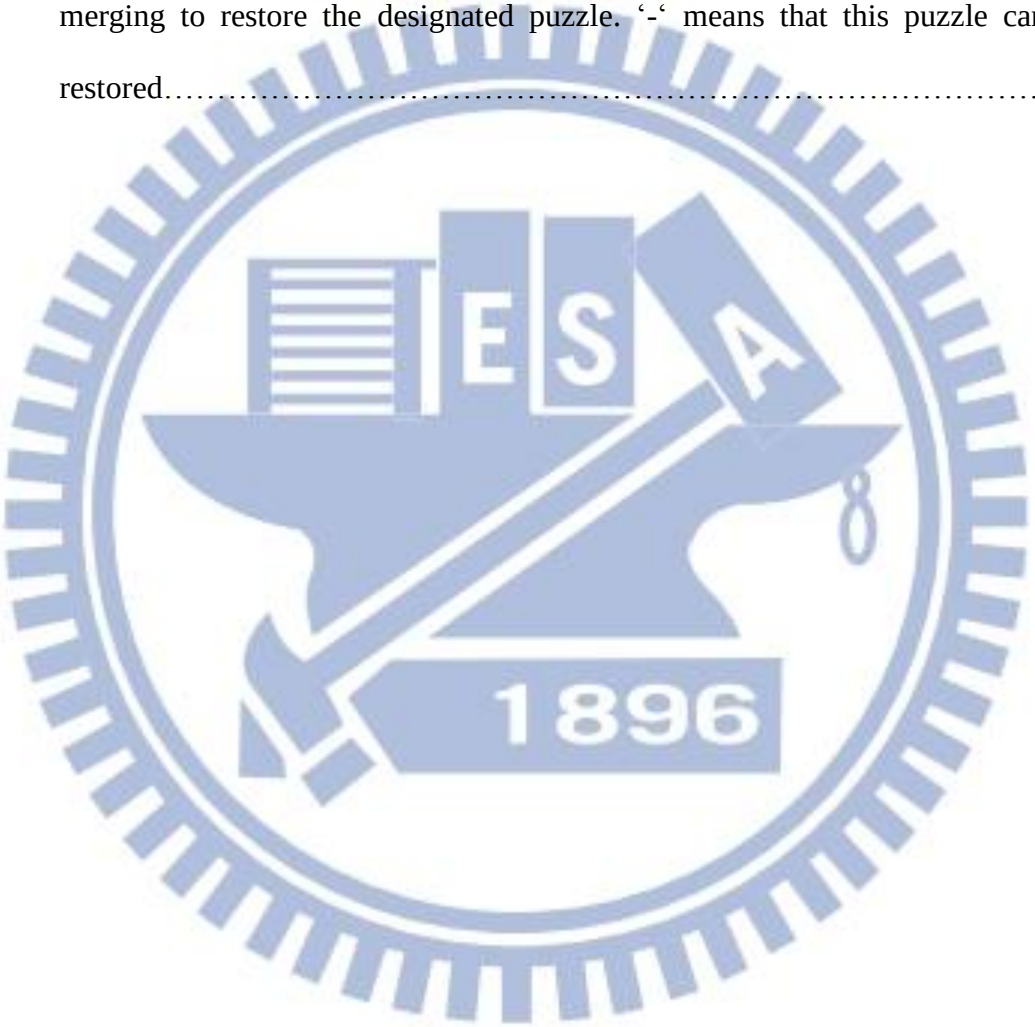
此外，還要感謝我的女朋友張書怡，在我遇到挫折時鼓勵我，感到煩悶時聽我發牢騷，在我怠惰的時候，陪我一起念書，一起改論文。

最後，要感謝我的父母以及哥哥，在我的研究生涯當中，提供各種生活上以及學業上的協助與建議，給予我信心與依靠，我才能順利完成本篇論文。謹以此篇論文獻給你們，以及所有關心我、鼓勵我的朋友，謝謝。

TABLR OF CONTENTS

LIST OF FIGURES

LIST OF TABLES



CHAPTER 1
INTRODUCTION

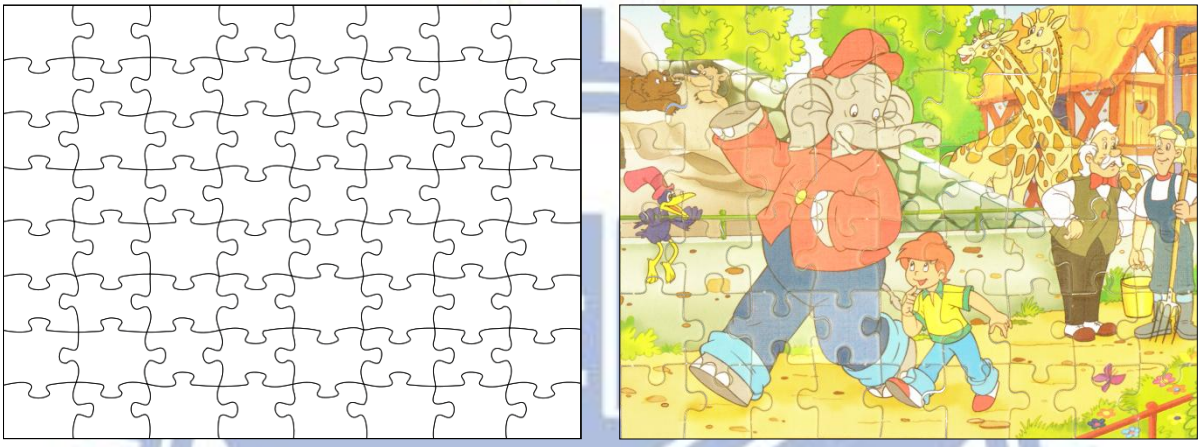
1.1 Motivation

A puzzle solving problem focuses on restoring a picture that is separated into several pieces by placing all pieces back to their original locations accurately. This problem includes many popular issues such as boundary detection, shape matching, and texture comparison, which are of general topics in the fields of computer vision and pattern recognition. In addition to solve jigsaw puzzle problem, the puzzle solving problem could also be applied to other fields such as the reconstruction of archeological artifacts [1-3], the molecular docking problem [4, 5], the speech descrambling [6], and the reconstruction of ripped-up documents [7].

The research on puzzle solving started since 1964 [8]. Most of the related works to solve puzzle were discussed under certain puzzle conditions like picture outfit existence, a specific number of neighbor fragments for each piece, and particular outfit shape of the target picture, etc. However, the applications of the puzzle solving problem in the real world will not be limited by these conditions. Therefore, we attempt to propose a solver that is able to handle all kinds of puzzles without limitations.

1.2 Related work

Generally, jigsaw puzzles can be classified into two classes. One has only shape information like Fig. 1(a). The other has both shape and color information like Fig. 1(b). Early researches mainly focus on solving the first class of puzzles.



(a) (b)

Fig. 1. Two various patterns of jigsaw puzzles. (a) Puzzle with only shape information. (b) Puzzle with both shape and color information.

In 1964, the idea to solve the Jigsaw puzzle problem automatically was firstly brought up by Freeman and Gardner [8]. They use partial boundary curve matching and some heuristics to handle this problem without referring to the original picture. Their work has become the foundation of many subsequent papers. Radack and Badler [9] in 1982 also use partial boundary curve matching, but they employ the polar coordinate system to take the place of the Cartesian coordinate system. Wolfson et al. [10] in 1988 proposed an algorithm, which uses the Schqartz-Sharir curve matching algorithm and optimized search trees, to solve the puzzle problem up to 104 puzzle pieces with two specific patterns. Based on the algorithm

proposed by Wolfson et al. [10], Goldberg et al. [11] tries to find indents, outdents, and straight sides of each piece, and then a global matching algorithm is provided to restore the puzzle picture. Their method can handle up to 204 puzzle pieces with different shapes, but it is still limited by the existence of picture outfit which will reduce the complexity on puzzle solving. All of the aforementioned methods only use shape information without color information.

Few researches deal with the puzzle solving problem using both shape and color information. Kosiba et al. [12] in 1994 firstly proposed an algorithm related to shape and color information of pieces. Their method can handle puzzles up to 54 colorful puzzle pieces with satisfactory results. In 1998, Chung et al. [13] use the distance from a point on boundary curves to the line determined by two neighboring corner points for shape matching, and they employ the local image features of a small region along the edge to calculate the similarity measure. This work is useful while handling puzzles under 54 colorful puzzle pieces. The idea of using both shape and color information was further used by Yao and Shao [14] in 2003. Their technique combines shape and image matching with a cyclic “growth” process, and it uses the integration degree on subdivided strips all along the edge. It can handle real-world images with dozens of jigsaw pieces. Nielsen et al. [15] in 2008 proposed a new method, which is the first work to solve the puzzle problem using only color information. This method calculates the similarity measure for each pair of puzzle pieces along their edges, and based

on the similarity measure, they restore the puzzle picture. At the process of merging pieces, the puzzle solving problem is split into two sub-problems of solving boarder and interior pieces. This actually reduces the complexity of the puzzle solving problem. This method can handle up to 320 puzzle pieces created by computers with a specific picture, which is the largest puzzle solved to date by a computer, and it tried to cope with two puzzles, which are real-world puzzles scanned by the scanner, of 24 and 54 pieces, but the puzzle of 54 pieces was not completely reconstructed by this method.

Though all the aforementioned works can handle puzzle solving problem with small puzzle pieces, but their methods are just confirmed on few specific pictures, and still limited by common puzzle conditions such as picture outfit existing, specific neighbor fragment number of each piece, and particular shapes of some specific pictures, which can reduce the complexity of puzzle solving problems.

The most recent work was proposed by Makridis and Papamarkos [17] in 2010. They properly handled the puzzle solving problem without using common puzzle conditions. Their method uses the IPAN99 algorithm [18] to find the characteristic points in each piece, and Kohonen self-organized feature map neural network technique [19] is used to reduce color information into ten dimensions. After that, each pair of pieces is examined and the most similar pair of pieces is merged. The merging procedure is repeated until all pairs are examined or only one piece is left. Although it can handle more complex puzzles than

previous works, but it is unable to handle regular puzzle pieces such as rectangular pieces or canonical jigsaw puzzles.

Table 1 summarizes the aforementioned works on handling the puzzle solving problem. Currently, there is no efficient way to cope with all kinds of puzzles without any common puzzle conditions. Thus, this thesis is dedicated to find out a method to restore all kinds of puzzles without any common puzzle conditions. The proposed method contains four steps:

(1) Extract each puzzle piece from the color input image.

(2) Extract features of each puzzle piece according to its shape and color information.

(3) Find a possible common boundary for each pair of puzzle pieces based on their features extracted in Step (2), and estimate the similarity measure.

(4) Based on the result of Steps (3), find out the pair of pieces with the maximum similarity measure and merge them according to the common boundary. Step (4) is repeated until only one piece is left.

In Step (4), a user interactive method is provided, it allows the user deciding if the pair of puzzle pieces selected by our solver is adjacent or not. If the current pair of puzzle pieces is not considered as adjacent pieces, our solver will immediately find another pair of puzzle pieces. After these four steps, the original puzzle picture can be restored completely.

Table 1. Categorization of methods on handling puzzle solving problems.

Method proposed by	Handle the puzzles with canonical jigsaw	Handle the puzzles with various shape	Use piece shape information	Use piece color information	Use the common puzzle conditions
Wolfson et al. (1988)	Yes	No	Yes	No	Yes
Kosiba et al. (1994)	Yes	No	Yes	No	Yes
Chung et al. (1998)	Yes	No	Yes	Yes	Yes
Goldberg et al. (2002)	Yes	No	Yes	No	Yes
Yao and Shao (2003)	Yes	No	Yes	Yes	Yes
Nielsen et al. (2008)	Yes	No	No	Yes	Yes
Makridis et al. (2010)	No	Yes	Yes	Yes	No
Proposed Method	Yes	Yes	Yes	Yes	No

1.3 Organization of this Thesis

The detail of our method to cope with the puzzle solving problem will be described in Chapter 2, and then the progress and difficulty in our research and the user interactive method are discussed in Chapter 3. In Chapter 4, the experimental results will be displayed. Finally, Chapter 5 makes a conclusion.

CHAPTER 2

THE PROPOSED METHOD

In this chapter, we will describe our proposed method in detail. Fig. 2 shows the block diagram of the proposed method. As mentioned in Chapter 1.2, our method contains four steps:

(1) Puzzle Piece Extraction.

(2) Feature Extraction.

(3) Common Boundary Finding.

(4) Adjacent Pieces Merging.

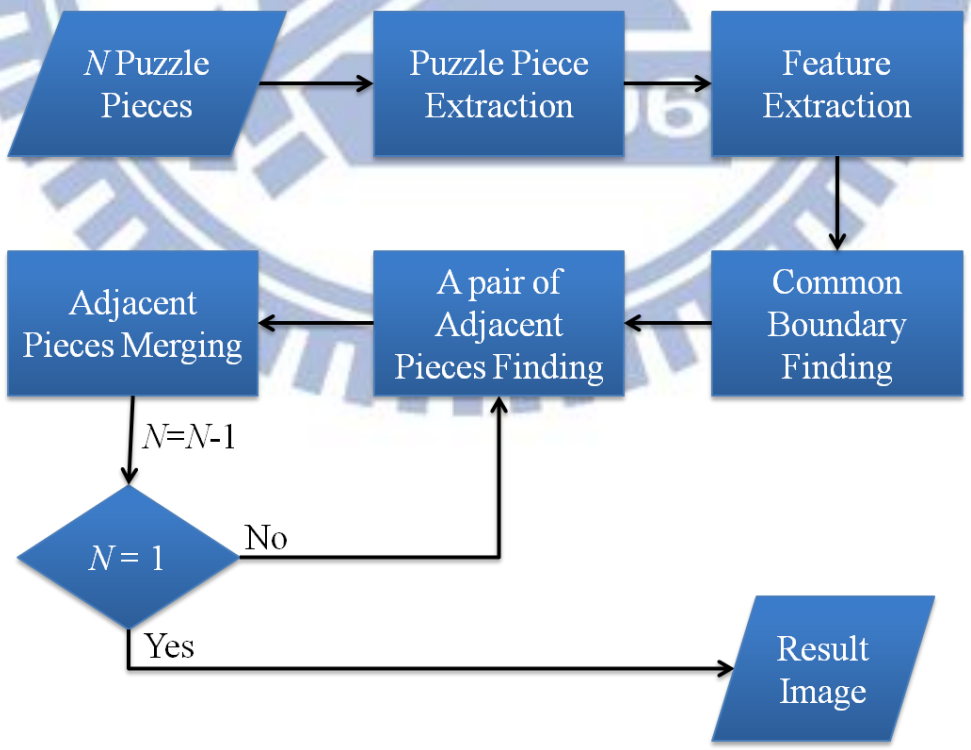


Fig. 2. The block diagram of the proposed method.

2.1 Puzzle Piece Extraction

At the beginning of puzzle solving, a series of images are inputted. Each image contains one puzzle piece (see Fig. 3(a)). In order to extract all puzzle pieces, each puzzle piece is placed over a background layer with a predefined color. This color should be chosen to make the contour of a puzzle piece be extracted easily on the input image; therefore we select the color, excluding those boundary colors, as the background color.

The proposed method is quoted from Makridis and Papamarkos [17]. First, it uses the background color to differentiate the background region from the puzzle piece region. A binary image BI will be built by the following Equation:

$$\mathrm{BI}(x,y)=\begin{cases}0, & \text{if the color on } (x,\ y) \text{ is equal to the background color}\\ 1, & \text{otherwise.}\end{cases} \tag{1}$$

Note that some points on the puzzle piece with background color might be judged as background and some noises on the background might be considered as a part of the puzzle piece (see Fig. 3(b)). To overcome this problem, all connected components on BI are found. Based on the hypothesis that the biggest connected component is either the puzzle piece or the background, all erroneous smaller connected components are reclassified.

More concretely, the input image is classified into two classes: (1) pixels with the background color and (2) all other pixels. Then, all connected components are found for both classes. After that, there will be two classes of objects; one for puzzle piece objects and

another for background ones. The biggest object of each class is considered as either puzzle piece or background. Each smaller object will be reclassified if it lies inside the biggest object, but belongs to the different class. Fig. 3 gives an example to illustrate puzzle piece extraction.

(a) (b) (c)

Fig. 3. Puzzle piece extraction. (a) The input image. (b) Binarization based on the predefined background color. (c) Extracted puzzle piece.

2.2 Feature Extraction

After correctly extracting all puzzle pieces, we will locate the boundary of each puzzle piece. All boundary points extracted are referred as sample points, which are chosen following the boundary of a piece pixel by pixel by selecting the next pixel from the 8-neighbors counterclockwise. However, two sets of sample points corresponding to the common boundary of two adjacent pieces may be different, that is, pixels on the common boundary of one piece are unmatched to those in the common boundary of the adjacent piece (see Fig. 4(a)). The occurrence of the above situation is due to the curvature of the boundary. Here, we

apply the method proposed by Nielsen et al. [15] to solve this problem. Their method uses the edge of the boundary to take the place of pixels on the boundary as sample points (see Fig. 4(b)).

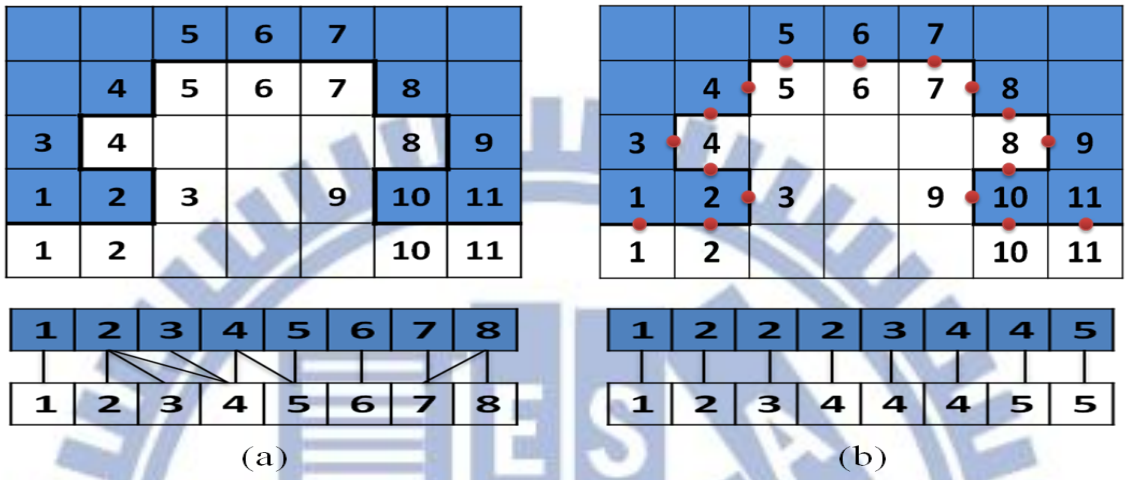


Fig. 4. An example of sample points on two adjacent pieces. (a) Sample points extracted from pixels on the common boundary with black color standing for one piece and white for the other piece. (b) Sample points extracted from the edge of the common boundary with circles denoting sample points.

2.2.1. Shape Feature Extracting

The technique used to extract shape information is an adaptation of Chain code [20] with 8-neighbors. It extracts the direction of each sample point on the boundary as the shape feature. For each sample point X, points P_1 and P_2 are located at the right and left sides of X with the same distance (see Fig. 5(a)). Then, the direction from P_1 to P_2 is considered as the direction of X. Here, we design eight direction codes based on our coding map, which normalizes all directions to eight values 0 ~ 7 (see Fig. 5(b)).

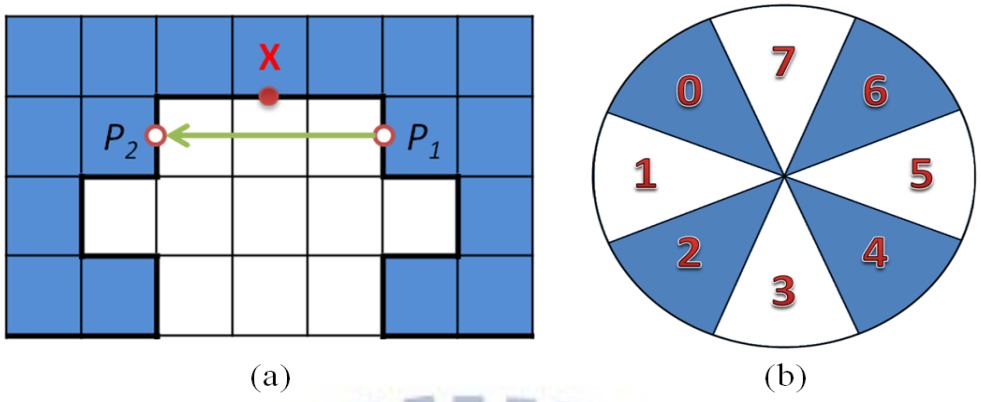


Fig. 5. An example to illustrate direction code extraction. (a) A sample point X with P_1 and P_2 located at its right and left sides. (b) The direction coding map.

After obtaining the direction codes of all sample points on each puzzle piece, we group these codes to form a shape feature string to represent the shape information of the puzzle piece. Note that the proposed method wants to find adjacent pieces, and then these adjacent pieces are merged to form the original image. Each pair of adjacent pieces should have a common boundary. This boundary will correspond to a segment of shape feature string of each piece. The segments representing the common boundary of two adjacent pieces will be similar.

In order to reduce the complexity of finding pairs of similar segments between two puzzle pieces, the run-length coding (RLC) algorithm is used to encode all shape feature strings; the result is called run length coding string (RLCS). However, there are some small runs in RLCS; these will increase the complexity of finding pairs of similar segments. These small runs are produced from those sample points with directions near the boundary of two neighbor direction codes. A technique is provided to eliminate these small runs.

For each RLCS S with $S=(R_1, R_2, ..., R_j, ..., R_m)$, where $R_j=(C_j, L_j)$ represents the jth run with direction code C_j and run length L_j. First, we classify all runs into four categories by threshold T_1, as shown in the following:

$$\begin{aligned}
&\text{Category 1: } L_j \geq T_1,\\
&\text{Category 2: } L_j < T_1 \text{ and } L_{j+1} \geq T_1,\\
&\text{Category 3: } L_j < T_1 \text{ and } L_{j+1} < T_1 \text{ and } (|C_j - C_{j+1}| > 1 \text{ or } |C_j - C_{j+1}| < 7),\\
&\text{Category 4: } L_j < T_1 \text{ and } L_{j+1} < T_1 \text{ and } (|C_j - C_{j+1}| = 1 \text{ or } |C_j - C_{j+1}| = 7).
\end{aligned}$$

Where T_1 is a constant defined by the user. Then, those runs in Category1, Category2, or Category3 will be held. Runs in Category4 will be merged together with their neighbor runs in Category4. This process will be executed until no run in Category4. To treat the above merging process, the number of direction codes will be increased to 16 corresponding to $\{0, 01, 1, 12, 2, 23, 3, 34, 4, 45, 5, 56, 6, 67, 7, 70\}$. Note that the new direction code C_1C_2 is created to represent the direction value of two merged runs with direction codes C_1 and C_2. For the convenience of using these values, we reset these direction codes to 0~15 (see Fig. 6). After that, if there are still small runs in RLCSs, that is, those runs are classified into Category2 or Category3. They will be averagely reassigned to the neighbors on both sides.

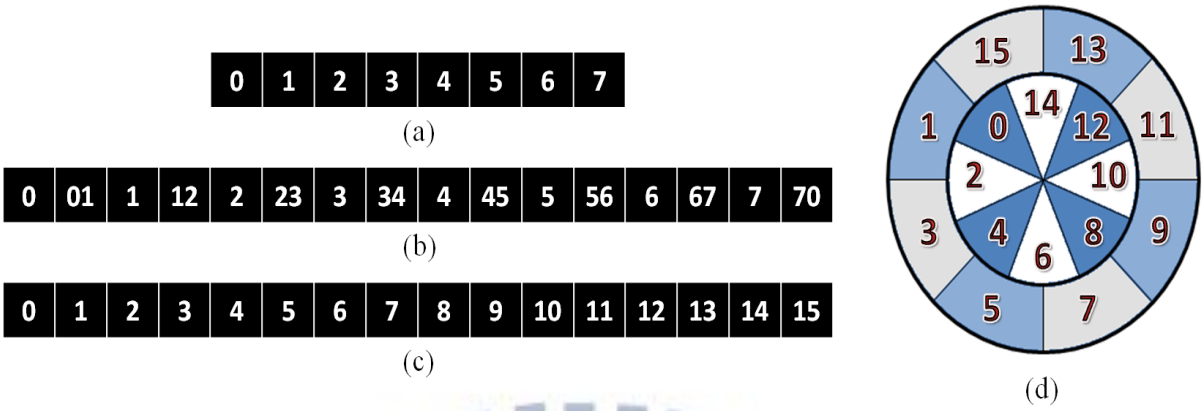


Fig. 6. The new 16 direction codes. (a) Original 8 direction codes. (b) After merging each pair of two neighboring direction codes. (c) The new direction codes. (d) New direction coding map.

2.2.2. Color Feature Extracting

The puzzles discussed here contain not only shape but also color information. In order to enhance the accuracy of reconstructing puzzle pieces, we consider the color feature around the boundary. According to the research of Nielsen et al. [15], using all HSI color components (Hue, Saturation, and Intensity) as the color feature can get the most correct result on restoring puzzle pieces, so we choose the HSI color model to represent colors of an input image. Because shadow exists on the boundary of a puzzle piece, this makes colors of boundary points darker than expected ones. Fig. 7 shows an example. To solve this problem, we take an inside point near a sample point to get a color representing the color of the sample point. Here, a distance D is used for preventing from the shadow effect. Note that D depends on the input image. If the image is produced by the scanner, then D is set to 2. Otherwise, if the image is created by a person, then D is set to 0.

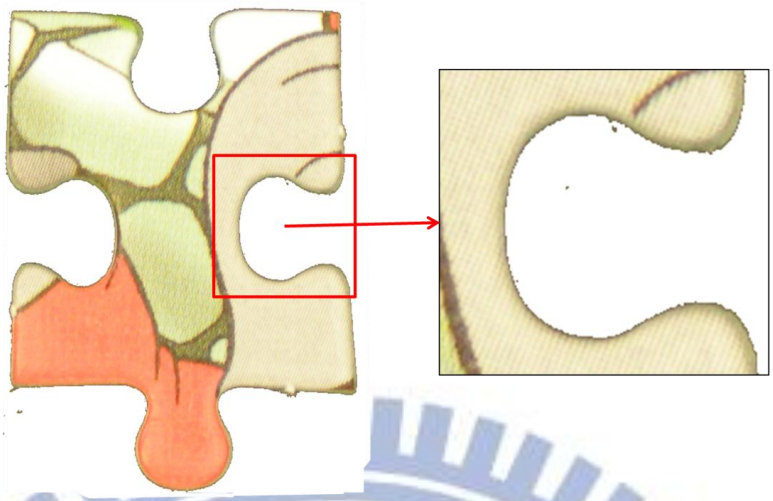


Fig. 7. An example to show that colors on boundary of a puzzle piece are darker due to shadows on the boundary.

For a sample point X, two sample points Y and Z are taken on its both sides within the same distance to get a line segment $\overline{YZ}$. Locate the point P inside the piece, and the line segment $\overline{XP}$ is perpendicular to $\overline{YZ}$ with a distance D. Then, the average hue, saturation, and intensity values of pixels within the 3×3 window centered at P are computed. These values are considered as the color features of X. Finally, color features of all sample points are grouped to form a color feature string. Fig.8 illustrates the color feature extraction.

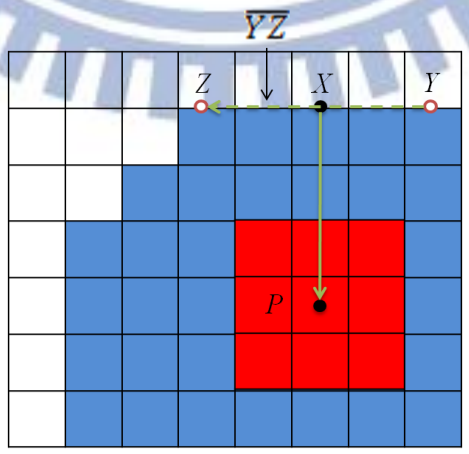


Fig. 8. An Example of color extraction with $D=3.5$.

2.3 Common Boundary Finding

To reconstruct the puzzle image, the first thing we have to do is to find a possible common boundary between two puzzle pieces. Note that this boundary will correspond to a pair of similar segments in these two pieces' RLCSs. In this section, we will describe how to find the most similar segment pair of two puzzle pieces. In some situations, it is difficult to find the correct pair of similar segments in two adjacent puzzle pieces by only using shape information, since there might be more than one pair of similar segments in two puzzle pieces (see Fig. 9). To treat the above problem, a common boundary finding technique is provided based on both shape and color features.

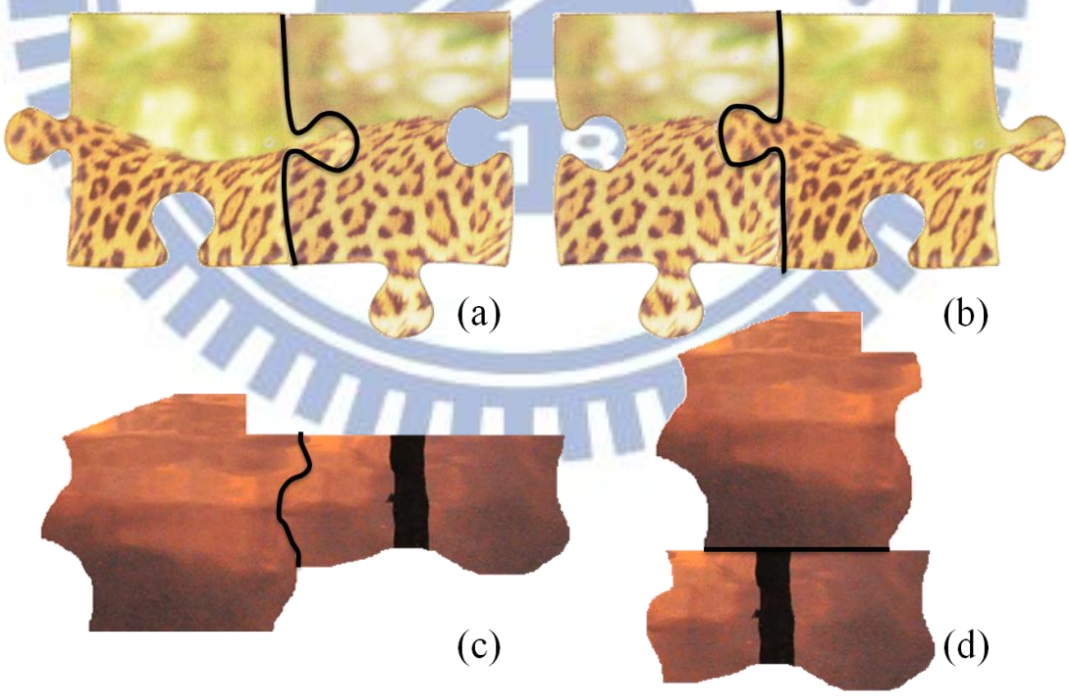


Fig. 9. Examples of multiple similar segments with only shape information used. (a) and (c) Results of correct pairs of similar segments. (b) and (d) Results of incorrect pairs of similar segments.

In the proposed technique, first, for two puzzle pieces, several possible common boundaries, which are apt to merge these two puzzle pieces, are found based on their RLCSs. Second, for each possible common boundary, the matching points of this boundary, which can align two corresponding similar segments representing the possible boundary, are located based on the shape and color feature strings. Finally, the best possible common boundary will be extracted. Following is the details of our common boundary finding method.

First, the proposed method attempts to find several possible common boundaries between two puzzle pieces based on their RLCSs. This method is an adaption of Smith-Waterman algorithm (SWA) [21], which is designed for determining similar regions between two nucleotide or protein sequences. It performs local sequence alignment, that is, SWA can find a pair of similar matching segments between two strings. Note that each element in a string is a single symbol. However, our purpose is to find several pairs of similar matching segments based on RLCSs, in which each element contains two values: direction code and its run length. Thus, SWA is modified to meet this situation. The details are described as follows.

Let $S_1 = (R_{11},...,R_{1m})$ and $S_2 = (R_{21},...,R_{2n})$ be the RLCSs of two puzzle pieces, where m is the number of runs in S_1, and n is the number of runs in S_2. Assume that $m \le n$, $S_1' = (R_{11},R_{12},\ldots,R_{1m},R_{11},R_{12},\ldots,R_{1m})$ and $S_2' = (R_{21},R_{22},\ldots,R_{2n},R_{21},R_{22},\ldots,R_{2m})$. S_1' and S_2' are created to be circular strings, they are used to deal with the problem that for a similar segment

pair, each segment may consist of two separated substrings, one in the front of a RLC string, the other in the rear of this string. Fig. 10 shows an example to illustrate this problem.

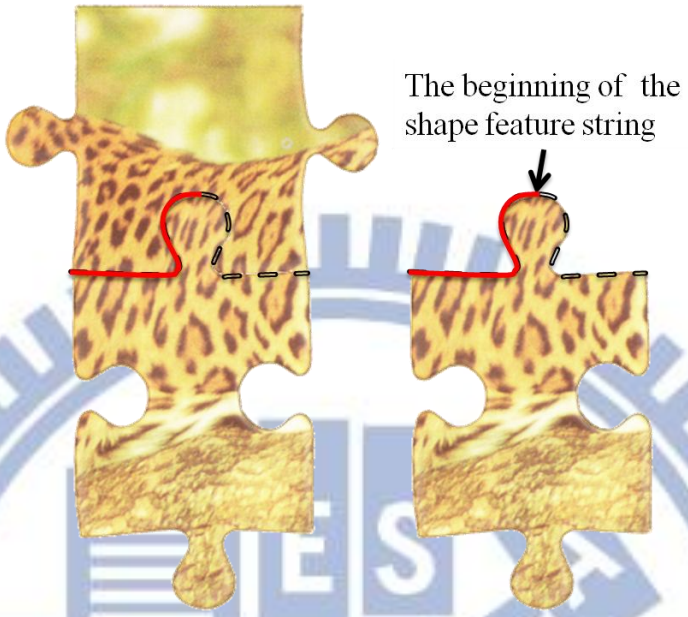


Fig. 10. An example of a real similar segment pair in which each segment consists of two separated substrings, one in the front of a RLCS, and the other in the rear of this string. Solid line represents the front substring, and dotted line represents the rear substring.

Then, a $(2m+1)\times(m+n+1)$ matrix H is built as follows:

$$H(i,0)=0, \quad \text{for } 0\le i\le 2m,$$
$$H(0,j)=0, \quad \text{for } 0\le j\le (m+n),$$

$$H(i,j)=\max\begin{Bmatrix} 0 & \\ H(i-1,j-1)+w(R_{1i},R_{2j}) & \textit{Match/Mismatch} \\ H(i-1,j)+w(R_{1i},-) & \textit{Deletion} \\ H(i,j-1)+w(-,R_{2j}) & \textit{Insertion} \end{Bmatrix}, \text{ for } 1\le i\le 2m, 1\le j\le (m+n), \tag{2}$$

$$w(R_{1i},R_{2j})=\begin{cases} \textit{SimiMat}[C_{1i}][C_{2j}]\times L_{1i}, & \text{if } L_{1i}<L_{2j} \\ \textit{SimiMat}[C_{1i}][C_{2j}]\times L_{2j}, & \text{if } L_{2j}<L_{1i} \end{cases}, \tag{3}$$

$$w(R_{1i},-)=(-10)\times L_{1i}, \tag{4}$$

$$w(-,R_{2j})=(-10)\times L_{2j}. \tag{5}$$

Where $w(a,b)$ is a gap-scoring scheme, $a,b \in S_1, S_2$, or $\{'-'\}$, '–' stands for missing element. R_{1i} (R_{2j}) is the ith (jth) run in S_1 (S_2). *SimiMat* is a similarity measure matrix to represent the similarity of two direction codes (see Table 2).

Table 2. The *SimiMat* matrix values with codes C_1 and C_2.

$\lvert C_1 - C_2 \rvert$	0	1	2	3	4	5	6	7
SimiMat[C_1][C_2]	10	5	-1	-4	-8	-10	-10	-10
$\lvert C_1 - C_2 \rvert$	8	9	10	11	12	13	14	15
SimiMat[C_1][C_2]	-10	-10	-10	-10	-8	-4	-1	5

Note that in SWA, two similar strings may have some insertions and deletions. But in puzzle solving, two similar strings should represent the common boundary of two adjacent puzzle pieces in the original puzzle image. Thus, only few insertions and deletions are allowed. This means that for a pair of similar segments, the number of consecutive unmatched elements should be less than a threshold T_2, which depends on the size of puzzle pieces. Therefore, on the process of building matrix H, a counter is used to record the number of consecutive unmatched elements. If the counter is larger than T_2 in calculating the value of $H(i,j)$, the value of $H(i,j)$ will be reset to 0.

Note that $H(i,j)$ stands for the maximum similarity score between segment MS_1 $(R_{1i'},...,R_{1i})$ and segment MS_2 $(R_{2j'},...,R_{2j})$, where $R_{1i'}$ ($R_{2j'}$) is the beginning run of MS_1

(MS_2). The bigger the value of $H(i, j)$ is, the more similar the pair of segments is.

To locate $R_{1i'}$ and $R_{2j'}$ for each similar segment pairs, we start from position (i, j), then, we go backwards to one of positions $(i-1, j)$, $(i, j-1)$, and $(i-1, j-1)$ depending on the direction of movement used to build $H(i, j)$. Keep the process until it reaches a position (i', j') with $H(i', j') = 0$ or position $(0,0)$.

Based on matrix H we can extract several similar segments. The top 10 largest values in H are extracted, each of which corresponds to a pair of similar segments. However, there may be some duplicates in these ten pairs since strings S_1' (S_2') is a concatenation of S_1 (S_2). Therefore, all pairs of similar segments are examined and duplications are discarded.

After several pairs of similar segments between two puzzle pieces are found, our method attempts to find a pair of matching points in RLCSs for each pair of similar segments, so that we can merge these two puzzle pieces based on the matching points.

Note that a pair of similar segments found might consist of matching runs and several insertions or deletions (see Fig. 11). That is each segment can be split into several sub-segments by these insertions or deletions. Based on this phenomenon, we take the longest sub-segment in each segment; it has the maximum similarity. Assume that the locations of matching points will be situated at the longest sub-segments. Then, the shape and color features on these two sub-segments are used to find the best pair of matching points on these two segments. More details are described as follows.

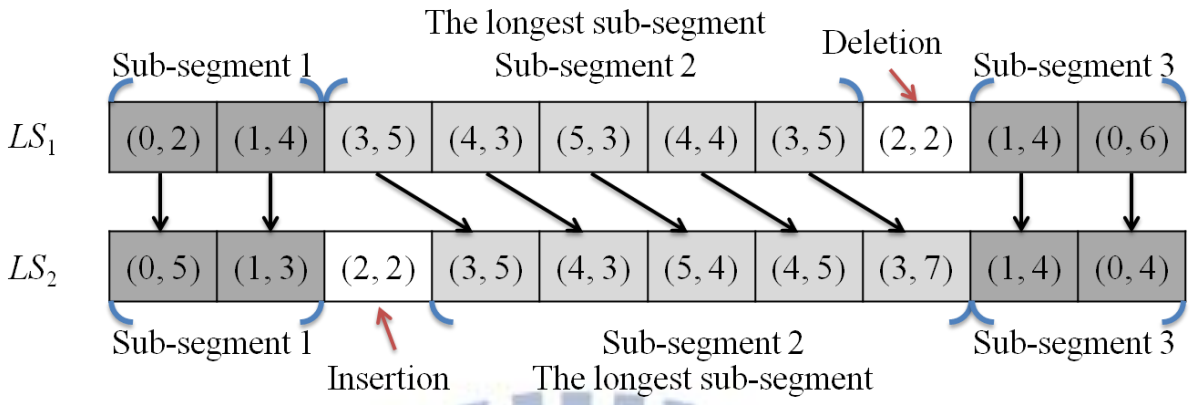


Fig. 11. A pair of similar segments with arrows standing for matching runs.

Let LS_1 (LS_2) be the longest sub-segment of MS_1 (MS_2), LsS_1 (LsS_2) be the result string of run-length decoding (RLD) for LS_1 (LS_2) with length l (l'), $LsS_1 = (C'_{11}, C'_{12}, ..., C'_{1l})$ ($LsS_2 = (C'_{21}, C'_{22}, ..., C'_{2l'})$), and LcS_1 (LcS_2) be the sub-string of color features of pixels on LS_1 (LS_2) with length l (l'), $LcS_1 = (\mathbf{CS}'_{11}, \mathbf{CS}'_{12}, ..., \mathbf{CS}'_{1l})$ ($LcS_2 = (\mathbf{CS}'_{21}, \mathbf{CS}'_{22}, ..., \mathbf{CS}'_{2l'})$). LsS_1 and LsS_2 are used to estimate a *Local Shape Similarity* (*LSS*), and LcS_1 and LcS_2 are used to estimate a *Local Color Difference* (*LCD*). In case these two sub-segments LsS_1 and LsS_2 are not perfectly aligned, that is $l \neq l'$, the shorter sub-segment may be shifted relative to the longer one to find the maximum similarity measure (see Fig. 12).

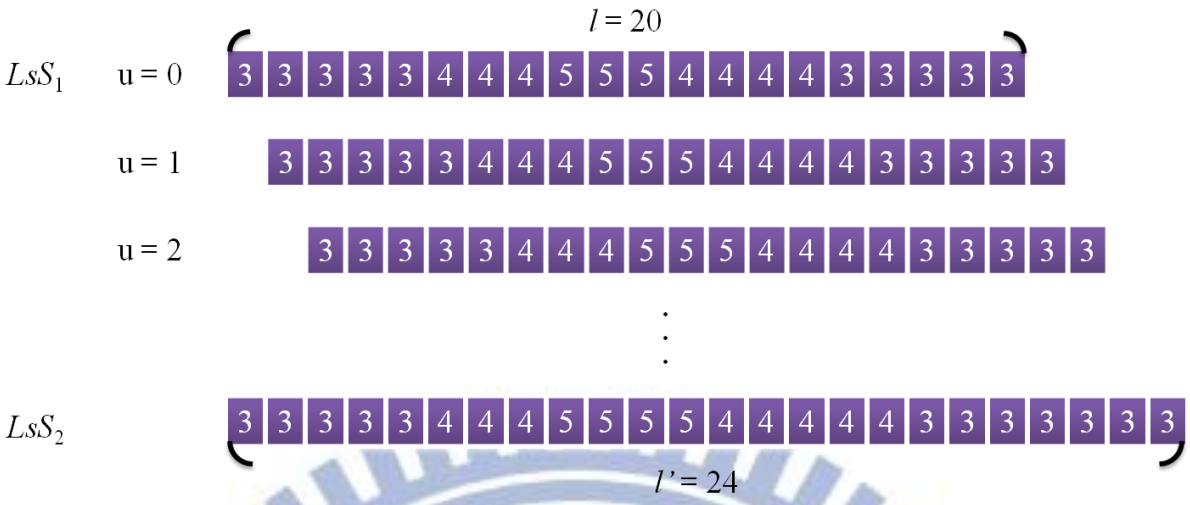


Fig. 12. An example of finding matching points with $l \neq l'$.

Let $l \leq l'$, *LSS* and *LCD* are computed as follows:

$$LSS_u = \sum_{t=0}^{l-1} SimiMat[C'_{1(1+t)}][C'_{2(1+t+u)}], \tag{6}$$

$$LCD_u = \frac{\sum_{t=0}^{l-1} \left(\left| \mathbf{CS}'_{1(1+t)} - \mathbf{CS}'_{2(1+t+u)} \right| \right)}{l}. \tag{7}$$

Where index *u* denotes the number of times the shorter sub-segment shifts. After obtaining all LSS_u and LCD_u, the proposed method finds the maxima LSS_{max} and $LCD_{max,}$ and the minima LSS_{min} and LCD_{min}, then uses these values to estimate the Local Similarity Measure (*LSM*) by Equation (8) for each possible pair of matching points.

$$LSM_u = \alpha \times \left(\frac{LSS_u - LSS_{min}}{LSS_{max}} \right) + (1-\alpha) \times \left(1 - \frac{LCD_u - LCD_{min}}{LCD_{max}} \right). \tag{8}$$

Where α is set to determine the contribution of shape and color features for *LSM*. The pair

of points with maximum *LSM* will be considered as the pair of matching points for segments MS_1 and MS_2.

According to the above method, a pair of matching points is found for each similar segment pair. After that, *Shape Similarity* (*SS*) and *Color Difference* (*CD*) are estimated for each pair of similar segments. Let (*x*, *y*) be the pair of matching points for a pair of similar segments MS_1 and MS_2, MsS_1 and MsS_2 be the result sub-string of RLD for MS_1 and MS_2 with length *k* and *k'*, and McS_1 and McS_2 be the corresponding color feature substrings of MS_1 and MS_2 with length *k* and *k'*, $MsS_1 = (C_{11},C_{12},\ldots,C_{1k})$, $MsS_2 = (C_{21},C_{22},\ldots,C_{2k'})$, $McS_1 = (\mathbf{CS}_{11},\mathbf{CS}_{12},\ldots,\mathbf{CS}_{1k})$, and $McS_2 = (\mathbf{CS}_{21},\mathbf{CS}_{22},\ldots,\mathbf{CS}_{2k'})$. Assume that $x < y$ and $k < k'$, the *SS* and *CD* will be calculated as follows:

$$SS = \sum_{t=0}^{x-1} SimiMat[C_{1(x-t)}][C_{2(y-t)}] + \sum_{t=1}^{k-x} SimiMat[C_{1(x+t)}][C_{2(y+t)}], \tag{9}$$

$$CD = \frac{\sum_{t=0}^{x-1}\left(\left|\mathbf{CS}_{1(x-t)} - \mathbf{CS}_{2(y-t)}\right|\right) + \sum_{t=1}^{k-x}\left(\left|\mathbf{CS}_{1(x+t)} - \mathbf{CS}_{2(y+t)}\right|\right)}{k}. \tag{10}$$

Then, the similarity measure (*SM*) of each pair of matching segments can be estimated by following Equation:

$$SM = \alpha \times \left(\frac{SS - SS_{min}}{SS_{max}}\right) + (1-\alpha) \times \left(1 - \frac{CD - CD_{min}}{CD_{max}}\right). \tag{11}$$

Where SS_{max}, CD_{max} are the maxima of all *SS*s and *CD*s, and SS_{min}, CD_{min} are the minima of all *SS*s and *CD*s. Finally, the maximum *SM* is considered as the similarity measure of the pair

of two pieces, and the pair of similar segments with maximum *SM* will be considered as the possible common boundary between these two pieces. If *SM* is close to 1 for a possible common boundary, the corresponding two puzzle pieces are more suitable to be merged.

2.4 Adjacent Pieces Merging

The final destination is to correctly restore the puzzle pieces; the aforementioned *SM* is used to reconstruct the puzzle piece by piece. First of all, the similarity measures of all pairs of puzzle pieces are ranked from maximum to minimum. Secondly, according to the order, our merging procedure executes the following steps:

(1) Take the pair of two puzzle pieces with maximum *SM* and merge them based on their matching points.

(2) Based on the merged piece, take the next pair of puzzle pieces, where one of them is a merged piece. If the overlap of the unmerged piece with the merged piece occupies 20% of the unmerged piece, it would not be merged. Otherwise, this piece will be merged with the merged piece based on the pair of matching points.

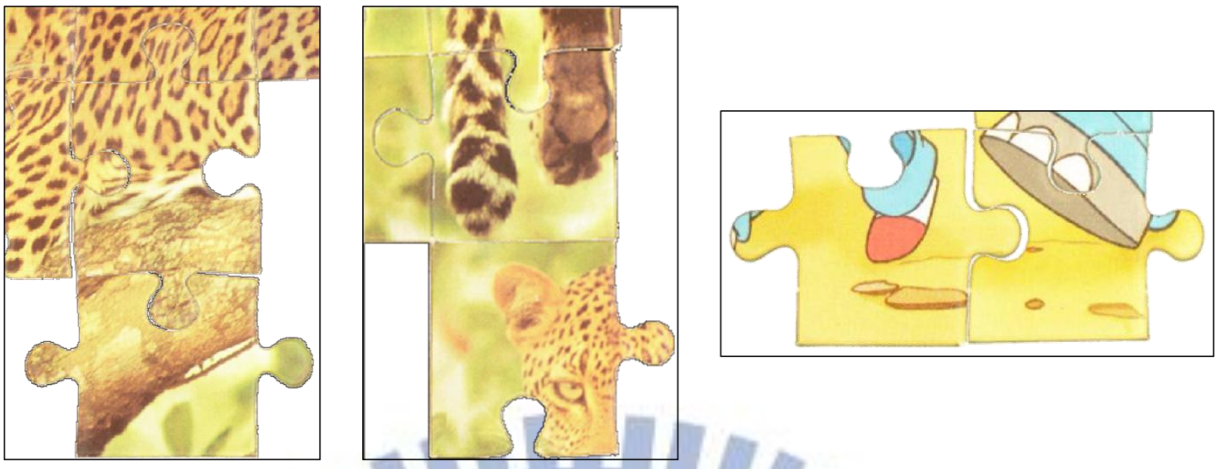
(3) Repeat Step (2) until all pieces have been merged.

The result will be considered as the original puzzle image.

CHAPTER 3

SEMI-AUTOMATIC PUZZLE SOLVER

The proposed method attempts to reconstruct all kinds of puzzle pattern without any common puzzle conditions. According to the adjacent pieces merging step described in Chapter 2.4, our method will automatically select a pair of puzzle pieces in each step, and merges them by their matching points, until all the pieces have been merged. However, our method might select few incorrect pairs of matching pieces. These errors occur when nonadjacent puzzle pieces have texture or similar colors and shapes near boundary. Fig. 13 demonstrates these situations. In Fig. 13(a), two nonadjacent puzzle pieces are merged due to that they have texture boundaries. In Figs. 13(b) and (c), two nonadjacent puzzle pieces are merged due to that their boundaries have similar shapes and colors. These errors are hard to be solved automatically, and this will result in that our solver cannot restore the puzzle pieces to the original image completely.



(a) (b) (c)

Fig. 13. Incorrect matching pairs of pieces selected by our method.

In order to overcome this problem, we propose a semi-automatic solver to successfully restore puzzle pieces because of these errors may be correctly recognized by a person. This solver uses the proposed method described in Chapter 2, but it allows a user deciding if the current merging step is correct or not. If a pair of matching pieces is recognized as an incorrect merging, this pair of puzzle pieces will not be merged. Fig. 14 gives an example. Nevertheless, there are still incorrect matching pieces that cannot be recognized by the user, so our solver implements a backtracking method that can remove some error merging. Finally, we can get the correct result of reconstructing puzzle pieces. Fig. 15 shows an example of the backtracking method. In Fig. 15(a), piece R2 is selected and the user determines that the merging of R1 and R2 is correct. After several mergings, R2, R3, and R4 are merged. Then, in Fig. 15(b), piece R5 is selected, and the user finds that the merging of R1 and R2 is incorrect; the previous mergings including R2 are removed. The result is shown in Fig 15(c).

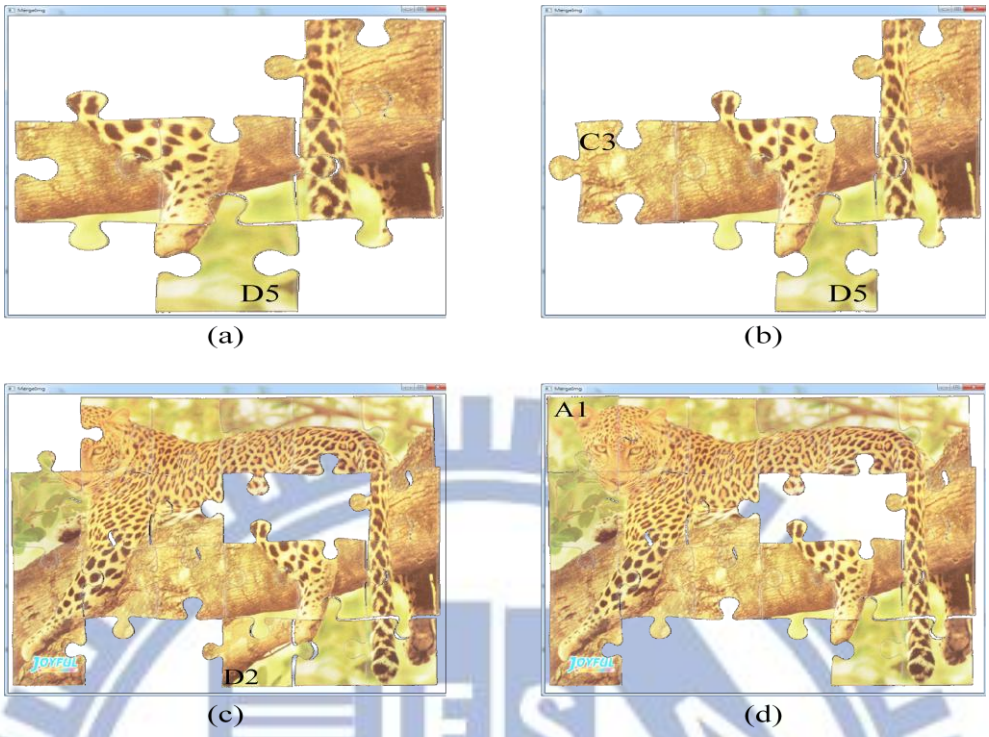


Fig. 14. User decision on the pieces merging step of puzzle solver. (a) D5 is selected in the merging step, and the user determines that this step is correct. (b) C3 is selected in the next step. (c) D2 is selected in the merging step, and the user determines that this step is incorrect. (d) D2 is discarded by the solver, and A1 is selected in the next step.

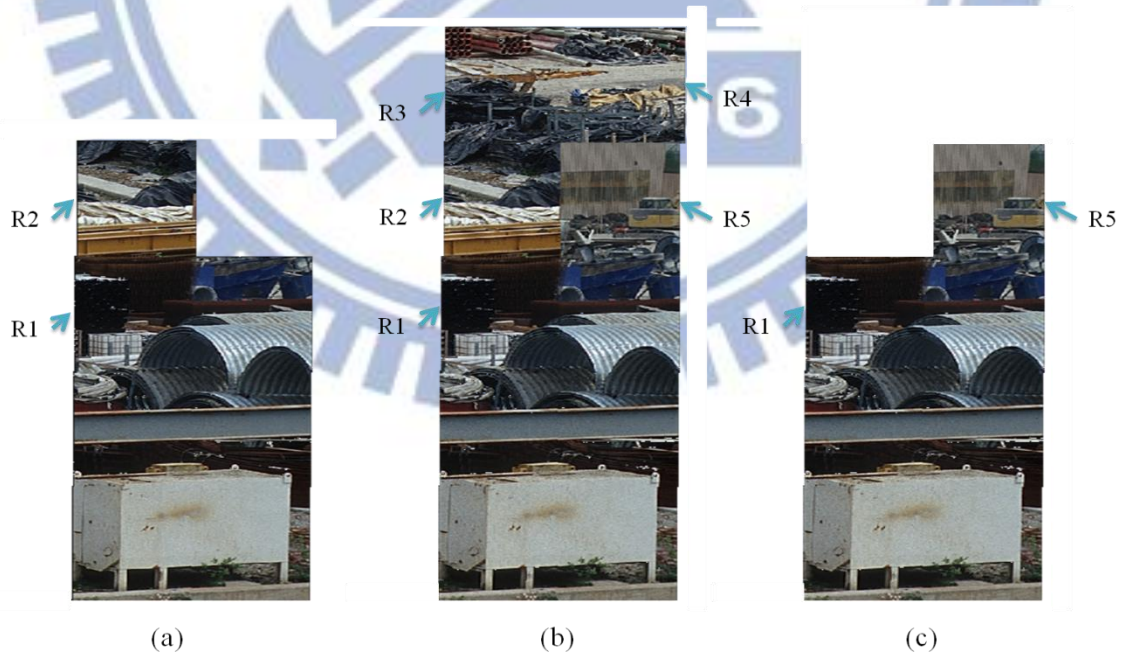


Fig. 15. An example of backtracking method. (a) Piece R2 is selected and merged with R1. (b) When R5 is selected, the merging of R2 and R1 is found incorrect. (c) The result after backtracking with R2, R3, and R4 removing.

To sum up, the block diagram of the semi-automatic puzzle solver is shown in Fig. 16 and it is depicted as follows:

(1) Our solver automatically extracts shape and color features for all puzzle pieces.

(2) It calculates a similarity measure for each pair of pieces.

(3) It ranks these similarity measures from maximum to minimum and restores the puzzle image based on this order.

(4) This solver implements the adjacent pieces merging step described in Chapter 2.4, but it requests the user to determine if two selected puzzle pieces should be merged in each merging.

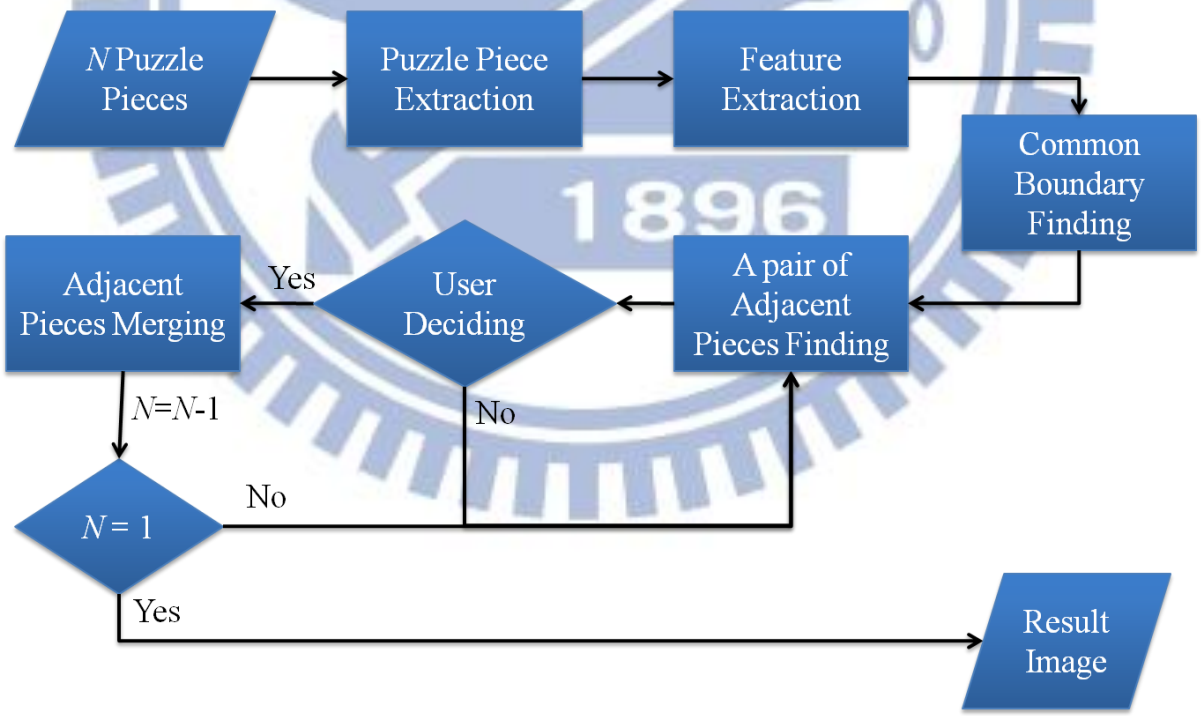


Fig. 16. The block diagram of the semi-automatic puzzle solver.

CHAPTER 4

EXPERIMENTAL RESULTS

The final goal of the semi-automatic puzzle solver is to restore all kinds of puzzles into their original images. Here, we will calculate the number of times for incorrect merging. Note that the solver will keep solving until all puzzle pieces are placed in their correct locations.

In this thesis, we present six puzzle images. Four of them are produced by factitiously splitting original puzzle images into various shapes of pieces. The other two are real-world puzzle pieces derived from the database of Nielsen et al. [16] which scan real-world puzzle pieces by a scanner. These puzzles have been chosen to show that this solver can cope with any kinds of puzzle pieces. The parameter α is set to 0.5. And if the puzzle images are created by a person, D is set to 0. Otherwise, if they are created by a scanner D is set to 2.

Example 1 (Blockhouse): In this example, the original puzzle image has been artificially divided into 13 pieces with different shapes (see Fig. 17). The proposed solver restores this puzzle without any incorrect merging. The result is shown in Fig. 18.

Example 2 (Lake): In this example, the original puzzle image has been artificially divided into 31 varied pieces (see Fig. 19). This puzzle is restored with 6 incorrect mergings. The result is shown in Fig. 20.

Example 3 (Leopard): This puzzle is a real-world puzzle of 24 pieces. The proposed solver restores this puzzle with 16 incorrect mergings. The result is shown in Fig. 21.

Example 4 (Benjamin): This puzzle is a real-world puzzle of 54 pieces. This puzzle is restored with 22 incorrect mergings. The result is shown in Fig. 22.

Example 5 (Construction50, Construction160): This puzzle image depicts a construction with varies colors and texture (see Figs. 23 and 24); it is derived from Nielsen et al. [16]. In this example, we divide this image into 50 and 160 rectangular pieces. The piece size of the first puzzle image is 154×198 pixels, and the other is 100×104 pixels. The First puzzle is restored without any incorrect merging, and the result is shown in Fig. 25. The second puzzle of 160 pieces is also successfully restored with 59 incorrect mergings.

In our proposed solver, α value is the key parameter to affect the number of incorrect mergings. This value is set to determine the contribution of shape and color information, and it is normalized to [0, 1]. If α is close to 1, the proposed solver coping with the puzzle mainly depends on shape information. On the contrary, if α is close to 0, our solver coping with the puzzle mainly depends on color information. In Table 3, five puzzles (*Benjamin*, *Blockhouse*, *Construction50*, *Lake*, *Leopard*) are used to analyze the effect of α by computing the number of incorrect mergings occurred. From this table, we can see that each puzzle can be restored with few incorrect mergings by using a specific α value. The selection of α will be based on the characteristics of each puzzle, if the colors and textures

in a puzzle are similar, then this puzzle should be restored using the shape information, that is α is set to close 1, like *Leopard*. On the other hand, if the shapes of pieces are similar, then this puzzle should be restored using the color information, that is α is set to close 0, like *Construction50*. We can also see that when $\alpha = 0.5$ our puzzle solver can restore all puzzles effectively. Therefore α is suggested to set to 0.5.

Table 3. Analysis results on different α. Each value in this table is the number of incorrect merging to restore the designated puzzle. '-' means that this puzzle cannot be restored.

α \ Puzzle	*Benjamin*	*Blockhouse*	*Construction50*	*Lake*	*Leopard*
0.1	45	0	0	-	98
0.2	29	0	0	11	86
0.3	19	0	0	6	56
0.4	20	0	0	2	33
0.5	22	0	0	6	16
0.6	30	0	8	10	16
0.7	45	2	55	16	16
0.8	49	2	-	28	2
0.9	63	2	-	49	4

Fig. 17. Puzzle pieces of puzzle *Blockhouse*.

Fig. 18. The restored image of puzzle *Blockhouse*.

Fig. 19. Puzzle pieces of puzzle *Lake*.

Fig. 20. The restored image of puzzle *Lake*.

Fig. 21. The restored image of puzzle *Leopard*.

Fig. 22. The restored image of puzzle *Benjamin*.

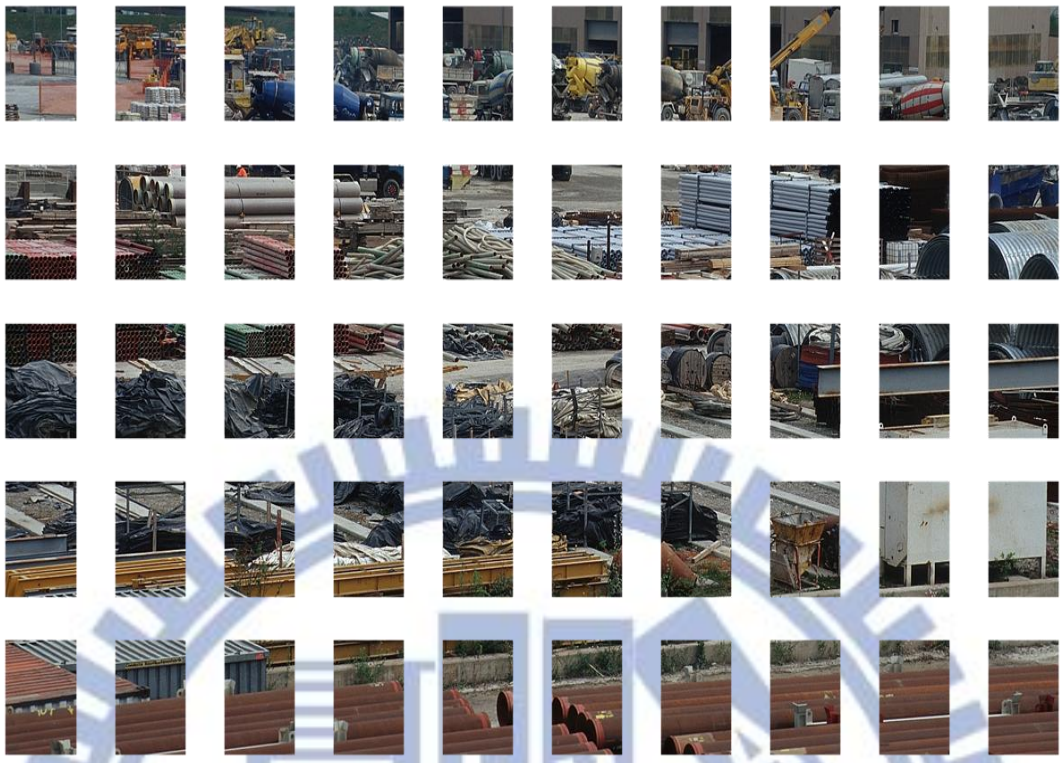


Fig. 23. Puzzle pieces of puzzle *Construction50*.

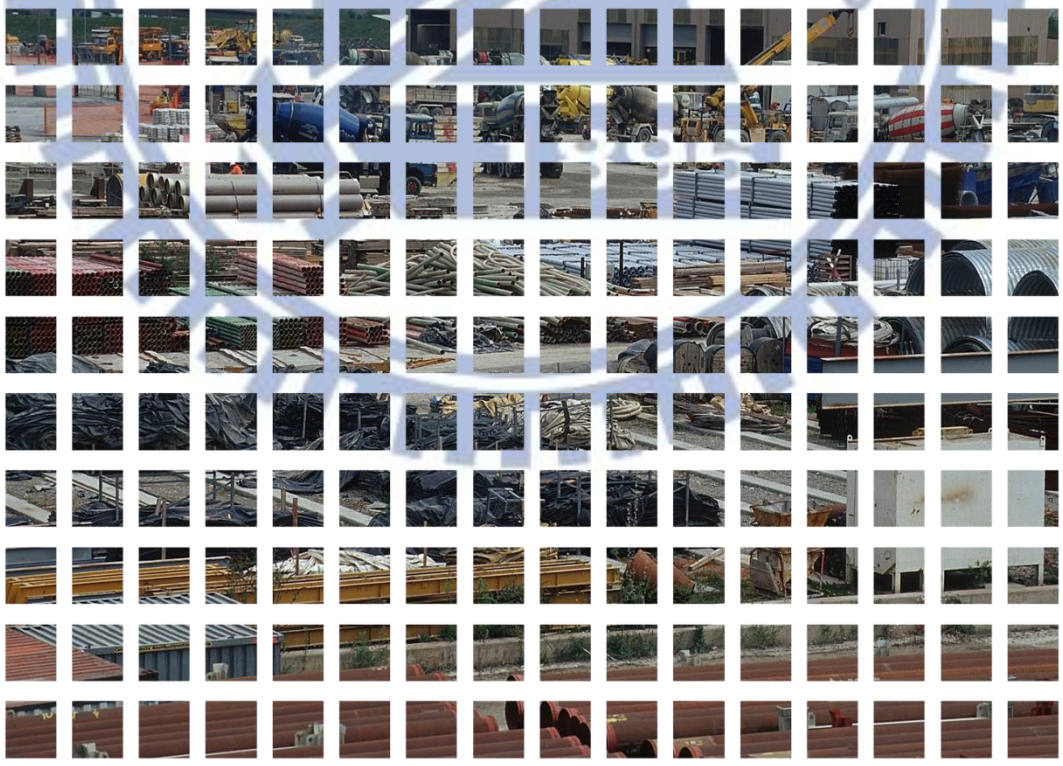


Fig. 24. Puzzle pieces of puzzle *Construction160*.

Fig. 25. The restored image of puzzle *Construction*.

CHAPTER 5
CONCLUSIONS

This thesis provides a semi-automatic puzzle solver, which is based on shape and color information. The target puzzles are not restricted by some specific patterns. And the solver does not use any common puzzle limitations. The experiments were performed with real-world and artificial puzzle images. The experimental results show that our method is successful for restoring all kinds of puzzle pieces.

In this method, first, the shape and color features are extracted; they are next used to estimate the similarity measure of a possible common boundary for each pair of puzzle pieces. Then, a simple user interactive method is provided to allow a user deciding if two selected pieces are adjacent or not. Our solver will select another pair of puzzle pieces which may be adjacent as soon as the user decides previous selected pieces are not adjacent. This method makes any kinds of puzzle pieces be restored completely. And a user can easily restore a puzzle by our puzzle solver.

REFERENCES

[1] H. C. da Gama Leitao and J. Stolfi, "*Automatic reassembly of irregular fragments*," Univ. Campinas, Campinas, Brazil, Tech. Rep. IC-98-06, 1998.

[2] H. C. da Gama Leitao and J. Stolfi, "*Information contents of fracture lines*," Univ. Campinas, Campinas, Brazil, Tech. Rep. IC-99-24, 1999.

[3] A. R. Willis and D. B. Cooper, "*Computational reconstruction of ancient artifacts*," *IEEE Signal Process. Mag.*, Vol. 25, No. 4, pp. 65–83, Jul. 2008.

[4] F. S. Kuhl, G. M. Crippen, and D. K. Friesen. "*A combinatorial algorithm for calculating ligand binding*," *Journal of Computational Chemistry*, Vol. 5, No. 1, pp. 24-34, 1984.

[5] Elaine C. Meng and Irwin D. Kuntz, "*Molecular docking: a tool for ligand discovery and design*," *Drug Information Journal*, Vol. 28, pp. 735-749, 1994.

[6] Y.-X. Zhao, M.-C. Su, Z.-L. Chou, and J. Lee, "*A puzzle solver and its application in speech descrambling*," *In ICCEA*, 2007.

[7] L. Zhu, Z. Zhou, and D. Hu, "*Globally consistent reconstruction of ripped-up documents*," *IEEE TPAMI*, 2008.

[8] H. Freeman and L. Gardner, "*A pictorial jigsaw puzzles: The computer solution of a problem in pattern recognition*," *IEEE Trans. Electron. Comput.*, Vol. EC-13, No. 2, pp. 118–127, Apr. 1964.

[9] G. Radack and N. Badler, "*Jigsaw puzzle matching using a boundary-centered polar encoding*," *Comput. Graphics Image Process. 19*, pp. 1–17, 1982.

[10] H. Wolfson, E. Schonberg, A. Kalvin, and Y. Lamdan, "*Solving jigsaw puzzles by computer*," *Ann. Oper. Res.*, Vol. 12, No. 1, pp. 51–64, Dec. 1988.

[11] D. Goldberg, C. Mallon, and M. Bern, "*A global approach to automatic solution of jigsaw puzzles*," in Proc. 18th Annu. Symp. Comput. Geometry, Barcelona, Spain, pp. 82–87, 2002.

[12] D. Kosiba, P. Devaux, S. Balasubramanian, T. Gandhi, and R. Kasturi, "*An automatic jigsaw puzzle solver*," In: Proceedings 12th IAPR International Conference on Computer Vision and Image Processing, Jerusalem, Vol. 1, pp. 616–618, October 1994.

[13] M. G. Chung, M. M. Fleck, and D. A. Forsyth, "*Jigsaw puzzle solver using shape and color*," in Proc. 4th Int. Conf. Signal Process., pp. 877–880, 1998.

[14] F. H. Yao and G. F. Shao, "*A shape and image merging technique to solve jigsaw puzzles*," *Pattern Recognit. Lett.*, Vol. 24, No. 12, pp. 1819–1835, Aug. 2003.

[15] T. R. Nielsen, P. Drewsen, and K. Hansen, "*Solving jigsaw puzzles using image features*," *Pattern Recognit. Lett.*, Vol. 29, No. 14, pp. 1924–1933, Oct. 2008.

[16] M. Makridis and N. Papamarkos, "*A new technique for solving puzzles*," *IEEE Trans. On systems, man, and cybernetics-part B: cybernetics*, Vol. 40, No. 3, June 2010.

[17] D. Chetverikov and Z. Szabo, "*A simple and efficient algorithm for detection of high curvature points in planar curves*," *Robust Vis. Ind. Appl.*, Vol. 128, pp. 175–184, 1999.

[18] T. Kohonen, "*Self-Organizing Maps*", 2nd ed. Berlin, Germany: Springer-Verlag, 1997.

[19] H. Freeman, "*On the encoding of arbitrary geometric configurations*," *IRE Trans. Electron. Comput.*, Vol. 10, No. 2, pp. 260–268, Jun. 1961.

[20] T. F. Smith and M. S. Waterman, "*Identification of Common Molecular Subsequences*," *Journal of Molecular Biology 147*, pp. 195–197, 1981.