Form Segmentation and Component Classification for Clinic Document Image Analysis[1]

Yung-Chao Tseng (曾詠超) Wen-Hsiang Tsai[2](蔡文祥)
Institute of Computer and Information Science
National Chiao Tung University
Hsinchu , Taiwan, Republic of China

Abstract

An integrated approach to form segmentation and component classifica-tion for clinic document image analysis is proposed. First, a modified Hough algorithm is proposed, which can be used to detect consecutive line segments from noisy form images. Based on this algorithm, a top-down recursive form frame extraction algorithm is proposed, by which form frames in a clinic data image can be extracted structurally. A masking method is then proposed to restore broken character strokes resulting from frame extraction, and a component extraction and merge method is proposed for form content extraction. Finally, several features are proposed and a back propagation neural network are employed, for classification of preprinted components and filled-in handwritten components in form fields. Experimental results are shown to prove the feasibility of the proposed approach.

Keyword: image segmentation、clinic document、line extraction、Hough algorithm、component classification 、back propagation neural network.

1. Introduction

Most clinic data are filled forms. A clinic data form is usually composed of frames and content components. The former ones consist of line segments, and the latter ones include preprinted and filled-in components, most of characters. Form segmentation for clinic data images mainly includes two parts: frame extraction and component extraction. In clinic data forms, filled-in data often touch frames. This causes the problem of yielding broken strokes after frame extraction, which need be solved. We also need to discriminate filled-in data from preprinted data for further analysis of clinic data images, so the classification of preprinted and filled-in components is also necessary. This study emphasizes the designs of both effective form segmentation and component classification algorithms for clinic document analysis.

The first step in form segmentation is frame extraction, whose objective is to separate form frames out of a clinic data image. Lu and Fan [1] proposed two methods to extract form frames. The first is to thin an input image, followed by the work of extracting the frames by feature point clustering. The second method is to trace the contours of the image components and extract the form frames by finding frame-like connected components. Li, et al. [2] proposed a frame line detection algorithm by mathematical morphology. Lam, et al. [3] proposed a form frame extraction method which gets lines from contours by chain code

[1] This work was supported by National Science Council under Grant NSC86-2213-E009-113.
[2] To whom all correspondence should be addressed.

tracing.

After frame extraction, the broken-stroke restoration problem arises, which is discussed in several studies. Yu and Jain [4] proposed a restoration method based on block adjacency graphs and restored broken strokes by interpolation. Maderlechner [5] introduced a method for restoring broken strokes based on the use of the stroke skeleton. Casey et al. [6] proposed a method which restores broken strokes by using a line tracking technique. Doermann and Rosenfeld [7] used cross section computation to solve the broken-stroke restoration problem.

About field content extraction which is the next step in form segmentation, there are several commonly used approaches, such as the run length smoothing algorithm (Wahl et al. [8]). The projection cut algorithm (Wang and Srihari {9}), and the connected component analysis algorithm (Fletcher and Kasturi [10]).

As to content component classification, Srihari et al. [11] proposed six types of symbolic features which can be extracted from connected components in form contents: the standard deviation of connected component widths, the standard deviation of connected component heights, the average component density, the aspect ratio, the distinct different heights, and the distinct different widths. The classifier employed is a Fisher's linear discriminant function.

In this paper, an integrated approach to form segmentation and component classification for clinic document image analysis is proposed. The major contribution is that more effective algorithms are designed for yielding less erroneous results. First, a modified Hough algorithm is proposed to detect line segments from noisy form images. A top-down recursive form frame extraction algorithm is proposed accordingly, by which form frames in a clinic data image can be extracted structurally. A method using a set of masks is then proposed to restore broken character strokes resulting from frame extraction, followed by the use of a component extraction and merge method for form content extraction. Finally, several features are proposed and a back propagation neural network are employed, for classification of preprinted and filled-in handwritten components in form fields. Experimental results shown to prove the feasibility of the proposed approach are also included.

In real cases, many clinic data images are quite complicated. In order to reduce the complexity of the processing work, some assumptions are made in this study, as described in the following.

1. Only gray-valued clinic data images are processed.
2. There is only one outer frame in each form.
3. All frames are composed of solid vertical lines and horizontal lines.
4. All printed Chinese characters are isolated.

A flowchart of the proposed clinic data image analysis system is shown in Fig. 1. After an input data form is scanned and binarized into a black and white image, form frame extraction is performed, based on a modified Hough algorithm.. This may result in broken strokes. So a restoration step is applied, using the proposed masking method. Form frames are then removed, leaving form field contents for further processing. The form may be skewed during scanning, so a skew correction is performed. Form component extraction is then proceeded, followed by classification of the extracted components, if necessary. Finally, form reconstruction is performed to produce a noise-free form sheet identical to the scanned image. The main components of the system and the employed image processing techniques will be described in detail in the following sections.

More specifically, in Section 2, the proposed form frame extraction method is described, which includes several proposed algorithms. In Section 3, the proposed form content extraction method is presented in detail. In Section 4, the proposed method for form content component classification method using the back propagation neural network is described. Section 5 includes some experimental results, followed by some conclusions in Section 6.

2.Form Frame Extraction

In this section, we describe the detailed steps of the proposed form frame extraction method, including the proposed modified Hough algorithm, a line detection procedure, a recursive form frame detection algorithm, the proposed broken-stroke restoration method, and a process to remove the detected form frame for further processing.

A. Modified Hough Algorithm

A good way to extract line segments in images is to use the Hough transform. However, in

the image of a filled clinic data image, noise coming from the filled-in handwritten data often produces false lines. By using the conventional Hough algorithm, such false lines will be detected and cannot be distinguished from true lines. In this study we proposed a modified Hough algorithm to ignore noise coming from filled-in data.

In the Hough algorithm, the major data structure is Accumulator[][] which is used to keep the accumulation values of the counting space of (ρ, θ) calculated by the formula:

$$\rho = x\cos\theta + y\sin\theta .$$

Besides the counting space, two additional data structures are needed in the proposed modified Hough algorithm. The array Buffer[][] is used to buffer the accumulation value yielded by applying the above formula. Another data structure LastPixel[][] is used to record the last pixel which is just calculated. Two pixel selection rules are used in the algorithm, as described in the following:

Rule 1: If the distance from the pixel being processed to the last pixel is smaller than a value *Dist*, then increase the value of Buffer[][],

Rule 2: If the accumulation of Buffer[][] is larger than a value *Sseg*, then add the value of Buffer[][] to Accumulator[][],

where *Dist* is a threshold value of the distance between two consecutive pixels, and *Sseg* is a threshold value of the length of the shortest line. With Rule 1, pixels with larger distances to one another will not be collected as line pixels, and with Rule 2, a piece of pixel component with only a small number of pixels will not be taken to be a true line segment. The detailed modified Hough algorithm is described as follows.

Procedure ModifiedHough

For each column in input binary image
  For each row in input image
    If pixel(x,y) is black, then
      For θ = θ1 to θ2 Compute
      $\rho = x\cos\theta + y\sin\theta$;
        If Distance(pixel(x, y), LastPixel[θ][ρ]) is smaller than *Dist*, then increase the value of Buffer[θ][ρ] by 1.
        Else
          If Buffer[θ][ρ] is larger than *Sseg*, then add they value of Buffer[θ][ρ] to Accumulator[θ][ρ];
          Set Buffer[θ][ρ] to 0;

  For each buffer in Buffer[θ][ρ]
    Check the value of Buffer[θ][ρ]. If it is not empty and is larger than *Sseg*, then add the value to Accumulator[θ][ρ];
End

In the above algorithm, θ1 is the starting angle, θ2 is the ending angle, of an orientation range in which line segments are detected. And Distance(p1, p2) is a function which returns the distance from pixel p1 to pixel p2. Because the length of consecutive line segments produced by noise coming from filled-in handwritten data or others is usually smaller than those of true lines, we can easily give values to thresholds *Dist* and *Sseg* to distinguish true line segments from noise. We use the following threshold values in our study: *Dist* = 1 and *Sseg* = 30. If the value of *Sseg* is set too large, then some short true line segments will be missed. On the other hand, if the value is not set large enough, noisy line segments will be detected.

B. Line Detection Algorithm

Line detection is an important basic work in form frame extraction. The proposed line detection algorithm is based on the modified Hough algorithm and can be used to find the candidates of true lines which come from form frames. The proposed line detection algorithm has two passes. Pass 1 is designed to find possible line candidates from the Hough counting space peaks larger than a proper threshold value, and Pass 2 is designed to find other candidates which might be missed by Pass 1, such as broken lines produced by poor scanning quality or image sources. An example is shown as Fig. 2, in which there is a line with the parameters $\theta = 90^\circ$ and $\rho = 10$. After we perform the modified Hough algorithm, the accumulation value of the line will be smaller than the threshold value. So, Pass 1 will miss the line. Since the pixels of the line distribute in ρ =10 and ρ = 9, the idea of Pass 2 is to get the vertical projection with two-pixel width, and this will collect the sum of the accumulation values of both ρ =10 and ρ = 9. Then, we check the projection values from left to right. If the number of the projection values which are larger than 0 is larger than the

threshold value, then we decide to have found a line. In this way the broken line Fig. 2 can be detected in Pass 2.

The first step before finding line candidates is to find the skew angle of lines. For example, if we want to detect horizontal lines, the orientation range of $\theta = 85^\circ$ through 95° is used to detect the skew angle of the horizontal lines. The algorithm for detecting the skew angle is as follows.

Procedure FindSkewAngle

For $\theta = 85^\circ$ to 95°
  For all ρ
    If Accumulator[θ][ρ] is larger than *Thf*/2,
      then add Accumulator[θ][ρ] to sum[θ];
    If sum[$\theta 0$] is the maximum, then $\theta 0$ is the skew angle of horizontal lines;
End

In the above algorithm, *Thf* is a threshold value, taken to be 0.95 times the length of the frame in this study. The idea of the algorithm is to find the angle of the largest number of line segments whose lengths are larger than about a half of the frame length, and take it to be the desired skew angle of the frame lines.

After we know the skew angle of the horizontal lines, we perform the following line detection algorithm to detect horizontal lines.

Procedure HorizontalLineDetection

Set $\theta 0$ = horizontal line skew angle detected by HorizontalLineDetection;
For all ρ
  Pass 1:
    If Accumulator[$\theta 0$][ρ] is larger than *Thf*, then
    Do FindLineWidth;
    Add ($\theta 0$, ρ) to horizontal line data structure;

  Pass 2:
    If Accumulator[$\theta 0$][ρ] + Accumulator[θ 0][ρ +1] is larger than *Thf*,then get the two-pixel width vertical projection which is vertical to $\theta 0$;
    Check the projection values from left to right;
    If the number of the projection values which is larger than 0 is larger than *Thf*, then Do FindLineWidth, and
      Add ($\theta 0$, ρ) to horizontal line data structure;
End

In the above algorithm, the FindLineWidth procedure checks neighboring lines which are attached to the detected line, and if it is found out that there exists a neighboring line segment whose length is large than *Thf*/2, then the neighboring line segment is merged to the detected line, and the width of the detected line is increased by 1. To complete the detection of the vertical frame lines, we also need a vertical line detection algorithm. The algorithm is similar to the horizontal line detection algorithm described above, and its detail is omitted here.

C. Top-Down Recursive Form Frame Detection Algorithm

Since most form frames are constructed with horizontal lines and vertical lines, we can extract form frames by extracting horizontal lines and vertical lines. There are several approaches to accomplishing this work. We can classify them into two kinds: top-down and bottom-up. The major difference between the two kinds of approaches is the structure of results. The results of the top-down approach are trees, and the results of bottom-up approach are linked lists or 1-D strings. We choose the top-down approach, and the reason is that we can build a form frame tree for an input form, which is useful for further steps of form understanding.

The basic idea of the proposed top-down recursive form frame detection algorithm is to find the outer frame first. Inside the outer frame is a subregion of the source image. We then perform the proposed line detection algorithm to find the horizontal lines and the vertical lines of the subregion with the subregion as an input. All the horizontal lines and the vertical lines detected will form several subregions. Then we perform the same algorithm again with these subregions as inputs recursively until no subregion remains. In this way, the frames are produced in the order of nodes of a tree, as shown in Fig. 3.

The top-down recursive form frame detection algorithm is described in detail as follows, in which the θ range for FindVerticalSkewAngle is from -5° through $+5^\circ$; the θ range for FindHorizontalSkewAngle is from 85° through 95° ; and FindOuterFrame is a

ugh 95° ; and FindOuterFrame is a procedure for finding the out-most frame of the input form image. Its detail is omitted.

Procedure TopDownFormFrame-Detection

```
  Do ModifiedHough
  Do FindVerticalSkewAngle
  Do FindHorizontalSkewAngle
  Do FindOuterFrame
  Do RecursiveFrameDetection
End
```

And the procedure RecursiveFrame-Detection used in the above top-down recursive form frame detection algorithm is described as follows.

Procedure RecursiveFrameDetection

```
  Do ModifiedHough
  Do HorizontalLineDetection
  Do VerticalLineDetection
  For each two neighboring horizontal lines
    For each two neighboring vertical lines
      Add a new subregion to subregion[] which
       is surrounded by the two
      horizontal lines and the two vertical lines

  For each subregion
     Do RecursiveFrameDetection
End
```

D. Masking Method For Broken Line Restoration

Most OCR systems cannot be used to recognize characters well with broken strokes. So the restoration of broken strokes is an important step after the segmentation of document images. Two examples of broken strokes after form frame extraction are shown in Fig. 4. The first example shows broken strokes which originally are touched by horizontal frames. The second example shows broken strokes which originally are touched by vertical frames.

The pixels in the touched area belong both to the frames and to the characters. The problem is to distinguish the stroke pixels from the pixels on the frames. After some observations, it can be found that the two ends of a stroke segment separated by a frame are almost in the same direction. So, we assume that when a stroke crosses a frame, the direction of the stroke will not change. For example, in the horizontal frame line case, we assume that there are three kinds of situations in which the stroke crosses the frame:

1. Crossing the horizontal frame line from the left-top side to the right-bottom side, as shown in Fig. 5(a).
2. Crossing the horizontal frame line from the upper side to the lower side, as shown in Fig. 5(b).
3. Crossing the horizontal frame line from the right-top side to the left-bottom side, as shown in Fig. 5(c).

The work we need to do is to check if a pixel on the horizontal frame line is on one of the three tracks. If so, then decide that the pixel belongs to the stroke. The work for the vertical frame line case is similar.

We propose three types of masks for the horizontal case and three other types of masks for the vertical case, as shown in Fig. 6. Because the input image is binary, we assume that the gray value of each black pixel is 0, and the gray value of each white pixel is 255. The input image is kept in an array, and each cell in the array is 8 bits. The gray values of the pixels which are detected by the form frame detection algorithm are taken to be 128.

We perform the AND operation on each pixel of the frame with all the three types of masks. Let S be the summation of the results of each AND operation with a mask. If S is equal to 0, then replace the gray values of the pixels which are marked with 128 in the masks with 0. The meaning of this rule is that if the pixel in the middle cell of each of the masks is a frame pixel, and the two end pixels are black, then the middle pixel is regarded as a stroke pixel.

E. Removal of Form Frames

After the above restoration, the pixels that belong to the frame need be removed. It is quite simple to remove the frame, because they are located and marked by the form frame detection algorithm. But there are still some noise pixels which are attached to the frame and are not removed. An example is shown in Fig. 7. So, we propose two rules to remove the frame and all noise pixels, as described in the following.

Rule 3: If the length of a vertical black run which is attached to horizontal frame lines is smaller than Tr , then remove the run.

Rule 4 : If the length of a horizontal black run

which is attached to vertical frame lines is smaller than *Tr*, then remove the run.

In the above rules, *Tr* is a threshold value which is taken to be 2 in this study.

F. Correction of Skew Angles

A scanned clinic data image is usually skewed. Most of the postprocessing works after segmentation, including OCR, cannot accept skewed segmentation results. So, the correction of skew angles is necessary. After form frame extraction is performed, we already know the vertical skew angle θv and horizontal skew angle θh . All we need to do is to rotate the image back. The following transform formula is used for this purpose:

$$x' = x \cos\theta + y \sin\theta$$
$$y' = -x \sin\theta + y \cos\theta$$

where θ is taken to be $\theta = (\theta v + (\theta h - 90^{\circ}))/2$ in this study.

The frames extracted by the form frame detection algorithm also need be corrected, because they are extracted in the skew angle. In form frame extraction, we keep the skew angle and two end points for each frame. Since we have the skew angle and two end points, we can rotate the end points of skewed frames by applying the above formula.

3.Form Content Extraction

After the form frame is removed, we perform the work of form content extraction, which includes the steps of component extraction and in-field line detection. The details are described in the following.

A. Component Extraction by Region Growing

In this study, we apply region growing for form component extraction. For a binary image, a seed pixel is needed for region growing and it should be a black pixel. All 8-neighboring black pixels of the seed pixel are grown together. Two situations might arise after region growing. The first is that there are several characters in a connected component. This situation can be accepted because it can be solved by postprocessing, such as character string recognition. The second situation is that an extracted connected component is only part of a character. This often occurs in Chinese characters. We show an example of the second situation in Fig. 8(b). This kind of results cannot be used as input to an OCR system because most OCR systems cannot deal with partial characters. So merge of connected components of this kind is necessary. In this study, we apply the morphological closing operation to the connected components resulting from the region growing step. The closing operation will merge those components which are sufficiently close to one another, as shown in Fig. 8(c).

After the morphological closing operation, some characters with large font sizes might still be separated into several connected components, like the example shown in Fig. 9(b), because of the large gaps between the components. After some observations, we found that if we draw a minimum rectangular shape to enclose each connected component, then the center point of each isolated component of a character with large font size is within the circumscribing rectangles of the other components of the character. For this reason, we use the following rule to merge all the isolated connected components which belong to one character but cannot be merged by the morphological closing operation:

Rule 5: If the central point of the connected component box is within the box of another connected component, then merge the two connected components.

B. In-field Line Detection

In some clinic forms, there are some lines which are not the components of form frames, but are part of the contents of certain form fields, such as underlines or others. An example is shown in Fig 10(a). These lines cannot be extracted by the form frame detection algorithm. Before classifying the form content components, these lines should be extracted in advance. The in-field line detection algorithm we use for this purpose is very similar to that for form frame extraction. The only difference is that the input to this algorithm is a connected component, not a subregion of the image. In more detail, we perform line detection on each connected component extracted by the region growing method. If there are lines in the connected component, the line detection algorithm will find them. And as before, we restore broken strokes after the line detection step. For Fig. 10(a), the result of these steps are shown in Fig. 10(b). All the lines

detected are added to the line structure of the data form.

4.Form Content Classification

In this section, we describe how form content components extracted from the previous steps are classified. We first describe the proposed features for classification. Then, structure of the back propagation neural network employed for the classification work is described, followed by the descriptions of the learning and classification steps using the network.

A. Features of Preprinted and Filled-in Components for Classification

The features we use for form content classification can be categorized into two types: one is statistical, and the other structural. These features are described in the following, in which the first four are statistical, and the last is structural.

(1) The aspect ratio of a connected component:

$$\text{Feature 1} = \frac{\text{height of connected component}}{\text{width of connected component}}$$

(2) The density of a connected component:

$$\text{Feature 2} = \frac{\text{number of black pixels}}{\text{number of pixels of connected component}}$$

(3) The density of vertical or horizontal pixels of a connected component:

$$\text{Feature 3} = \frac{\text{number of vertical or horizontal pixels}}{\text{number of pixels of connected component}}$$

where vertical and horizontal pixels are defined in the following way: for each black pixel in the connected component, compute the number of vertical or horizontal pixels by the four AND masks as shown in Fig. 11; if the value of one of these masks is equal to zero, then the pixel is regarded as a vertical pixel or a horizontal pixel.

(4) The balance degree of a connected component:

$$\text{Feature 4} = \frac{\text{horizontal distance from } C_b \text{ to } C_p}{\text{width of connected component}} + \frac{\text{vertical distance from } C_b \text{ to } C_p}{\text{height of connected component}}$$

where C_b is the center of the box enclosing the connected component, and C_p is the central point of the connected component pixels.

(5) The touch of a connected component with the form frame:

In this system, only one structural feature is used, which is binary and indicates whether a connected component touches the form frame. We call this feature the fifth feature.

If a connected component touches the form frame as indicated by the feature 5, then this connected component must be a filled-in data component. On the other hand, if the connected component does not touch the form frames, then we extract the four statistical features mentioned previously and use them as input to the back propagation neural network to classify the connected component.

B. Classification by Back Propagation Neural Network

There are two reasons why we use the back propagation network for form component classification: it's short response time of classification and it's capability of auto-analysis of input features, although it takes longer time to complete the learning process, and it is sensitive to the quality of training samples. In the back propagation network, we use four input nodes in the input layer, seven nodes in the hidden layer, and two nodes in the output layer. The number of input nodes is decided by the number of features. The number of hidden nodes is decided by experience in the learning process. And the number of output nodes is decided by the number of classes: the preprinted component class (class 1), and the filled-in handwritten component class (class 2). All the connections between the input layer and the hidden layer, and those between the hidden layer and the output

layer, are associated with weights of the network.

In the input layer, we use the linear function $f(x) = x$ to transform the values of features into the values of input nodes. In the hidden layer, we use the sigmoid function to transform the values of the linear combination of the input layer nodes into the values of the hidden layer nodes. In the output layer, we use the sigmoid function to transform the values of linear combinations of the hidden layer nodes into the values of the output layer nodes, too.

C. Learning Process for Neural Network

Before we use the back propagation network for classification, we have to train it with given sample data. In the learning process, the output value of the jth node of the nth layer A_j^n is a non-linear function of the $(n - 1)$th nodes:

$$A_j^n = f(net_j^n)$$

where net_j^n is the summation function:

$$net_j^n = \sum_i W_{ij} A_i^{n-1} - \theta_j$$

in which W_{ij} is the weight of the connection from the ith node of layer $n - 2$ to the jth node of layer $n - 1$, and θ_j is the bias of the jth node of layer $n - 1$. And $f(x)$ is a transfer function, for which we use the sigmoid function. The energy function we use is:

$$E = \frac{1}{2}\sum_j (T_j - A_j)^2$$

where T_j is the desired output value of the output layer, and A_j is the inferred output value of the output layer. In the learning process, we perform the following steps.

Step 1. Initialize the parameters, weights, and biases. The parameters include the learning rate η and a momentum term α. We let η = 10 and α = 0. The initial weights are taken to be random numbers generated by a random number generator.

Step 2. Start a training cycle in which the training of a batch of training examples is performed.

Step 3. Input a training example. Let X_1 = feature 1, X_2 = feature 2, X_3 = feature 3, X_4 = feature 4 and let desire_Y_1 = 1, desire_Y_2 = 0 if the training data is an element of class 1, or let desire_Y_1 = 0, desire_Y_2 = 1 if the training data is an element of class 2.

Step 4. Compute Y_1, Y_2, and E as follows:

$$net_h = \sum_i W\_xh_{ij} \cdot X_i - \theta\_h_h$$

$$H_h = f(net_h) = \frac{1}{1 + e^{-net_h}}$$

$$net_j = \sum_h W\_hy_{hj} \cdot H_h - \theta\_y_j$$

$$Y_j = f(net_j) = \frac{1}{1 + e^{-net_j}}$$

$$E = \frac{1}{2}\sum_j (desire\_Y_j - Y_j)^2$$

where (1) W_xh are the weights between the input layer and the hidden layer, (2) W_hy are the weights between the hidden layer and the output layer, (3) θ_h are the biases of the hidden layer, and (4) θ_y are the biases of the output layer.

Step 5. Compute δ values by

$$\delta_j = Y_j \cdot (1 - Y_j) \cdot (T_j - Y_j)$$

$$\delta_h = H_h \cdot (1 - H_h) \cdot \sum_j W\_hy_{hj} \delta_j .$$

Step 6. Compute ΔW values by the following steps.

(a)Compute ΔW_hy and $\Delta\theta\_y$:

$$\Delta W\_hy_{hj} = \eta \cdot \delta_j \cdot H_h$$

$$\Delta\theta\_y_j = -\eta \cdot \delta_j .$$

(b)Compute ΔW_xh and $\Delta\theta\_h$:

$$\Delta W\_xh_{ih} = \eta \cdot \delta_h \cdot X_i$$

$$\Delta\theta\_h_h = -\eta \cdot \delta_h .$$

Step 7. Add the weights and biases of the training example in the cycle.

(a) Add the weights and biases of the output layer:

$$\Delta W\_hy_{hj} = \Delta W\_hy_{hj} + \Delta W\_hy_{hj}$$

$$\Delta\theta\_y_j = \Delta\theta\_y_j + \Delta\theta\_y_j .$$

(b) Add the weights and biases of the hidden laye:.

$$\Delta W\_xh_{ih} = \Delta W\_xh_{ih} + \Delta W\_xh_{ih}$$

$$\Delta\theta\_h_h' = \Delta\theta\_h_h' + \Delta\theta\_h_h .$$

And repeat **Step 3** to **Step 7** until the cycle ends.

Step 8. Update the weights and biases:

$$W\_hy_{hj} = W\_hy_{hj} + \frac{\Delta W\_hy_{hj}}{n}$$

$$\theta\_y_j = \theta\_y_j + \frac{\Delta\theta\_y_j}{n}$$

$$W\_xh_{ih} = W\_xh_{ih} + \frac{\Delta W\_xh_{ih}}{n}$$

$$\theta\_h_h = \theta\_h_h + \frac{\Delta\theta\_h_h}{n}.$$

Repeat **Step 2** through **Step 8** until the average value of E is smaller than a threshold value T_e.

We train the network in the way of batch learning. There are 200 training examples in one cycle. The threshold value T_e is 0.02. After performing these steps above, we keep the weights and biases in files.

D. Classification Process

After learning, we can use the network for classification according to the following steps.

Step 1. Read the weights and biases from files.

Step 2. Input a test data. Let X_1 = feature 1, X_2 = feature 2, X_3 = feature 3, X_4 = feature 4.

Step 3. Compute Y_1 and Y_2 by the following:

$$net_h = \sum_i W\_xh_{ij} \cdot X_i - \theta\_h_h$$

$$H_h = f(net_h) = \frac{1}{1+e^{-net_h}}$$

$$net_j = \sum_h W\_hy_{hj} \cdot H_h - \theta\_y_j$$

$$Y_j = f(net_j) = \frac{1}{1+e^{-net_j}}.$$

Step 4. If Y_1 is larger than Y_2, then the input data is decided to belong to class 1. Otherwise, the input data is decided to belong to class 2.

5. Experimental Results

Many clinic data images were tested by the proposed approach on a pentium-133 PC using the Visual C++ language. The tested images were obtained by scanning with an Umax vista s-8 color scanner at 200 dpi with 256 gray levels. To demonstrate the performance of the back propagation network model for classification, 386 test components were extracted by the system and classified. The classification rate was 94.5%. The errors mainly come from inappropriate density values in the extracted character components. Some experimental results of segmentation form images and classification of form content components are shown in Fig. 12. The average times of some intermediate processing steps are shown in Table 1.

Table 1. Computation times of some intermediate processing steps.

Processing step	Time
Form frame extraction	10 sec.
Form content enhancement (broken stroke restoration, frame removal, skew angle correction)	3 sec.
Form component extraction and in-field line detection	15 sec.

6. Conclusions

A clinic document image analysis system useful for data form segmentation and form content classification has been successfully implemented. Several major achievements in different processing stages of the system are summarized as follows.

1. In the stage of form frame extraction, a modified Hough algorithm for more effective line detection and a corresponding top-down recursive form frame detection algorithm have been proposed. These algorithms can be used to extract form frames with noise.

2. In the stage of form content enhancement, a masking method has been proposed for more effective restoration of broken strokes. A simple frame skew angle detection method has also been proposed.

3. In the stage of form component extraction, a component extraction algorithm using some component merging and in-field line detection techniques have been proposed. The algorithm is effective for extracting Chinese characters which are often composed of several parts.

4. In the stage of form component classification, several new and effective features are proposed and the back propagation neural network is used effectively for classification of fill-in components and pre-printed characters. The results of

form component classification is very useful for the further step of form understanding.

The experimental results have revealed the feasibility of the proposed algorithms.

References

[1] T. M. Lu and K. C. Fan, "Form segmentation and recognition by clustering of feature points and matching of feature graphs," *Proc. 1994 Workshop of Center of Excellence for Computer System Technology at National Chiao Tung University*, Hsinchu, Taiwan, pp. 120-133, 1994.

[2] W. Li, J. Hong, A. Zhang, and B. Chen, "A atatistical form reading system," *Proc. 1993 TENCON*, Vol. 2. pp. 1062-1065, 1993.

[3] S. W. Lam, L. Javanbakht, and S. N. Srihari, "Anatomy of a form reader," *Proc. 2nd International Conf. Document Analysis and Recognition*, pp. 506-509.

[4] B. Yu and A. Jain, "A generic system for form dropout," *IEEE Trans. PAMI*, Vol. 18, No. 11, pp. 1127-1134, 1996.

[5] Maderlechner, "Symbolic subtraction from fixed formatted graphics and text from filled in forms," *Machine Vision and Applications*, Vol. 3, pp. 457-459, 1990.

[6] R. Casey, D. Ferguson, K. Mohiuddin, and E. Walach, "Intelligent forms processing system," *Machine Vision and Applications*, Vol. 5, pp. 143-155, 1992.

[7] D. Doermann and A. Rosenfeld, "The interpretation and reconstruction of interfering strokes," *Proc. International Workshop on Frontiers in Handwriting Recognition*, pp. 41-50, 1993.

[8] F. M. Wahl, K. Y. Wong, and R. G. Casey, "Block segmentation and text extraction in mixed text/image document," *Computer Graphics and Image Processing*, Vol. 20, pp. 357-390, 1982.

[9] D. Wang and S. N. Srihari, "Classification of newspaper image blocks using texture analysis," *Computer Vision, Graphics, and Image Processing*, Vol. 47, pp. 327-352, 1989.

[10] L. A. Fletcher and R. Kasturi, "A robust algorithm for text string separation from mixed text/graphics images," *IEEE Trans. PAMI*, Vol. 10, pp. 910-918, 1988.

[11] S. N. Srihari, Y. C. Shin, and V. Ramanaprasad, "A system to read names and addresses on tax forms ," *Proc. IEEE*, Vol. 84, No. 7,pp.10381049,1996.

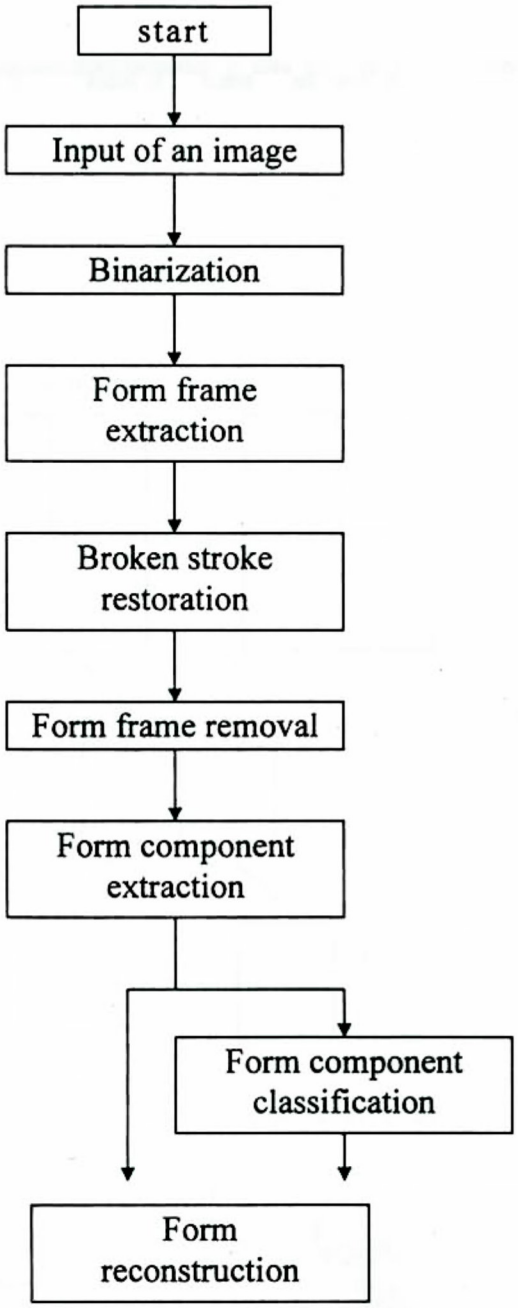


Fig. 1 Flowchart of the proposed clinic data image analysis system.

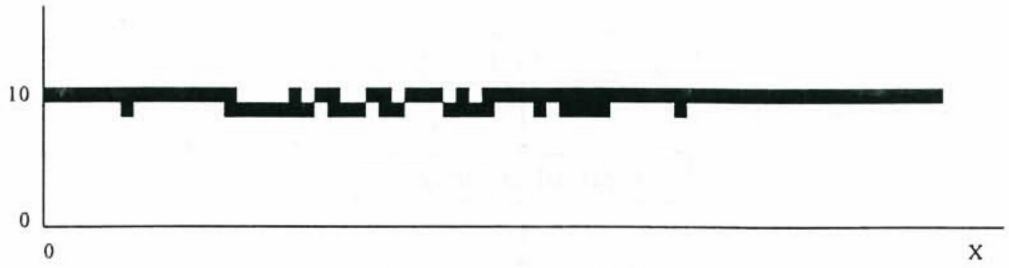


Fig. 2 A broken line.

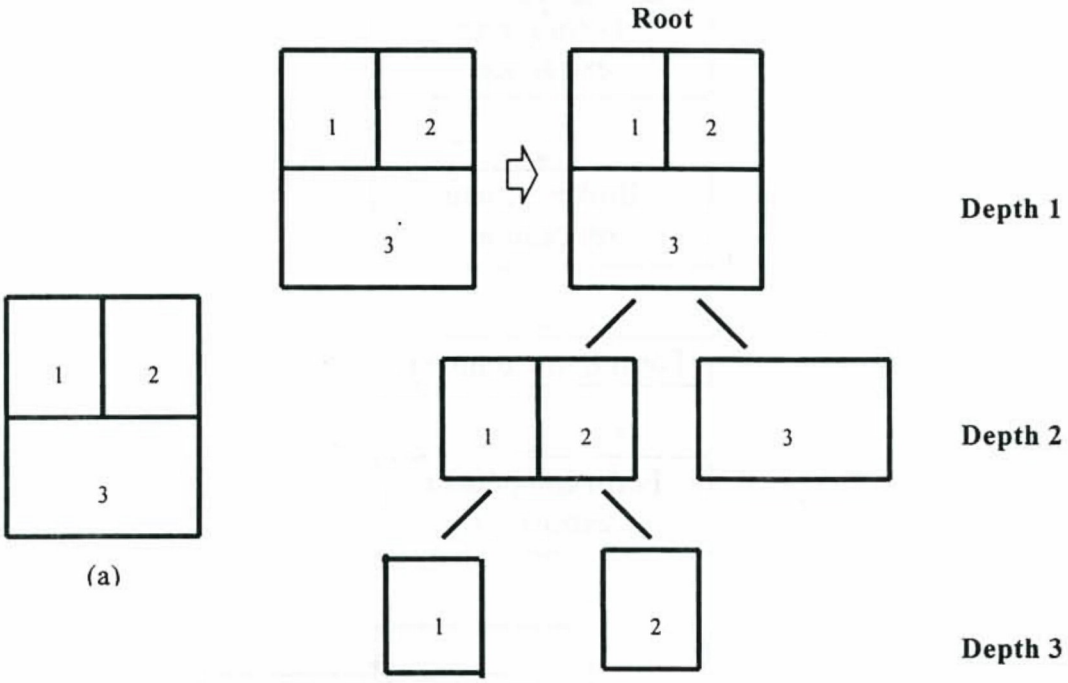


Fig. 3 The tree structure of frame detection results. (a) Input image. (b) Frame tree.

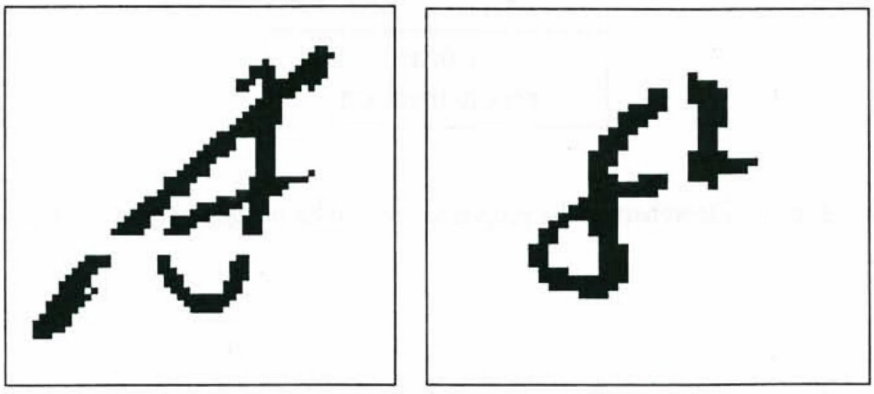
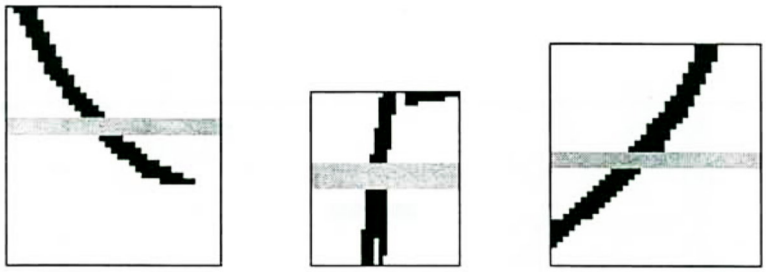
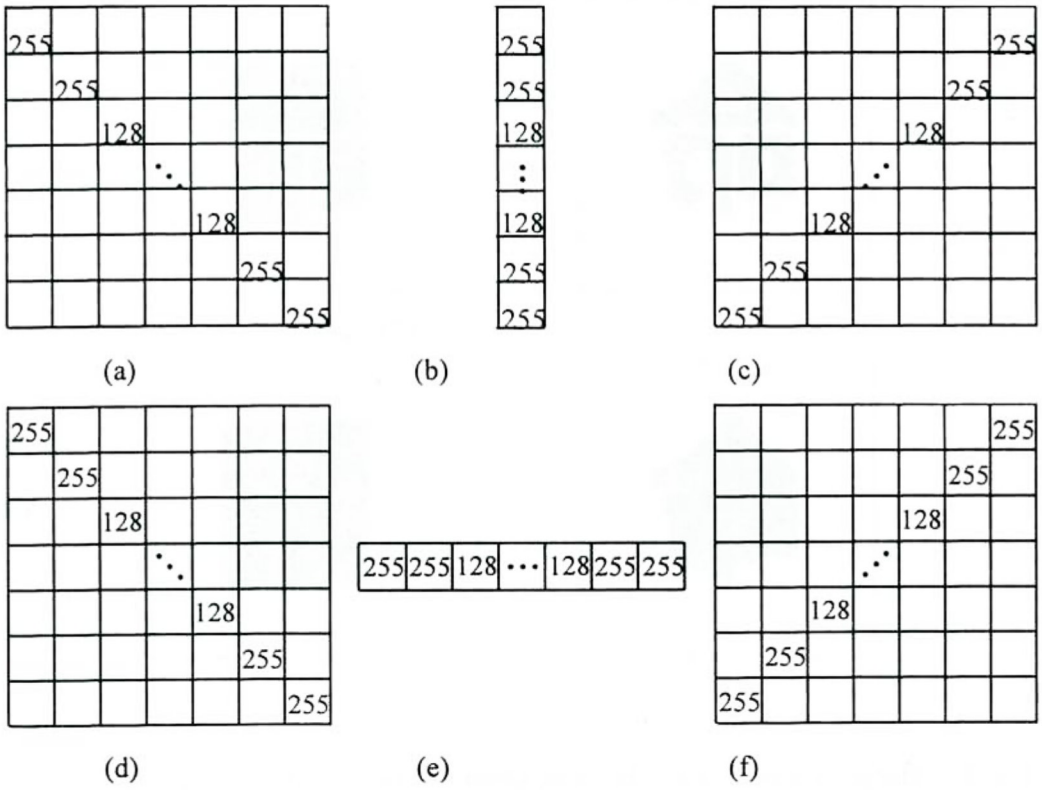


Fig. 4. Examples of broken strokes. (a) Touched by a horizontal frame. (b) Touched by a vertical frame.



(a) (b) (c)

Fig. 5. Three situations in which a stroke crosses a horizontal frame. (a) Crossing horizontal frame from left-top side to right-bottom side. (b) Crossing horizontal frame from upper side to lower side. (c) Crossing horizontal frame from right-top side to left-bottom side.



(a) (b) (c)

(d) (e) (f)

Fig. 6. Three types of restoring masks for the horizontal case and three types of restoring masks for the vertical case. (a) The mask for horizontal state 1. (b) The mask for horizontal state 2. (c) The mask for horizontal state 3. (d) The mask for vertical state 1. (e) The mask for vertical state 2. (f) The mask for vertical state 3.

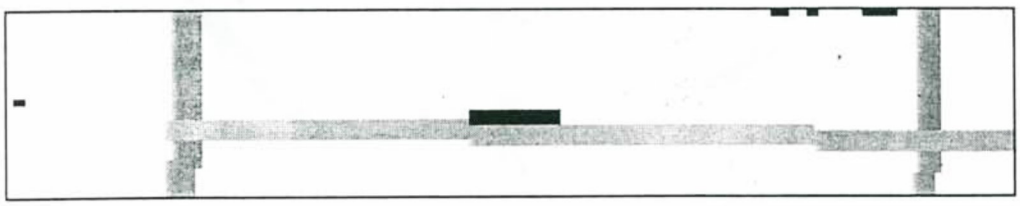


Fig. 7. The gray pixels belong to form frames, and black pixels which are attached to the horizontal frame are noise that also need be removed.

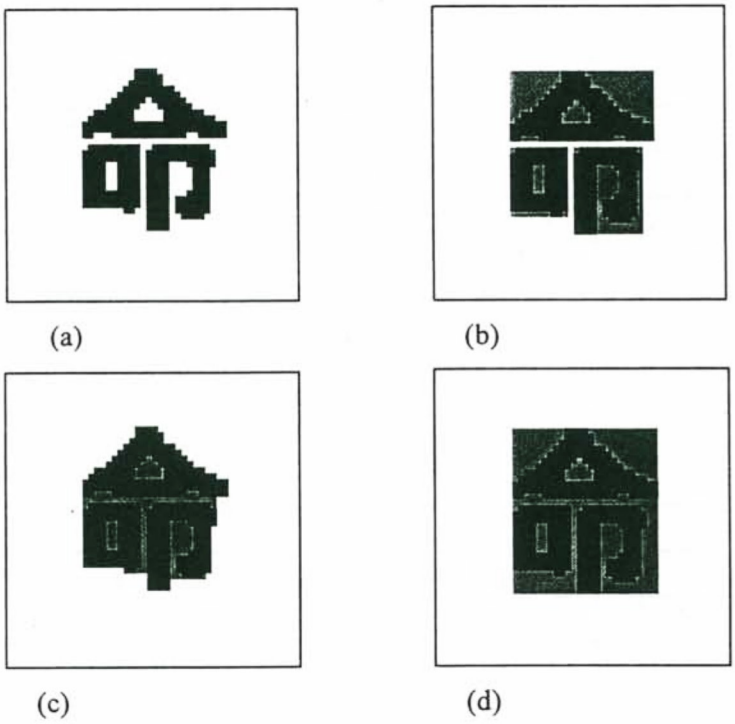


Fig. 8. Merge by morphological closing operation. (a) Source image, a Chinese character. (b) Several connected components, the result of region growing. (c) The result of merging by morphological closing operation. The size of the structure element for the operation is 5×5. (d) The single connected component resulting from merging.

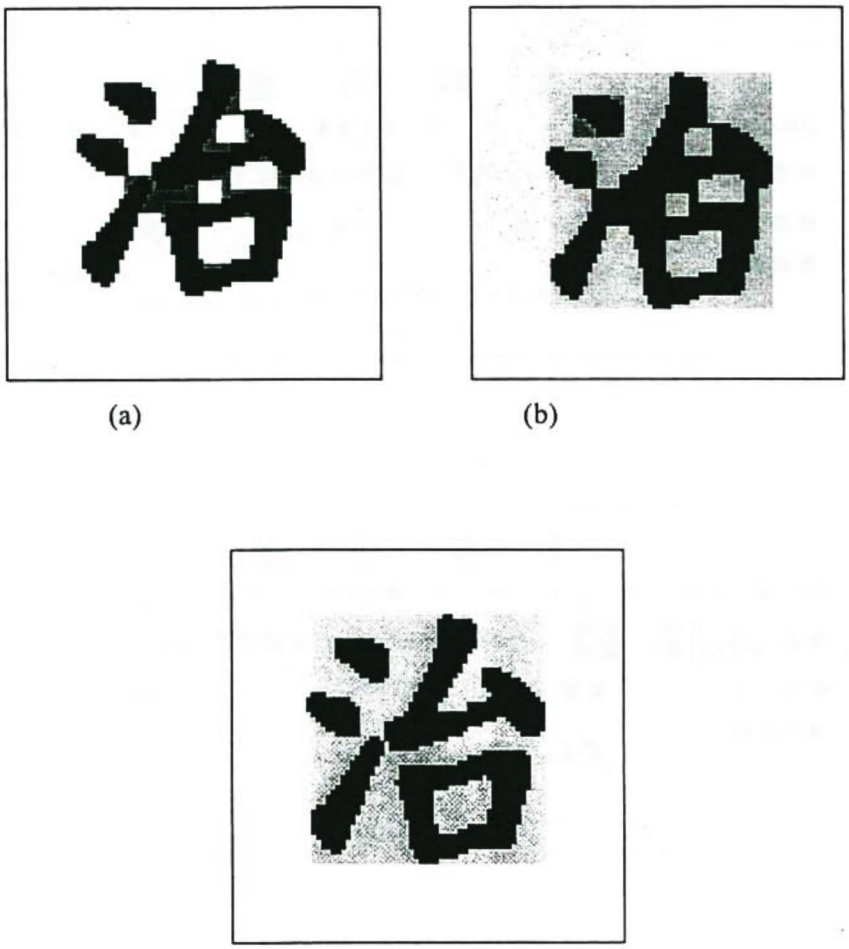
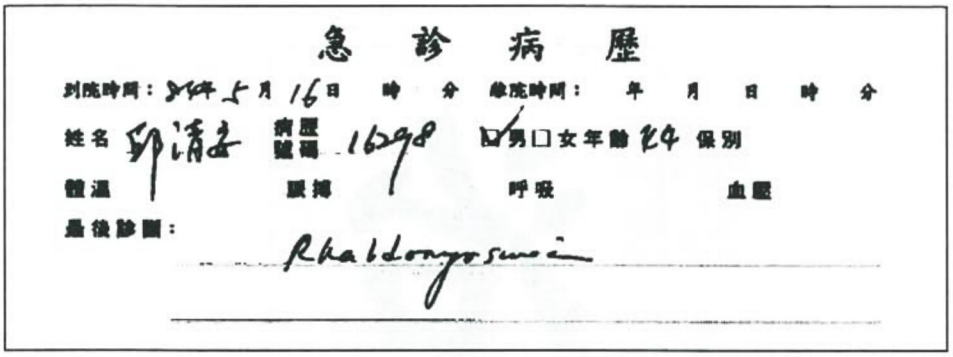


Fig. 9. Merge of components which belong to a single Chinese character with large font size. (a) The result of region growing. (b) The result of component merge by close operation. There are still two connected components. (c) The result of merging by the proposed rule.

(a)



(b)

Fig. 10. In-field line detection.. (a) The result of frame detection.(b) The result of in-field line detection and restoration of broken strokes.

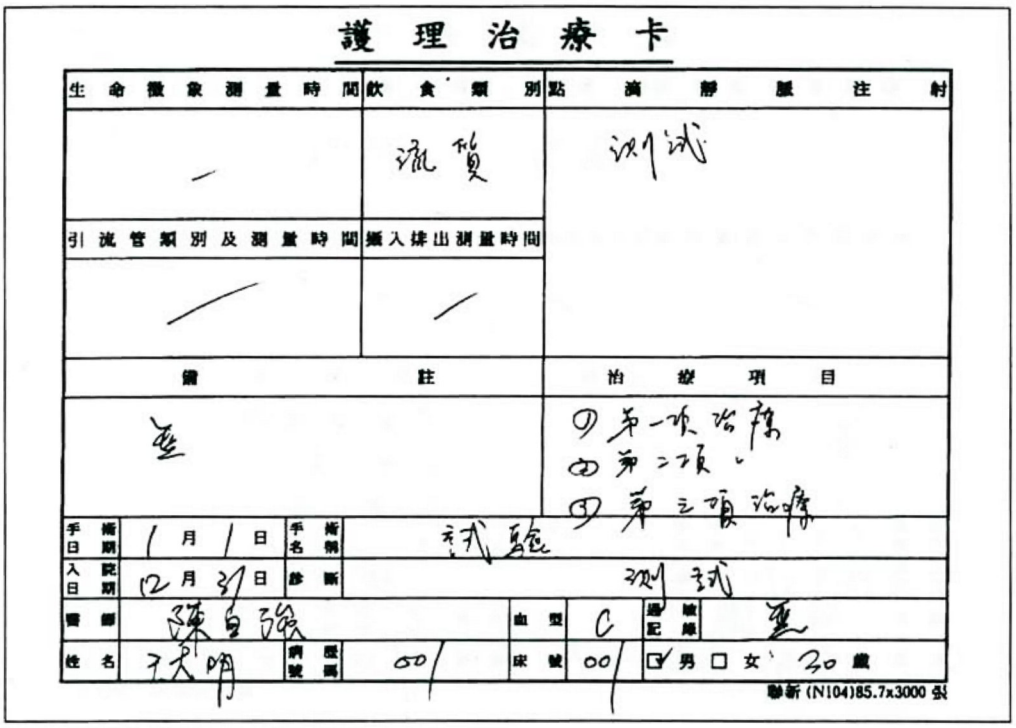
x	255	0
x	255	0
x	255	0

0	255	x
0	255	x
0	255	x

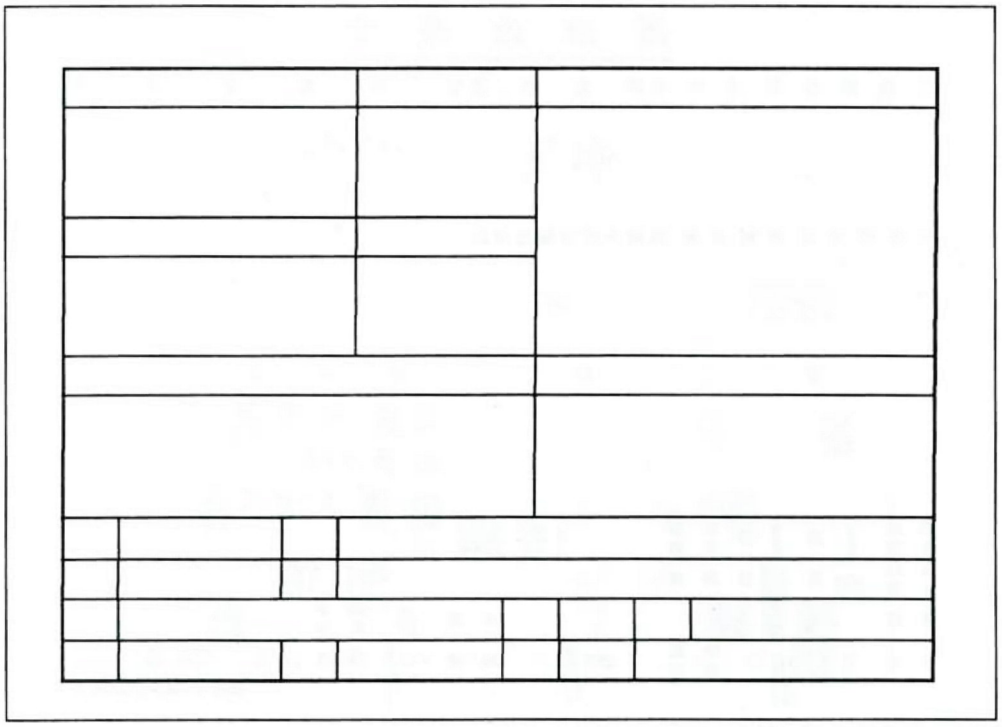
0	0	0
255	255	255
x	x	x

x	x	x
255	255	255
0	0	0

Fig. 11 . Four masks used to check vertical and horizontal pixels.

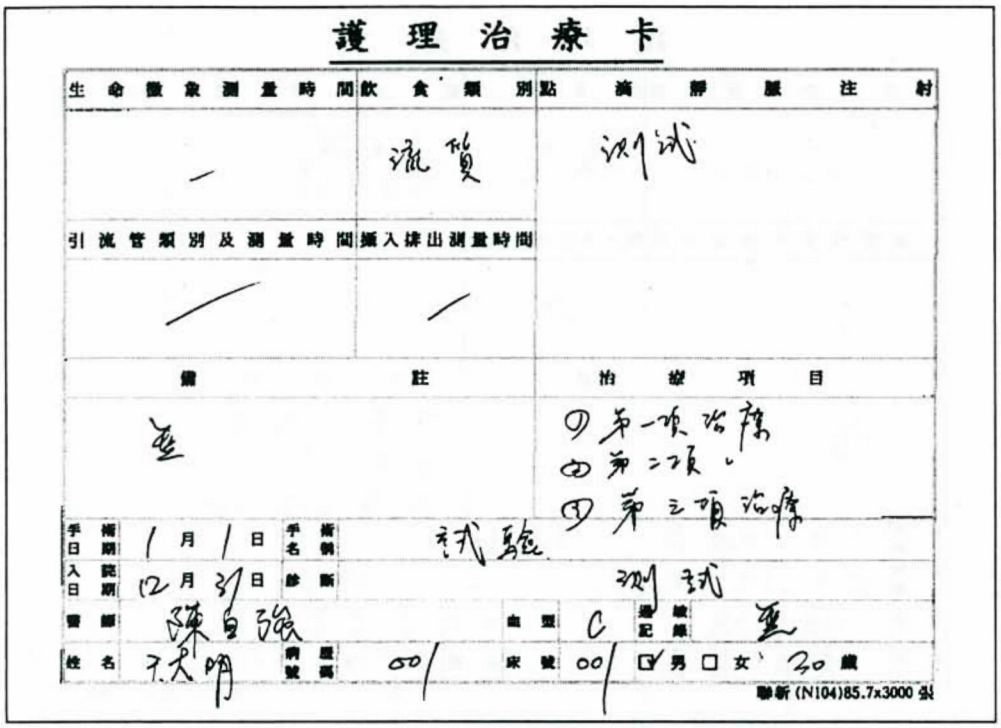


(a)

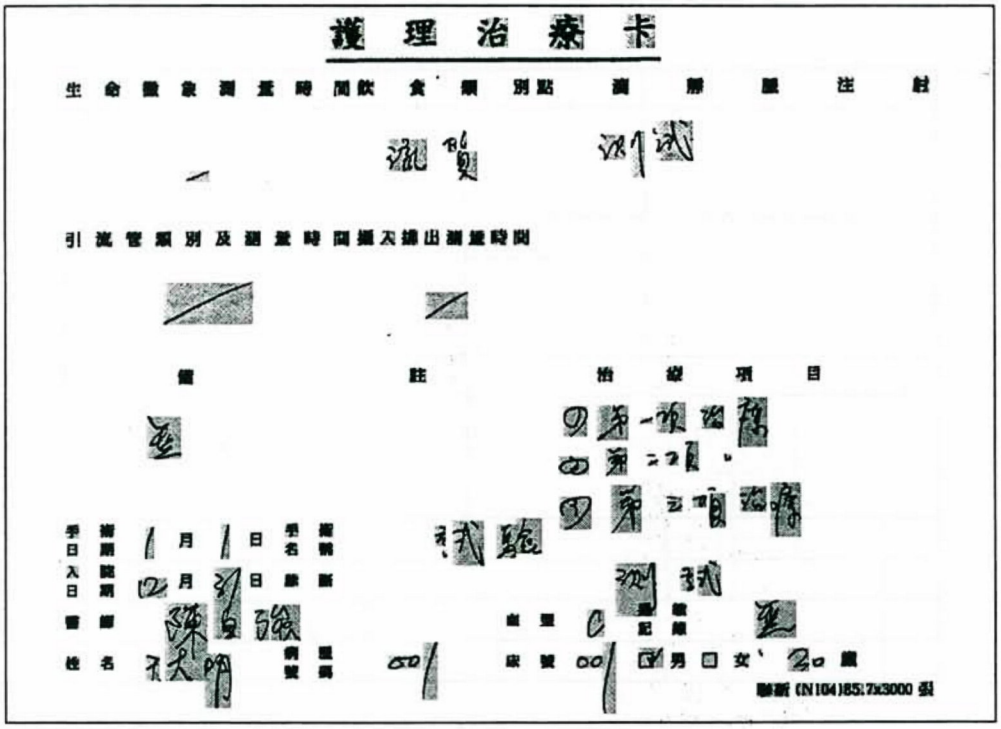


(b)

Fig. 12. An example of experimental results. (a) The source image after binarization. (b) The form frame extraction result. (c) The broken stroke restoration result. (d) The form component extraction result. (e) The in-field line detection and restoration result. (f)The classification result. The gray blocks are filled-in components.

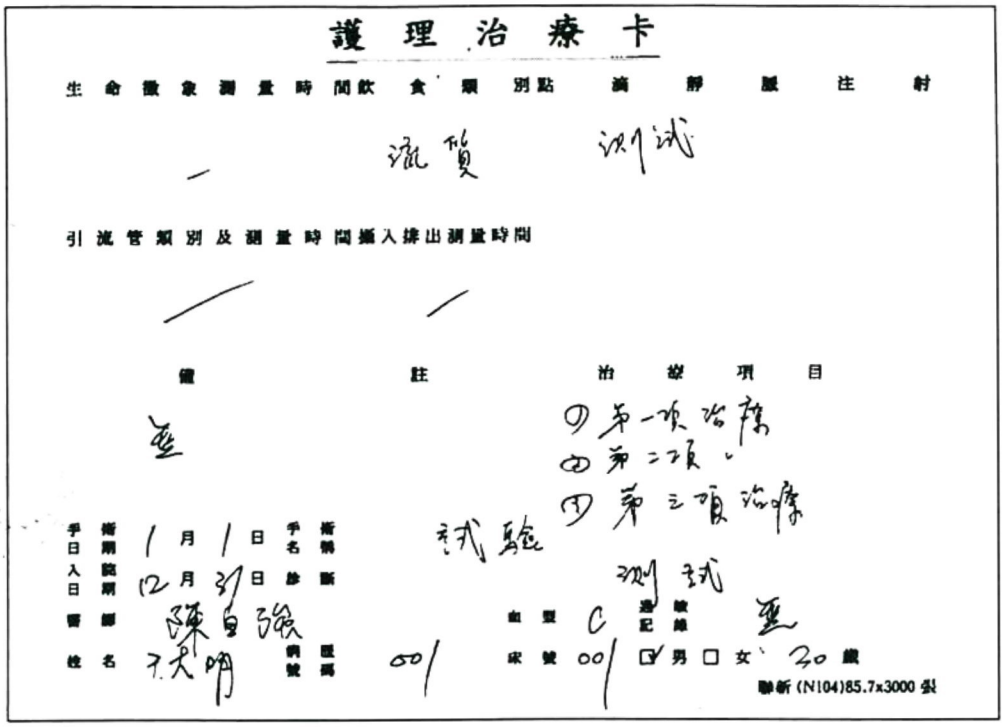


(c)

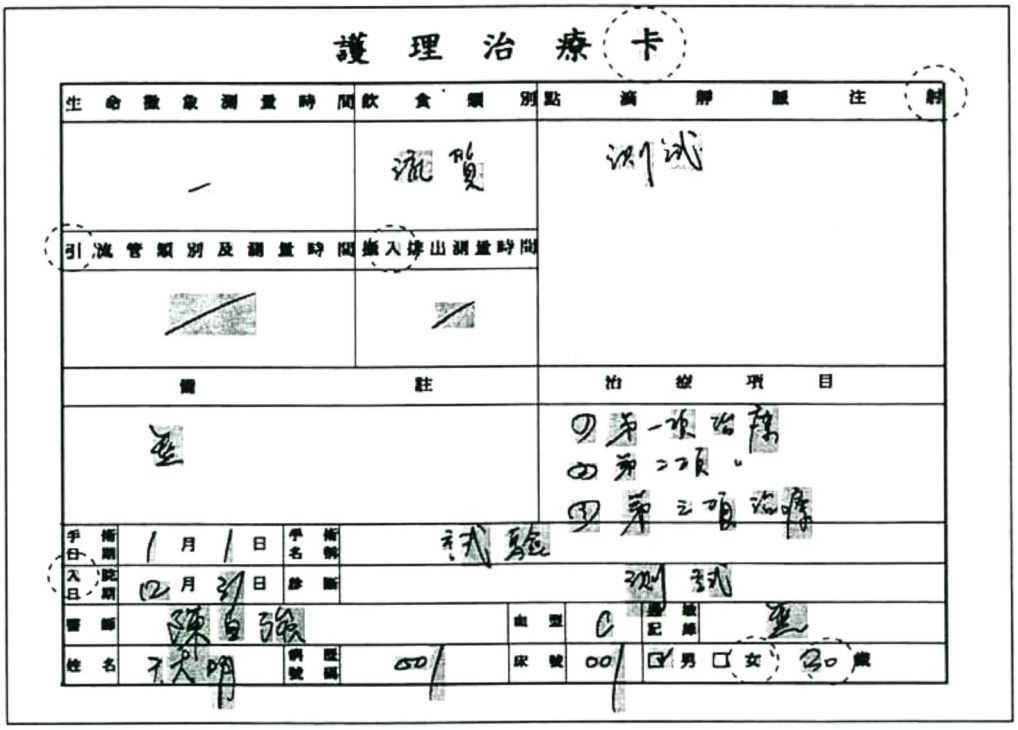


(d)

Fig. 12. An example of experimental results (continued). (a) The source image after binarization. (b) The form frame extraction result. (c) The broken stroke restoration result. (d) The form component extraction result. (e) The in-field line detection and restoration result. (f)The classification result. The gray blocks are filled-in components.



(e).



(f).

Fig. 12. An example of experimental results (continued). (a) The source image after binarization. (b) The form frame extraction result. (c) The broken stroke restoration result. (d) The form component extraction result. (e) The in-field line detection and restoration result. (f)The classification result. The gray blocks are filled-in components.