Pergamon

*Pattern Recognition*, Vol. 31, No. 9, pp. 1353–1368, 1998
© 1998 Pattern Recognition Society. Published by Elsevier Science Ltd
All rights reserved. Printed in Great Britain
0031-3203/98 $19.00 + 0.00

PII: S0031-3203(97)00156-8

# AN EFFICIENT ALGORITHM FOR FORM STRUCTURE EXTRACTION USING STRIP PROJECTION

JIUN-LIN CHEN and HSI-JIAN LEE*

Department of Computer Science and Information Engineering, National Chiao Tung University, Hsinchu, Taiwan 30050, R.O.C.

(*Received* 13 *November* 1997)

**Abstract**—This paper presents an efficient strip-projection-based approach to extracting form structures from form documents for office automation. To locate the data, we have to extract and interpret the form structure. In this paper, a strip projection method is presented for extracting the form structure. We first segment input form images into uniform vertical and horizontal strips. Since most form lines are vertical or horizontal, we project the image of each vertical strip horizontally and that of each horizontal strip vertically. The peak positions in these projection profiles denote possible locations of lines in form images. We then extract the lines starting with the possible line positions in the source image. After all lines have been extracted, redundant lines are removed using a line-verification algorithm and broken lines are linked using a line-merging algorithm. Experimental results show that the proposed method can extract form structures from A4-sized documents in about 3 seconds which is very efficient, compared with the methods based on Hough transformation and run-based line-detection algorithms. © 1998 Pattern Recognition Society. Published by Elsevier Science Ltd. All rights reserved

Form-document processing  Strip projection  Line-detection and verification  Field extraction

## 1. INTRODUCTION

Form documents are widely used for many purposes such as application forms, order forms, archival forms and medical forms. Processing such form documents is usually a routine task in many business and government organizations. Computers can be used to process, transfer and store the form documents. To process form documents electronically, we have to digitize them and extract the data we want. In order to extract the relevant data from the form, we have to interpret the form structure. In this paper, we propose an efficient approach to extracting the form structures of input form documents using strip projection.

Several form document processing systems have been presented in recent years.[1–8] Casey, Ferguson, Mohiuddin and Walach[1] developed an intelligent form-processing system for form recognition. The form model they used for form recognition consisted of information about the locations and lengths of significant lines, the positions and sizes of fields, and offset and skew information. Watanabe, Luo and Sugie[2] proposed an analytical method for recognizing the layout structures of table-form documents. In their approach, form structures were described according to mutual relationships among line segments. A global binary tree representing neighboring relationships among horizontal and vertical line segments, and a local binary tree containing detailed information about individual blocks were constructed for form recognition. Taylor, Fritzson and Pastor[3] presented a system for manipulating preprinted forms. They used 10 corners as features for form registration and field description. In their system, a form image was first partitioned into nine equal rectangular blocks, and the numbers of each type of corner in each block were counted. The feature vectors were then fed into a neural network for recognition. Fujisawa, Nakaro and Kurino[4] presented a pattern-oriented segmentation method for separating touching handwritten characters. Two types of form frames, label frames and data frames, were extracted. By recognizing characters inside label frames and identifying the semantic relationships between label frames and data frames, a frame interrelationship matrix was generated to recognize the form. Fan, Lu, Wang and Liao[6] presented a clustering-based form-recognition method. In their approach, feature points were clustered to separate characters from form documents by creating graphs for form document recognition using a graph-matching method.

The purpose of form-structure extraction is to determine the construction lines in a particular form using image-analysis methods. A number of techniques such as Hough transformation, connected-component analysis, and contour detection and tracing with line-verification algorithms have been proposed. Hough transformation is widely used for detecting lines in arbitrary directions, as are contour detection and tracing. These methods have a major weakness: they are too time-consuming. Form-structure extraction is the very first step in form document

*Author to whom correspondence should be addressed.

processing systems; thus practical systems should spend little time on it. Since most form lines are horizontal and vertical, algorithms that can detect lines in arbitrary directions are not necessary in form-processing systems. Connected-component analysis has also been used to detect horizontal and vertical line segments according to the lengths and aspect ratios of components in some studies.[8] Such methods can detect horizontal and vertical lines much more efficiently than the Hough transform and contour-detection methods do, since they focus only on horizontal and vertical lines, however they still spend too much time detecting connected components. Another problem with connected-component analysis is that it cannot handle lines touching characters or other data. Ho and Chen[9] proposed a modified high-speed Hough transform method for extracting arbitrary lines from images by shifting the source images, but their method cannot handle horizontal and vertical lines directly.

Projection is the simplest and fastest way to find horizontal and vertical lines in a given image, since such lines produce peaks in projection profiles. Even if the lines touch other data or are broken, we can still determine their locations from the projection profiles. But projection of whole images can only be used to detect long lines. Shorter lines will not produce high enough peaks in the projection profiles, and thus, will be lost. In this paper, we propose a strip-projection method for extracting form structures from form documents that projects individual strips rather than entire images. The system diagram is shown in Fig. 1.

There are six major steps in this proposed method. Preprocessing removes noise and performs binarization. We use the Niblack's[10,11] method for binarizing input form images. After preprocessing, we detect the bounding rectangles in the input form images in order to remove the surrounding empty spaces. This allows us to cut strips exactly on the image of the input form. Next, we divide the form image into $D$ uniform strips in the horizontal and vertical directions, and perform horizontal projection of vertical strips and vertical projection of horizontal strips. From the projection results, we can determine possible line locations in each strip by applying a threshold to the projection results, then choose start points for line tracing according to these possible locations.

We then trace horizontal lines from those start points detected in vertical strips, and trace vertical lines from start points detected in horizontal strips. We apply a verification procedure to each extracted line to determine whether it is a proper line or not. After this, we merge broken lines into longer lines, then use these lines to extract form fields and remove superfluous and false lines. We then have obtained fields from which to construct the form, and lines with which to construct the fields comprising the structure of the input form.

## 2. STRIP PROJECTION

### 2.1. Bounding rectangle detection

An input form image is usually surrounded by white margins, that do not contain any useful information. Since we will determine some variables, such as the number of strips, according to the actual image size, it is necessary to remove these meaningless margins before further processing. To obtain the exact size of the input form, we apply a bounding-rectangle detection procedure.

In this procedure, we scan the input image from all four sides to find the first scan line that has at least 10 black pixels, and take these lines as the boundary line. The bounding rectangle is composed using these boundary lines; Fig. 2 shows a detected bounding rectangle.

### 2.2. Strip projection

Figure 3 shows the horizontal and vertical projection profiles for Fig. 2. We can see that projection of the whole image can only indicate longer lines. We cannot determine shorter lines' locations from these projection profiles since these lines do not produce high enough peaks. Even if the peak produced by a given short line is high enough, we still cannot know where this line should be. Is the line centered, or not?

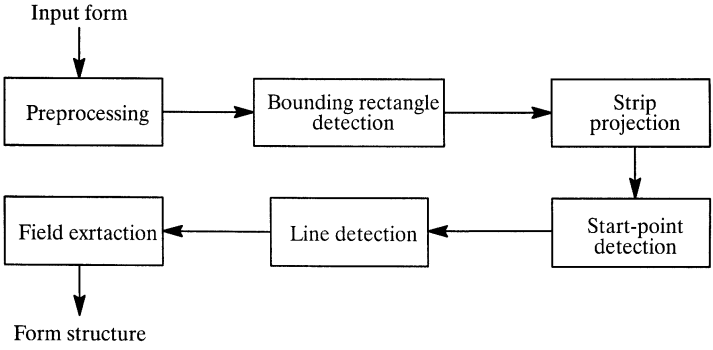



Fig. 1. The system diagram.

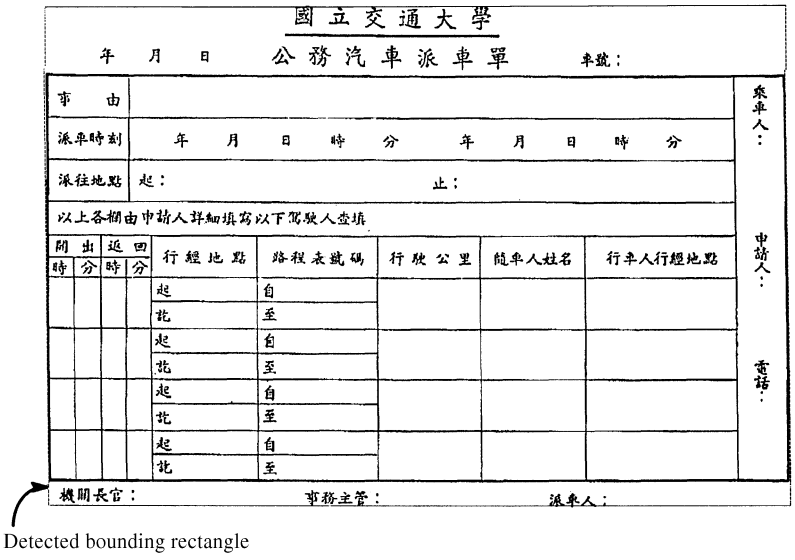



Fig. 2. Bounding rectangle detected from a form image.

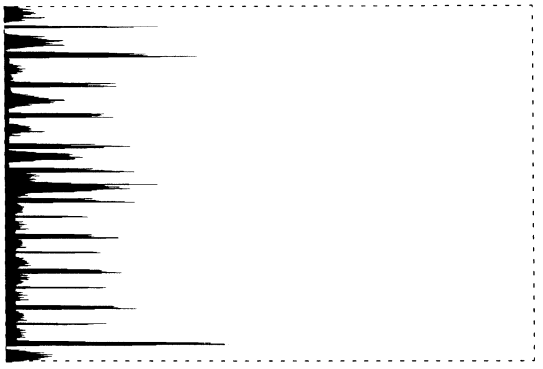



Fig. 3. (a) Horizontal projection of Fig. 2, and (b) vertical projection of Fig. 2.

Thus, we cannot obtain enough information to extract shorter lines by projecting the entire image. This problem becomes more serious when there is handwritten data and other information inside the input form document. To overcome this problem, we cut the input form document image into strips and perform projection of each strip. In Fig. 4, we see an input image cut into 10 vertical strips.

Since each strip is smaller in width, the peaks in the strip projection profile will not be affected by small skew angles. Figure 5a shows a horizontal strip that has three vertical lines on it, two long horizontal lines and six Chinese characters. In the vertical projection shown in Fig. 5b, the three vertical lines correspond to three peaks. Note that in the ideal case, the number of points in the projection corresponds to the line lengths. If the form document is slightly skewed, we can sum up the numbers of pixels around the points with peaks. Figure 5d shows a skewed projection profile of the strip shown in Fig. 5c.

When a form image is input, we first divide this input form into $D$ horizontal and $D$ vertical strips. Choosing a suitable value for $D$ can optimize the performance of our algorithm. However, our algorithm is not sensitive to slight changes in $D$. If $D$ is set too large or too small, extra lines may be extracted or lines may be missed. To achieve optimal performance, the strip number $D$ is chosen such that it satisfies the following criteria:

$$D \geq (\text{Document Width})/(\text{length of the shortest horizontal form line})$$

and

$$D \geq (\text{Document Height})/(\text{length of the shortest vertical form line}).$$

By applying these criteria, we make sure that we can locate at least one start point for each form line from the strip projection profiles in later sections. However, the computation time will increase for

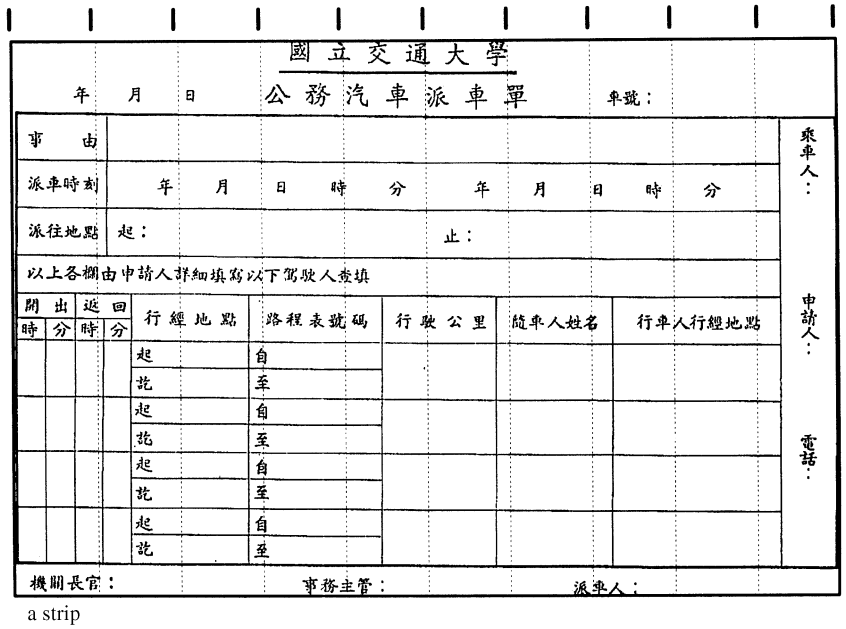



Fig. 4. A form image cut into ten vertical strips.

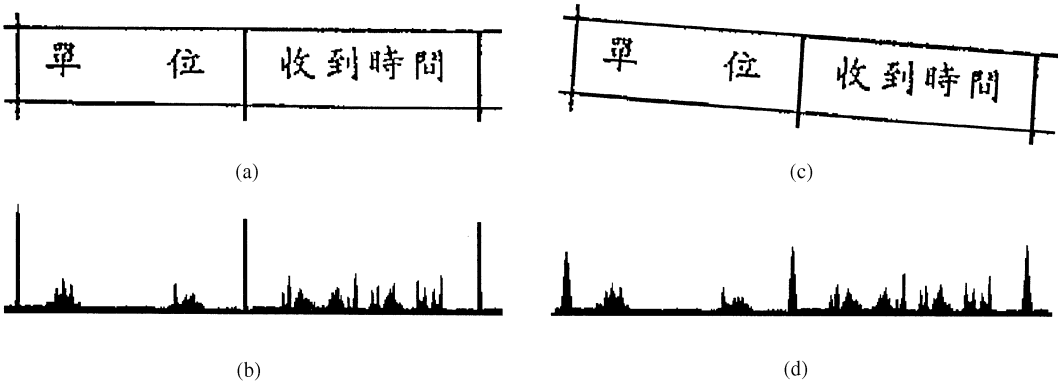



Fig. 5. (a) Part of a horizontal strip from a form image. (b) Vertical projection profile of (a). (c) part of a horizontal strip from a skewed form image. (d) vertical projection profile of (c).

larger $D$, since there will be more start points for line tracing. We may also extract lines from characters when $D$ is set too large. According to the form documents we have collected, we set an upper bound of 30 on strip number $D$ in our system. The actual strip number $D$ can be determined by users according to the documents they are processing. Otherwise, $D$ is set to a default value. In our experiments, a default value of 10 for $D$ is suitable for most office form documents.

In choosing $D$, we want the projection peak of the shortest form line to be higher than half the width of a strip, then this peak can be located for line extraction. If the value of $D$ is too small, shorter lines will not be extracted since their projection values will not be higher than half the strip width; if $D$ is too large, the projection results of characters and short lines will be confusing, since character projections will then be large enough to indicate the existence of lines.

In Fig. 6, we can see quite clearly that all lines in this form correspond to peaks in the strip projection profiles. Shorter line segments can be found by locating peaks in some strips. From the horizontal projection profiles of the vertical strips, we try to locate the peaks that correspond to the horizontal lines in the strips. Also, from the vertical projection profiles of the horizontal strips, we try to find vertical lines. By combining the results of both projections, we can find form structures that consist of horizontal lines and vertical lines. Other operations required are explained in the next section.

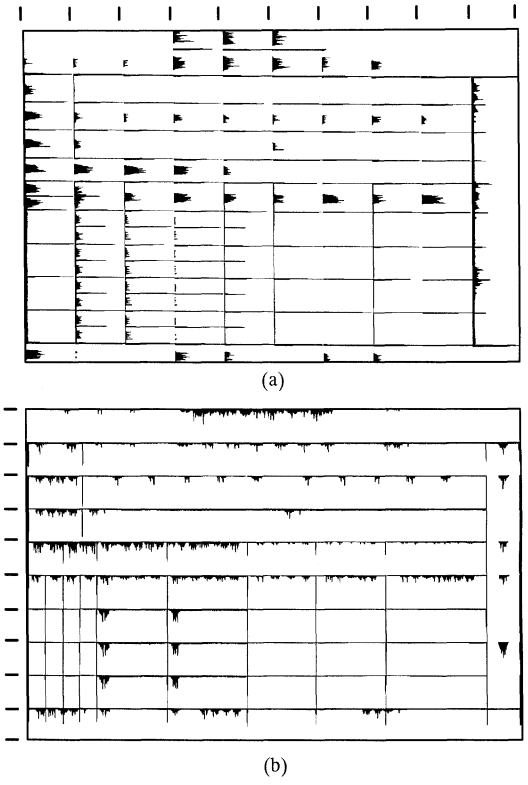



Fig. 6. (a) Horizontal projection of ten vertical strips, and (b) vertical projection of ten horizontal strips. Horizontal lines can be extracted from the projection profile shown in (a). Notice that the lines shown in (a) and (b) are not real lines; they are the projection profiles of form lines. Since the scales of the profiles and the image are the same, the heights of peaks in the horizontal profile are equal to the lengths of horizontal lines in that vertical strip.

Below, we define some variables used in this paper. These values can be calculated according to the image size and the number of strips, $D$.

- *Minimum field size* ($F_m$): The minimum field size is estimated from the strip number $D$ and is defined as:

$$F_m = \frac{\text{Min}(Width,\ Height)}{2D},$$

where *Width* and *Height* represent the input form size. This value is used for estimating the length of the shortest field-line in the input form image. For the example in Fig. 6, *Width* is equal to 1093, *Height* is equal to 717, and thus $F_m$ is equal to 35.

- *Character size* ($C$): This size represents the approximated width or height of a character in the form document image. We define this value as half of the minimum field size $F_m$.

$$C = \frac{F_m}{2}.$$

- *Minimum line length* ($l_m$): This value represents the minimum length of a valid extracted line, which is defined as the minimum field size:

$$l_m = F_m.$$

- *Line width* ($W_l$): $W_l$ represents the width of a form line. In our example, the minimum value for line width is set to 5, and the maximum value is set to 10. $W_l$ is estimated from the character size, $C$. If the value of $C/15$ is not within a pre-defined range of line widths, we set $W_l$ back to 5 or 10 according to the value of $C/15$:

$$W_l = \begin{cases} 5 & \text{if } C/15 < 5, \\ 10 & \text{if } C/15 > 10, \\ C/15 & \text{otherwise.} \end{cases}$$

$W_l$ is used to check the validity of start points. If the distance between a given start point and a previously extracted line is smaller than $W_l$, we skip this start point in the line-tracing process.

- *Projection peak threshold*: This threshold value is used to find the start points for the line-tracing process that follows. If the peak in the projection profile is greater than this threshold, there is probably a line at the peak position. We then set the peak position as a possible line position. Both horizontal and vertical projection-peak threshold values are related to the number of strips, $D$.

$$Peak_h = \frac{1}{2}\frac{Width}{D} \quad \text{for horizontal strips;}$$

$$Peak_v = \frac{1}{2}\frac{Height}{D} \quad \text{for vertical strips.}$$

From the strip projection profile shown in Fig. 6, we locate the peak positions greater than $Peak_h$ or $Peak_v$, which may correspond to form lines. These located positions are recorded as $\{Ph_i\}$ and $\{Pv_i\}$ for vertical and horizontal projection strips, respectively.

## 3. LINE DETECTION

After we locate the peak positions, we can detect horizontal and vertical line segments using a line-tracing algorithm. These lines are traced from the start points obtained from the peak positions in the form image. After all lines have been traced, invalid lines are removed by a line-verification algorithm and broken lines are re-linked by a line-merging algorithm.

### 3.1. Start-point detection

The peak positions obtained from projection profiles represent the possible existence of lines in the input form. Some false peak positions must be skipped in the line-tracing and line-verification procedures. Validating these false peak positions does not take much computation time because most false positions exist in characters and characters consist of short lines. We detect a set of horizontal start points,

$\{Sh_i\}$, for each horizontal peak position, $Ph_i$, and a set of vertical start points, $\{Sv_i\}$, for each vertical peak position, $Pv_i$, to be used in the line-tracing process described in the next subsection. Since the peak thresholds we use to validate possible line positions are set at half a strip width, the best position to start line tracing is right at the mid-point of a strip. Thus, the start points, $\{Sh_i\}$ and $\{Sv_i\}$, can be defined as follows.

Horizontal start points in vertical strips:

$$\begin{cases} Sh_i.x = Ph_i.x + \dfrac{W_s}{2} - k, & \text{where } k = \min_j\left(I\left(Ph_i.x + \dfrac{W_s}{2} - j, Ph_i.y\right) = \text{Black}\right), \\ Sh_i.y = Ph_i.y, & 0 \le j \le \dfrac{Width}{D}. \end{cases}$$

Vertical start points in horizontal strips:

$$\begin{cases} Sv_i.x = Pv_i.x, & \text{where } k = \min_j\left(I\left(Pv_i.x, Pv_i.y + \dfrac{H_s}{2} - j\right) = \text{Black}\right), \\ Sv_i.y = Pv_i.y + \dfrac{H_s}{2} - k, & 0 \le j \le \dfrac{Height}{D}. \end{cases}$$

Note that $W_s$ represents the width of a vertical strip $H_s$, represents the height of a horizontal strip, and $I(x, y)$ represents the image labeling function. We show part of a vertical strip and its corresponding horizontal projection profile in Fig. 7 to illustrate the above equations. The bullets shown in Fig. 8 indicate the detected start points, $\{Sh_i\}$ and $\{Sv_i\}$, for Fig. 2.

### 3.2. Feasible Skew Angle

Given the projection method and strip widths we use, our method can tolerate some level of skewness. We locate start points by applying the peak threshold values from horizontal and vertical projection profiles. Thus, if the input image projection profiles produce peaks higher than the peak thresholds, we can locate the correct start points. The feasible skew angle is illustrated in Fig. 9.

The value of the maximum feasible skew angle can be determined from the line width and peak threshold, as shown below:

$$\text{Max. Feasible Skew Angle} = \tan^{-1}\frac{W_l}{\text{Max}(Peak_h, Peak_v)}.$$

### 3.3. Line tracing

After we find the sets of start points, $\{Sh_i\}$ and $\{Sv_i\}$, we can use them to trace and extract all form lines. In the line-tracing process, we use a $3 \times 3$ window for connectivity checking. The labels for the pixels in the $3 \times 3$ window are shown in Fig. 10. Since we find horizontal lines from the start points on vertical strips and find vertical lines from the start points on horizontal strips, we start tracing horizontal and vertical lines from $\{Sh_i\}$ and $\{Sv_i\}$, respectively.

*3.3.1. Horizontal line tracing.* When tracing horizontal lines, we take $\{Sh_i\}$ as start points, and trace left and right to find the end-points of each horizontal line. For a particular start point in $\{Sh_i\}$, the steps used to detect the left end-point are as follows:

*Step* 1: Set $P$ as a start point in $\{Sh_i\}$.
*Step* 2: If $P_5$ is a black pixel, move $P$ to $P_5$. Repeat step 2.
*Step* 3: If $P_6$ is a black pixel, move $P$ to $P_6$, and go to step 2.
*Step* 4: If $P_6$ is a black pixel, move $P$ to $P_4$, and go to step 2.
*Step* 5: If there is a black pixel $P_i$ on the left side of $P$ and also in the same row as $P$, and *Distance* $(P_i, P) < C/5$. Then, move $P$ to $P_i$ and go to step 2.
*Step* 6: Record $P$ as the left end-point of the current trace line.

The right end-point can be found by replacing $P_5$, $P_4$, $P_6$ with $P_1$, $P_2$ and $P_8$, respectively, and using a similar algorithm.

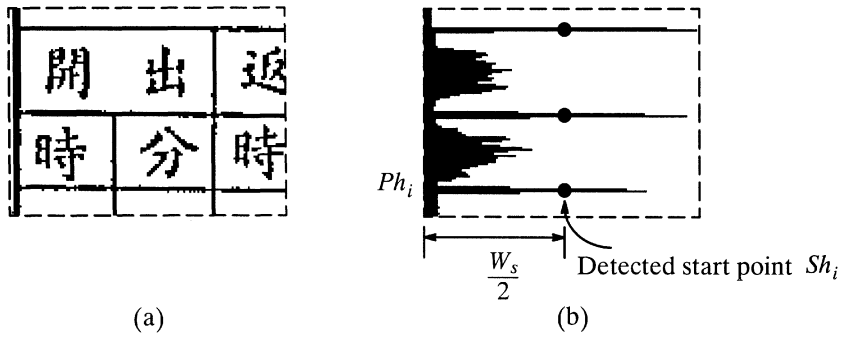
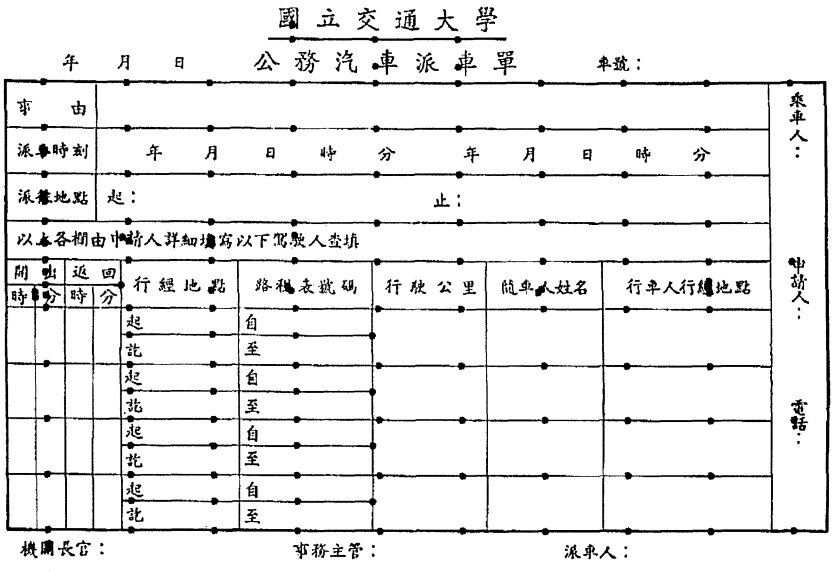



Fig. 7. (a) Part of a vertical strip of Fig. 2; (b) horizontal projection profile and detected start points of (a).


國立交通大學

年 月 日 公務汽車派車單 車號：

事由 乘車人：

派車時刻 年 月 日 時 分 年 月 日 時 分

派往地點 起： 止：

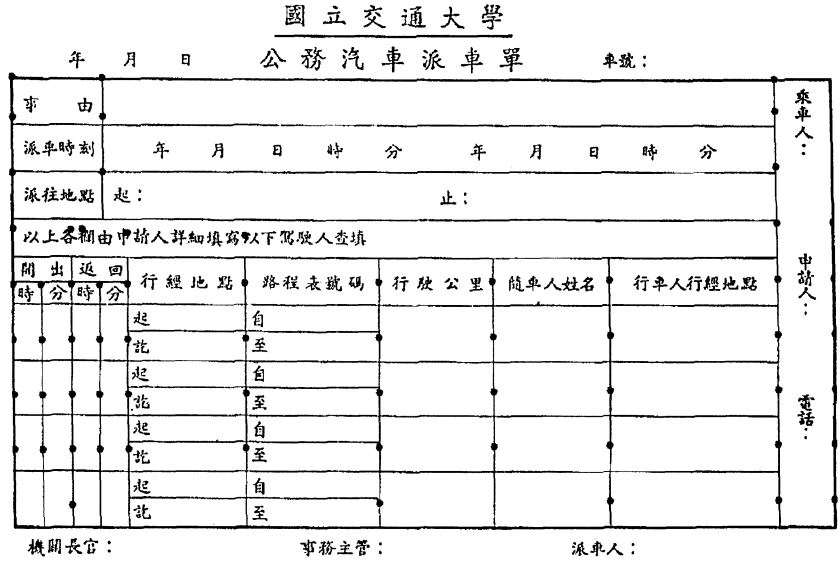
以上各欄由申請人詳細填寫以下駕駛人查填

| 開出 時 | 開出 分 | 返回 時 | 返回 分 | 行經地點 | 路程表號碼 | 行駛公里 | 隨車人姓名 | 行車人行經地點 |
|---|---|---|---|---|---|---|---|---|
| | | | | 起 | 自 | | | |
| | | | | 訖 | 至 | | | |
| | | | | 起 | 自 | | | |
| | | | | 訖 | 至 | | | |
| | | | | 起 | 自 | | | |
| | | | | 訖 | 至 | | | |
| | | | | 起 | 自 | | | |
| | | | | 訖 | 至 | | | |

申請人： 電話：

機關長官： 事務主管： 派車人：

(a)


國立交通大學

年 月 日 公務汽車派車單 車號：

事由 乘車人：

派車時刻 年 月 日 時 分 年 月 日 時 分

派往地點 起： 止：

以上各欄由申請人詳細填寫以下駕駛人查填

| 開出 時 | 開出 分 | 返回 時 | 返回 分 | 行經地點 | 路程表號碼 | 行駛公里 | 隨車人姓名 | 行車人行經地點 |
|---|---|---|---|---|---|---|---|---|
| | | | | 起 | 自 | | | |
| | | | | 訖 | 至 | | | |
| | | | | 起 | 自 | | | |
| | | | | 訖 | 至 | | | |
| | | | | 起 | 自 | | | |
| | | | | 訖 | 至 | | | |
| | | | | 起 | 自 | | | |
| | | | | 訖 | 至 | | | |

申請人： 電話：

機關長官： 事務主管： 派車人：

(b)

Fig. 8. (a) Horizontal start points from vertical strips; (b) vertical start points from horizontal strips.

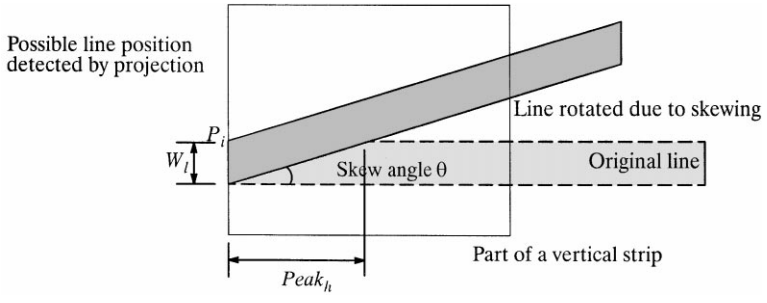



Fig. 9. Feasible skew angle.

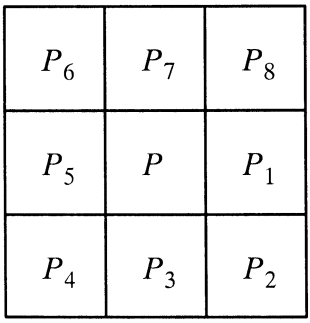
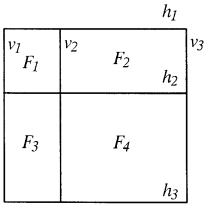



Fig. 10. Three × three windows used in line-tracing.

After we obtain the two end-points, a line segment is extracted. The extracted line is located on the upper edge of the actual line in the input form image. We then check this extracted line using another verification process described below. This line-tracing method can trace not only horizontal and vertical lines, but can also trace lines skewed less than 45°. Thus, we can find lines using this line-tracing method from given start points even if the input form image is skewed.

*3.3.2 Small-gap skipping.* While performing step 5 of the horizontal line-tracing algorithm, we detect and fill small gaps in lines that may have been caused by bad scanning quality or bad printing quality. The threshold value for this gap size should be relatively small, otherwise, some unwanted lines may be extracted when very short line segments in characters are connected, a situation that often occurs in Chinese documents. We set this gap threshold to $C/5$ in our experiments. Thus some broken lines may not be connected in this step; we can connect these broken lines with line merging later on.

*3.3.3. Vertical line tracing.* The vertical line-tracing algorithm is quite similar to that for horizontal line tracing, except that the tracing directions are changed to up and down. The points $P_5$, $P_4$, $P_6$ used in left end-point detection are replaced by $P_7$, $P_6$, $P_8$, respectively, in tracing toward the upper end-point, and replaced by $P_3$, $P_4$, $P_2$ in tracing toward the lower end-point. The extracted vertical line is



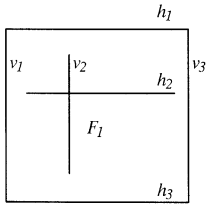

(a)

Horizontal lines: $h_1, h_2, h_3$.
Vertical lines: $v_1, v_2, v_3$.
Field extraction:

| | |
|---|---|
| 1. $(h_1, v_1, h_2, v_2)$ | Obtain a field $F_1$. Try next left-line. |
| 2. $(h_1, v_2, h_2, v_3)$ | Obtain a field $F_2$. Try next left-line. |
| 3. $(h_1, v_3, h_2, -)$ | This is not a field. |
| 4. $(h_1, v_3, h_3, -)$ | This is not a field. |
| 5. $(h_2, v_1, h_3, v_2)$ | Obtain a field $F_3$. Try next left-line. |
| 6. $(h_2, v_2, h_3, v_3)$ | Obtain a field $F_4$. Try next left-line. |
| 7. $(h_2, v_3, h_3, -)$ | This is not a field. |
| 8. $(h_3, v_1, -, -)$ | This is not a field. |
| 9. $(h_3, v_2, -, -)$ | This is not a field. |
| 10. $(h_3, v_3, -, -)$ | This is not a field. |

Extracted fields: $F_1, F_2, F_3, F_4$.

(b)




(c)

Horizontal lines: $h_1, h_2, h_3$.
Vertical lines: $v_1, v_2, v_3$.
Field extraction:

| | |
|---|---|
| 1. $(h_1, v_1, h_2, -)$ | This is not a field. |
| 2. $(h_1, v_1, h_3, v_2)$ | This is not a field. |
| 3. $(h_1, v_1, h_3, v_3)$ | Obtain a field $F_1$. Try next left-line. |
| 4. $(h_1, v_2, -, -)$ | This is not a field. |
| 5. $(h_1, v_3, h_2, -)$ | This is not a field. |
| 6. $(h_1, v_3, h_3, -)$ | This is not a field. |
| 7. $(h_2, v_1, -, -)$ | This is not a field. |
| 8. $(h_2, v_2, h_3, -)$ | This is not a field. |
| 9. $(h_2, v_3, -, -)$ | This is not a field. |
| 10. $(h_3, v_1, -, -)$ | This is not a field. |
| 11. $(h_3, v_2, -, -)$ | This is not a field. |
| 12. $(h_3, v_3, -, -)$ | This is not a field. |

Extracted fields: $F_1$.

(d)

Fig. 11. (a) A form with four fields and six lines. (b) The trace of field-extraction of form (a). (c) A form with one field and six lines. (d) The trace of field extraction of form (c).

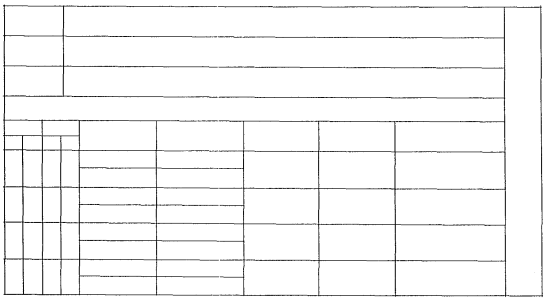


Fig. 12. Form structure extracted from Fig. 2.

represented by both upper and lower end points and will also be verified later. This extracted line is located at the left edge of the actual line in the input form image.

### 3.4. Line verification and merging

*3.4.1. Line verification.* Figure 8 shows some useless or redundant start points. Some of them are on Chinese characters, and some of them are on the same lines as other start points. Some of these points are caused by sequences of characters within fields. These character sequences make the projection values at those positions high enough to be treated like possible line positions. Since these points are not correct start points, we have to remove them.

Before tracing a line, we check whether the start point is on any line already extracted. For a particular start point, $S$, and each extracted line, $L$, we check whether the distance from $S$ to $L$ is less than the threshold, $W_l$. If we find such a line $L$, then the start point $S$ is taken as a redundant start point and eliminated.

Next, we check the lengths of lines extracted by the line-tracing process and remove lines shorter than the threshold $l_m$; such lines often exist in Chinese characters.

*3.4.2. Line merging.* Figure 13 shows some small broken form lines in the input form image resulting from poor scanning or printing quality. Since lines with large gaps cannot be connected by the method mentioned in Section 3.2.2, we connect extracted lines that are co-linear when the distance between them is smaller than the minimum field size, $F_m$. The acceptable gap in this step is much greater than the gap mentioned in Section 3.2.2. Since these lines have already been extracted, we do not have to consider lines embedded in Chinese characters here.

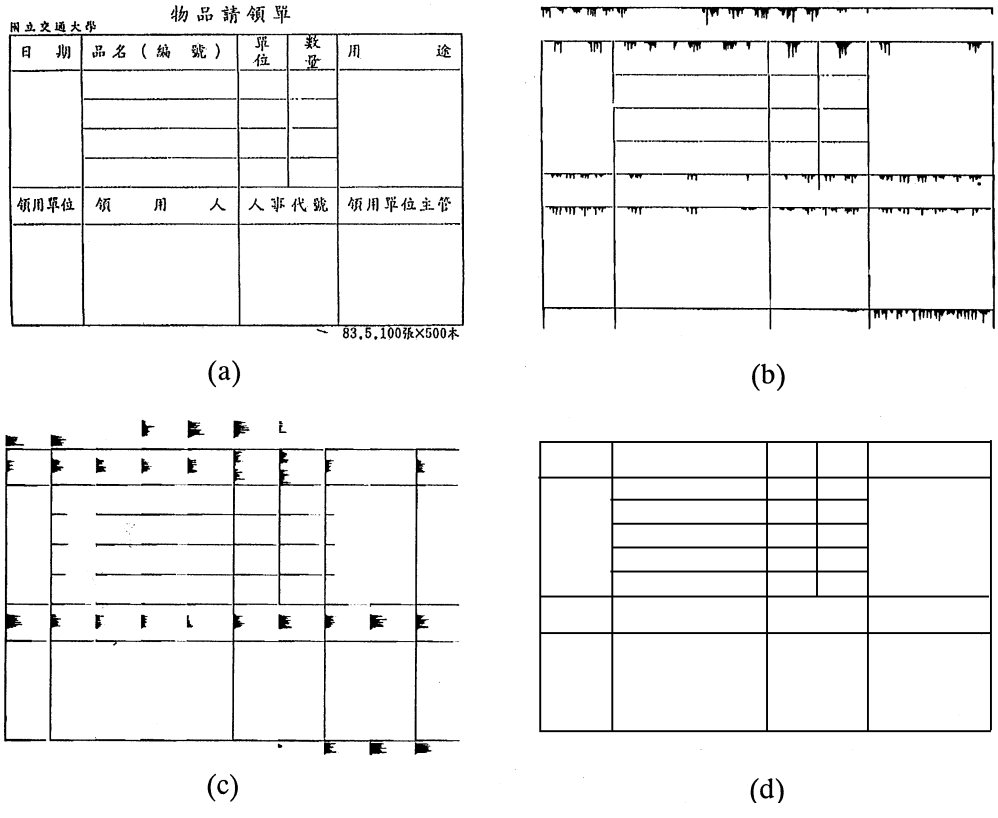



Fig. 13. Another results: (a) Input form image; (b) vertical projection on horizontal strips; (c) horizontal projection on vertical strips; (d) form structure extracted.

### 4. FIELD EXTRACTION

Even after we perform line verification, some non-form lines may survive due to touching characters, especially handwritten Chinese characters; some underlines may also survive. To eliminate these non-form lines and interpret the actual form structure, we perform field extraction. In extracting fields, we determine where fields are, so we can use that information to extract data and delete lines not related to any fields.

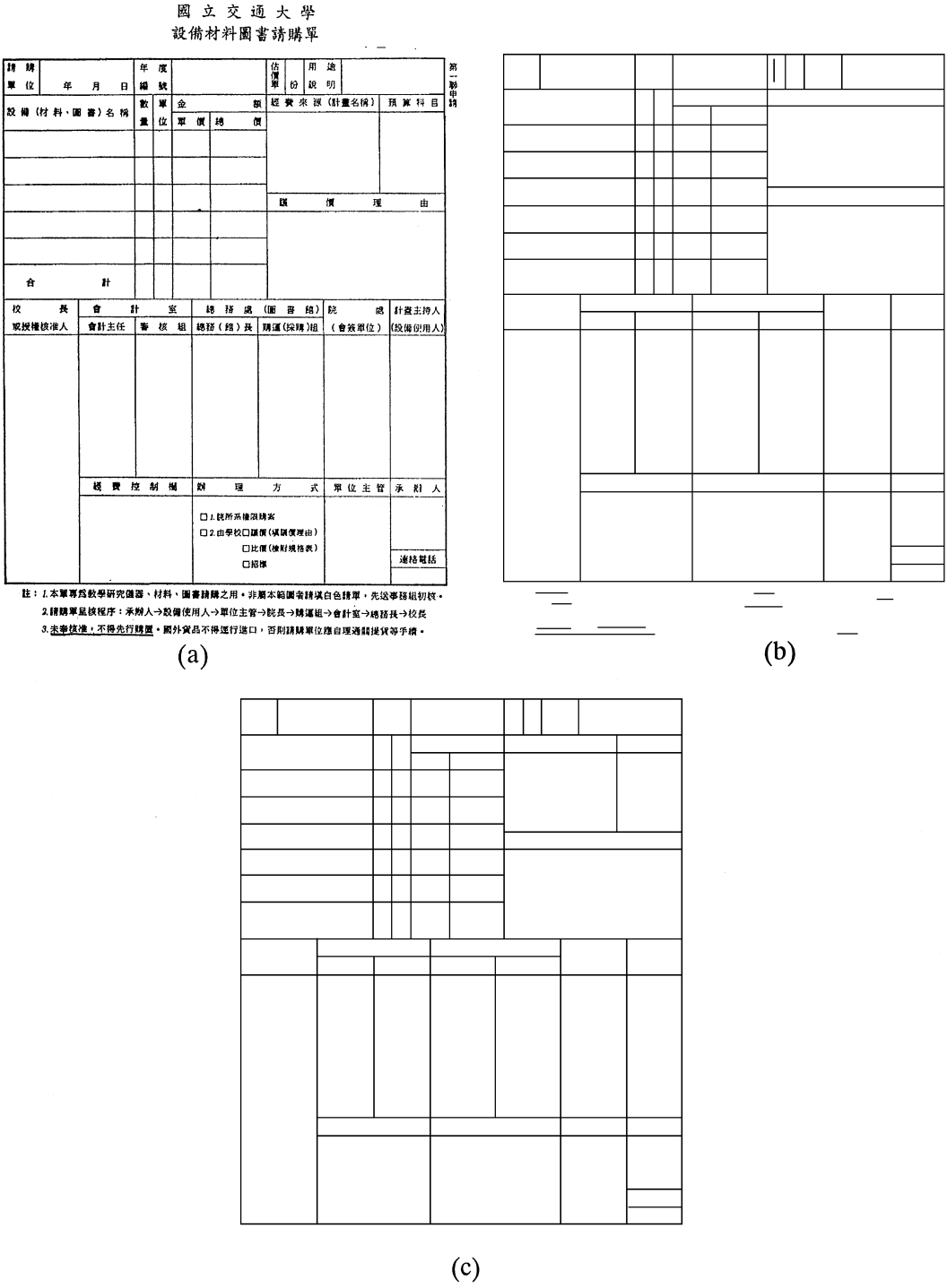



Fig. 14. Another result: (a) Input binarized form image; (b) result after line extraction; (c) final result.

The line-extraction procedure examines the start points from left to right, top to bottom. Thus, all lines are extracted in the order of their positions without additional sorting. After extracting all lines, we take every two horizontal lines as a horizontal pair and every two vertical lines as a vertical pair in the order of their positions. We then go through all horizontal and vertical pairs to check whether these four-line groups form fields. If so, we mark these four-line groups to indicate that these lines are meaningful to the form structure, then continue finding fields until there are no more unprocessed lines. Figure 11 shows two forms with different form structures, and the traces of field extraction. According to our method, four fields in Fig. 11(a) and one field in Fig. 11(c) can be extracted correctly. Note that a field is formed by a 4-tuple, (*Top-line*, *Left-line*, *Bottom-line*, *Right-line*). The field-construction procedure is shown

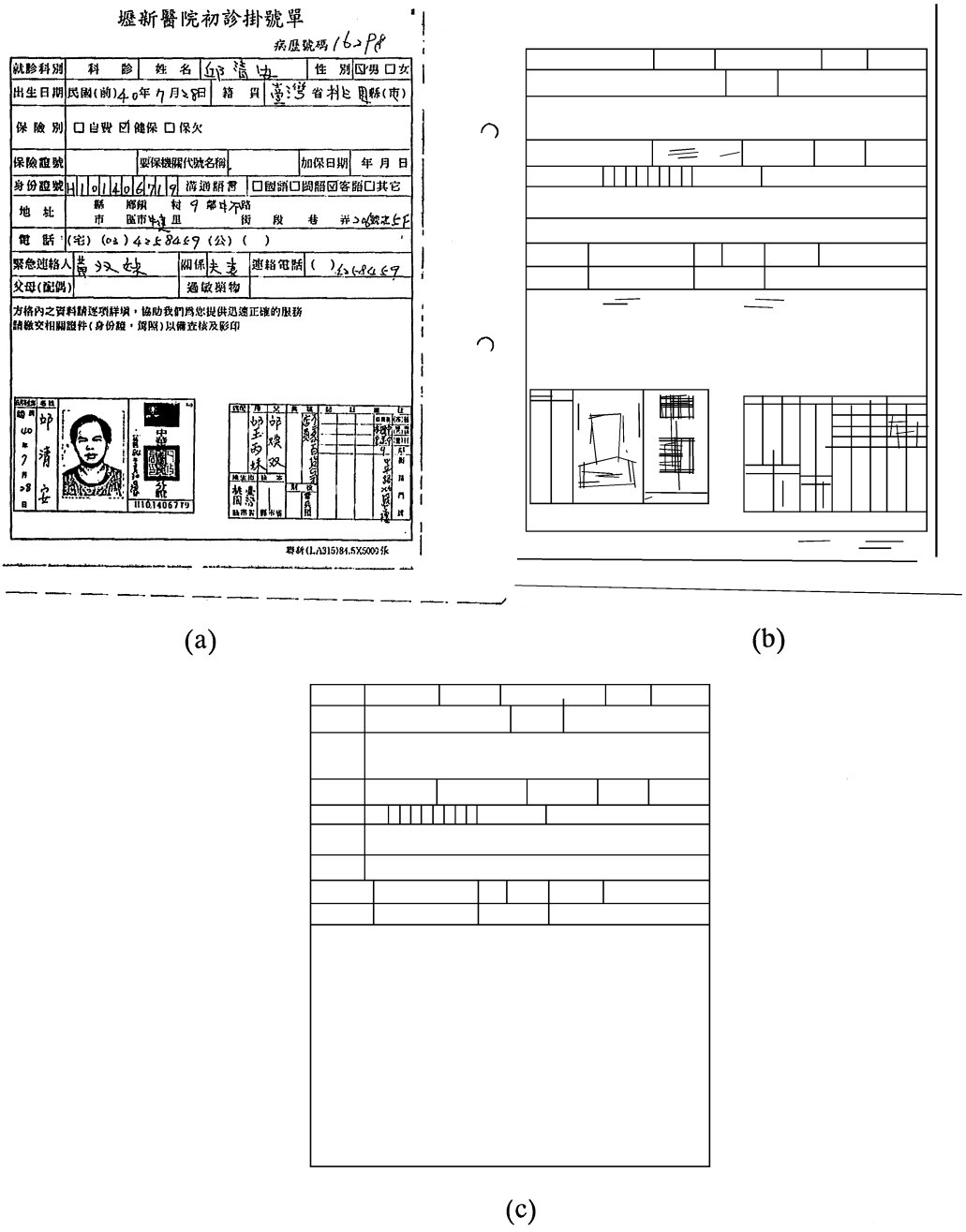
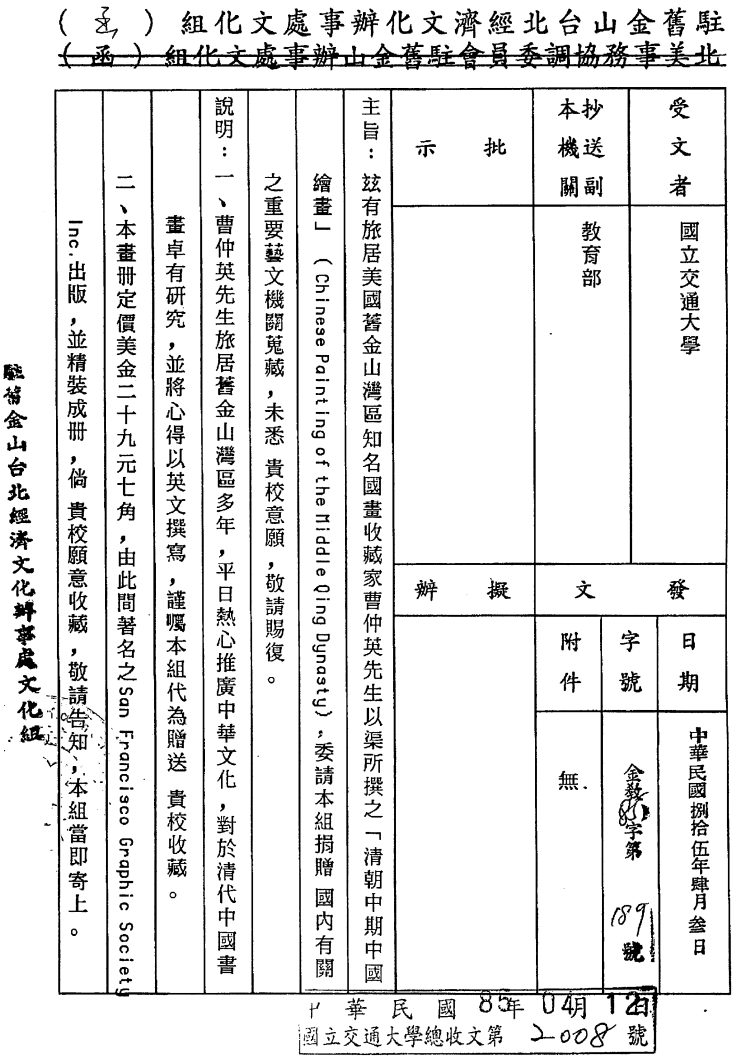



Fig. 15. An example with data fields: (a) Original binarized image; (b) results after line extraction; (c) final result.

below:

for *hline1* in all horizontal-lines
    if *hline1* is in some extracted field, try next line.
    for *vline1* in all-vertical-lines
        if *vline1* is in some extracted field, try next line.
        if *hline1* and *vline1* do not intersect, try next line.
        for *hline2* in remaining-horizontal-lines
            if *hline2* is in some extracted field, try next line.
            if *vline1* and *hline2* do not intersect, try next line.
            for *vline2* in remaining-vertical-lines
                clear *GotAField*.
                if *vline2* is in some extracted field, try next line.
                if *hline2* and *vline2* do not intersect, try next line.
                *hline1*, *vline1*, *hline2* and *vline2* form a filed, record this field.
                mark these four lines as meaningful.
                break for next *hline2*, and set *GotAField*.
    if *GotAField*, then break for next *vline1*.




(a)

Fig. 16. An example with a stamp: (a) Original binarized image; (b) extracted lines; (c) final result.

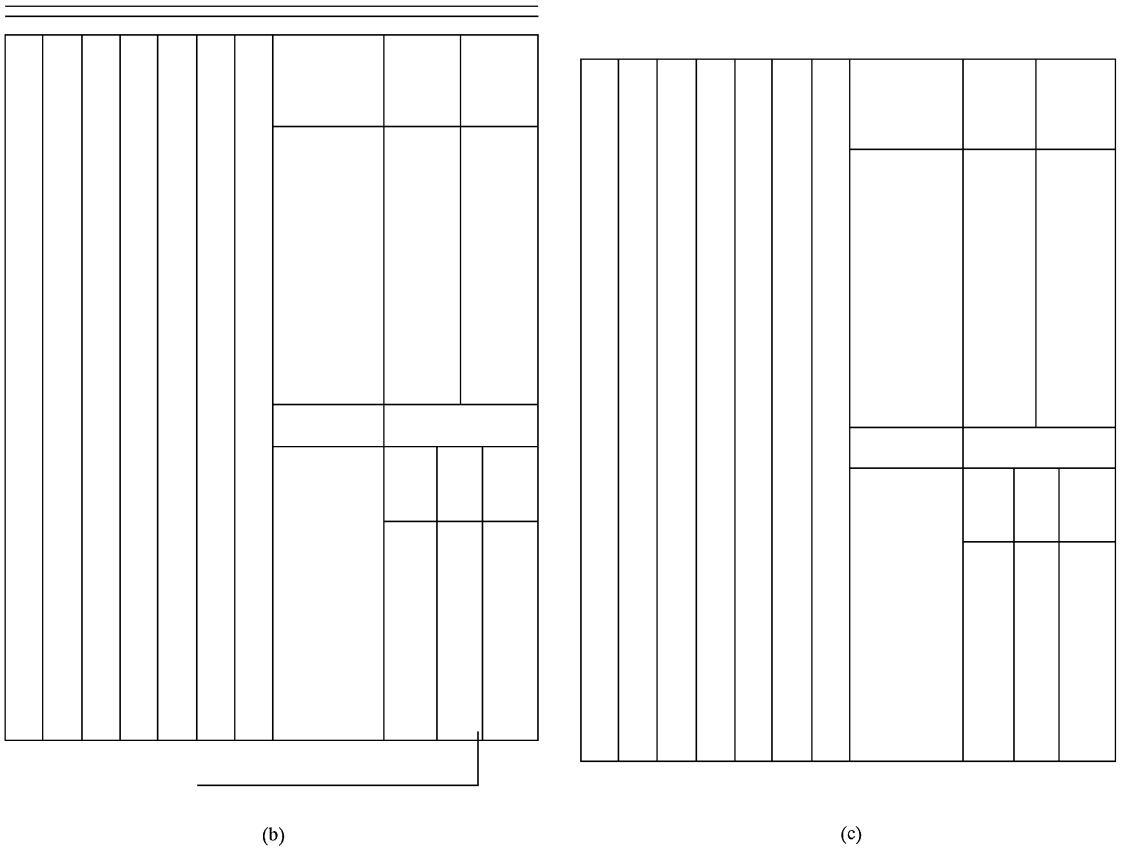



Fig. 16. *continued*

After extracting all fields, we can determine which lines are meaningful in the form structure and then remove those lines not belonging to any fields. The remaining form lines and extracted field structures represent the final form structure information we want.

Our field-extraction algorithm discards lines enclosed in some extracted fields, because we do not accept form documents containing tables within tables. In our experiments, most lines extracted from inside fields are caused by characters, graphs or images, and thus are not parts of the form structures we want. Therefore, we eliminate these lines in the line-verification and field-extraction processes.

The short lines in the lower half of Fig. 15 (b) are caused by the image of an ID card copy stuck on the input form document. They are not real form lines belonging to that document, and must be removed in the field-extraction process, as shown in Fig. 15 (c).

## 5. EXPERIMENTAL RESULTS

In this section, some experimental results are presented to verify the validity of our proposed method. Our system was implemented in C language on a SparcStation 10. The input forms we used are typical office forms and the scanning resolution was 200dpi (dots per inch). The form structure extracted from Fig. 2 is shown in Fig. 12. Figure 13 shows another input form with vertical projections, horizontal projections and the extracted form structure.

The proposed strip projection method works well in extracting form structures from form documents. The computation time without code optimization for a $1614 \times 1043$ form image, as shown in Fig. 12, is about 1.1 s.

The maximum feasible skew angle for Fig. 12 is about 4.2°. We do not perform any skew correction in our system. The image shown in Fig. 15(a) is slightly skewed, which does not influence the results of line extraction. Accordingly, our method can tolerate a certain degree of skewness.

Figures 14 and 15 show form documents with more complex structures. Filled-in data and pictures are included in Fig. 15 and some fields in these two forms are quite small. According to the strip number consideration described above, we know that 10 strips in both directions would not be suitable for these two forms. So, we used 13 horizontal and vertical strips for the image in Figs. 14 and 15 for the image in Fig. 15.

We can see that there is a short line absent from the extracted structure shown in Fig. 13(c). This is because the acceptable minimum line length, $l_m$, was longer than the length of this line by two pixels. Such problems can be solved by increasing the strip number, $D$.

Figure 16 shows an example containing a stamp. As shown in Fig. 16(b) and (c), some stamp lines were extracted during line-tracing, then removed during field extraction. Figure 17 shows a noisy input image. Such noise may be caused by poor binarization, but our proposed method is not affected by this kind of noise. We can still extract correct form structures, as shown in Fig. 17(c).

Execution times and extraction percentages are shown in Tables 1 and 2. As Table 1 shows, the execution time depends on the image resolution. The complexity of the input image does not influence execution speed much.

To demonstrate the efficiency of our algorithm to extract form lines, we compared our approach with two other algorithms, the run-based method and the Hough transform method. We recorded the execution times of the three algorithms, as run on a Pentium-90

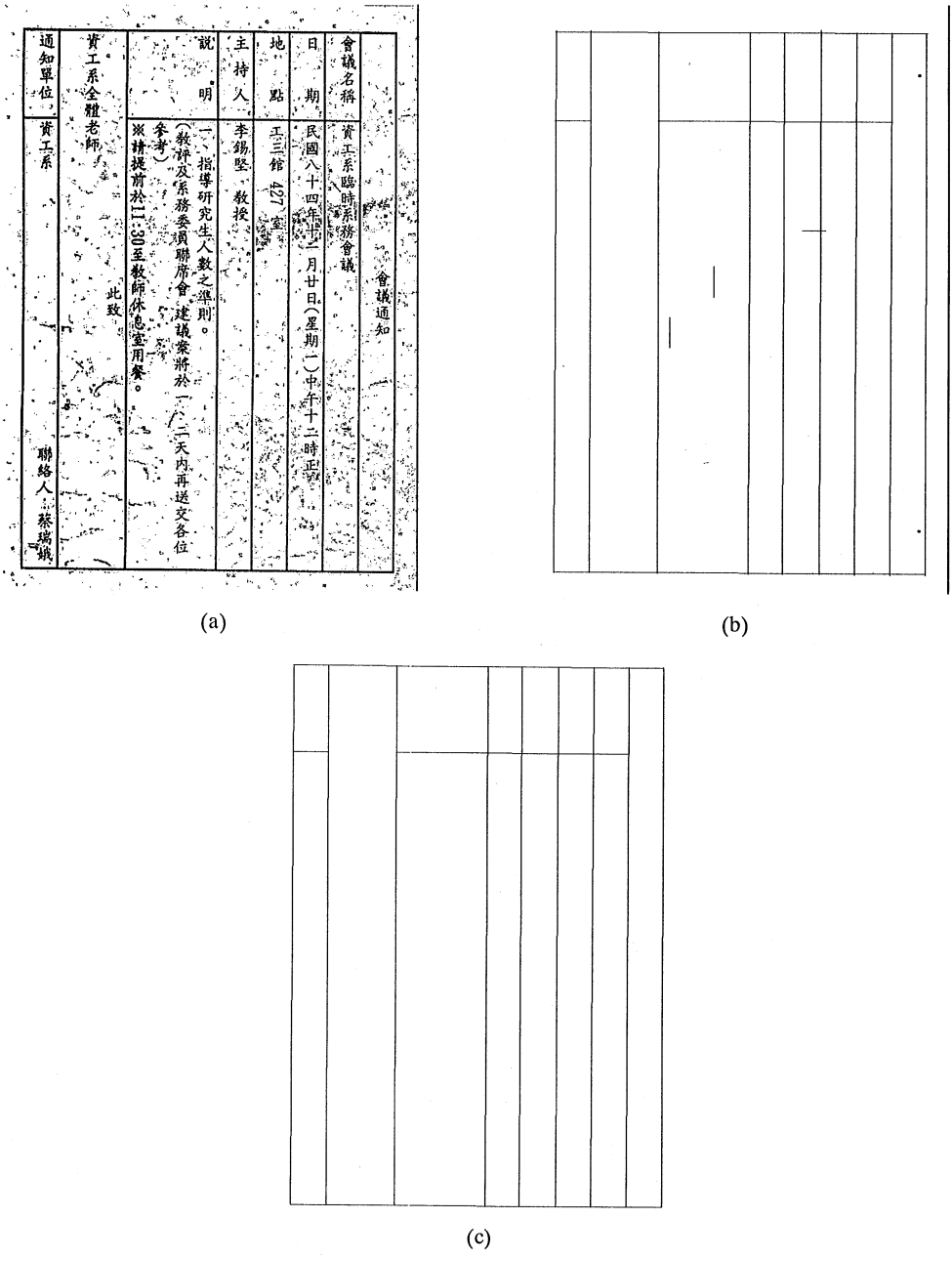



Fig. 17. A noisy example: (a) Original input form image; (b) extracted lines; (c) final result.

Table 1. Execution time

| | Total time (s) | Disk I/O time (s) | Extraction time (s) |
|---|---|---|---|
| Fig. 12; 200 dpi, 1614 × 1043 pixels | 0.87 | 0.35 | 0.52 |
| Fig. 13; 200 dpi, 1064 × 725 pixels | 0.47 | 0.20 | 0.27 |
| Fig. 14; 200 dpi, 1600 × 2123 pixels | 2.16 | 0.87 | 1.29 |
| Fig. 15; 200 dpi, 1255 × 1650 pixels | 1.37 | 0.43 | 0.94 |
| Fig. 16; 300 dpi, 2400 × 2937 pixels | 5.42 | 1.64 | 3.78 |
| Fig. 17; 200 dpi, 828 × 1165 pixels | 0.69 | 0.29 | 0.40 |

Table 2. Form-line extraction percentages

| | Total lines | Extraction lines | Extraction percentage (%) |
|---|---|---|---|
| Fig. 12; 200 dpi, 1614 × 1043 pixels | 15 horizontal<br>12 vertical | 15 horizontal<br>12 vertical | 100 |
| Fig. 13; 200 dpi, 1064 × 725 pixels | 8 horizontal<br>6 vertical | 8 horizontal<br>6 vertical | 100 |
| Fig. 14; 200 dpi, 1600 × 2123 pixels | 18 horizontal<br>18 vertical | 18 horizontal<br>18 vertical | 100 |
| Fig. 15; 200 dpi, 1255 × 1650 pixels | 11 horizontal<br>29 vertical | 11 horizontal<br>28 vertical | 97.5 |
| Fig. 16; 300 dpi, 2400 × 2937 pixels | 6 horizontal<br>13 vertical | 6 horizontal<br>13 vertical | 100 |
| Fig. 17; 200 dpi, 828 × 1165 pixels | 4 horizontal<br>9 vertical | 4 horizontal<br>9 vertical | 100 |

Table 3. Execution times for three form-line extraction methods

| | Strip projection (s) | Run-based method (s) | Hough transform (s) |
|---|---|---|---|
| Fig. 11; 200 dpi, 1614 × 1043 pixels | 0.88 | 2.53 | 203.76 |
| Fig. 12; 200 dpi, 1064 × 725 pixels | 0.44 | 1.43 | 93.37 |
| Fig. 13; 200 dpi, 1600 × 2123 pixels | 1.71 | 6.37 | 250.64 |
| Fig. 14; 200 dpi, 1255 × 1650 pixels | 0.42 | 3.35 | 411.15 |
| Fig. 15; 300 dpi, 2400 × 2937 pixels | 3.07 | 12.41 | 853.19 |
| Fig. 16; 200 dpi, 828 × 1165 pixels | 0.88 | 2.09 | 116.76 |

personal computer with 40 MB of RAM. Table 3 shows that our algorithm has the best performance among the three.

## 7. CONCLUSIONS

In this paper, we have proposed a strip-projection method for extracting form structures from form documents. This method is based on the simplest projection method, so implementation is quite easy. From the comparison results shown in Table 3, we know that our approach is more efficient than other methods, since we use projection to locate lines. With cutting input form images into strips, our method avoids losing short lines between fields, which may happen when using conventional projection methods. Our approach can detect lines touching characters or

other data that cannot be found using connected-component-based methods and run-based methods. The strip approach can also tolerate larger skew angles than other projection methods can. And, since the start-points we detect for line-tracing represent lines in certain forms, they can also be treated as features for form documents recognition. This is left for future research.

In the future, we will try to optimize the source codes of this method to further improve its performance. Character extraction and recognition will be added to the system based on the results in this paper. A complete medical information processing system, involving many form documents will be built using this approach.

## REFERENCES


1. R. G. Casey, D. R. Ferguson, K. Mohiuddin and E. Walach, Intelligent forms processing system, *Mach. Vision Applic*., **5,** 143–155 (1992).
2. T. Watanabe, Q. Luo and N. Sugie, Layout recognition of multi-kinds of table-form documents, *IEEE Trans. Pattern Anal. Mach. Intell*., **17**(4), 432–445 (1995).
3. S. L. Taylor, R. Fritzson and J.A. Pastor, Extraction of data from preprinted forms, *Mach. Vision Applic*. **5,** 211–222 (1992).
4. H. Fujisawa, Y. Nakano and K. Kurino, Segmentation method for character recognition: from segmentation to document structure analysis, *Proc. IEEE* **80**(7), 1079–1091 (1992).
5. Y. Y. Tang, C. De Yan and C.Y. Suen, Document processing for automatic knowledge acquisition, *IEEE Trans. Knowledge Data Engineering* **6**(1), 3–21 (1994).
6. K. C. Fan, J.M. Lu, L.S. Wang and H.Y. Liao, Extraction of characters from form documents by feature point clustering, *Pattern Recognition Lett*. **16,** 963–970 (1995).
7. S. W. Lam, L. Javanbakht, and S.N. Srihari, Anatomy of a form reader, *Proc. 2nd Int. Conf. Document Anal. Recognition*, 506–509 (1993).
8. X. N. Chen and D.C. Tseng, Form-structure extraction for table-form recognition, *Proc. 8th IPPR Conf. Computer Vision, Graphics and Image Processing*, pp. 496–503, Taiwan (1995).
9. C. T. Ho and L.H. Chen, A High-speed algorithm for line detection, *Pattern Recognition Lett*. **17,** 467–473 (1996).
10. W. Niblack, *An Introduction to Digital Image Processing*, pp. 115–116, Prentice Hall, Englewood Cliffs, Nw Jersey, (1986).
11. Ø. D. Trier and T. Taxt, Evaluation of binarization methods for document images, *IEEE Trans. Pattern Analy. Mach. Intell*. **17**(3), 312–315 (1995).



**About the Author**—JIUN-LIN CHEN received his B.S. degree from National Chiao-Tung University in computer science and information engineering in 1992. He is currently a Ph.D. studient in computer science and information engineering at National Chiao-Tung Univeristy. His research interest is document processing and image processing.

**About the Authors**—HSI-JIAN LEE received the B.S., M.S., and Ph.D. degrees in computer engineering from National Chiao Tung University, Hsinchu, Taiwan, in 1976, 1980, and 1984, respectively. From 1981 to 1984, he was a lecturer in the Department of Computer Engineering, National Chiao Tung University, and from 1984 to 1989 an associate profesor in the same department. Since August 1989, he has been with National Chiao Tung University as a professor. He is now the president of the Oriental Language Computer Society (OLCS), the editor-in-chief of the International Journal of Computer Processing of Oriental Language (CPOL), and an associate editor of the International Journal of Pattern Recognition and Artificial Intelligence, and Pattern Analysis and Applications. He has been a member of the executive committee of the Chinese Society on Image Processing and Pattern Recognition. He was responsible for the 1992 R.O.C. Computational Linguistic Workshop and 1993 R.O.C. Conference on Computer Vision, Graphics, and Image Procesing. He was the program chair person of the 1994 International Computer Symposium and the Fourth International Workshop on Frontiers in Handwriting Recognition (IWFHR). In 1992–94, he was a winner of outstanding researchers of the National Science Council, R.O.C. His current research interests include document analysis, optical character recognition, image processing, pattern recognition, and artificial intelligence. He is a member of Phi Tau Phi.