

A Form Dropout System

Bin Yu and Anil K. Jain

Dept. of Computer Science, Michigan State University

A714 Wells Hall, East Lansing, MI 48824-1027

binyu@cps.msu.edu jain@cps.msu.edu

Abstract

This paper describes a system for form dropout when the filled-in characters or symbols are either touching or crossing the form frames and the form model is unknown. Since some of the character strokes are either touching or crossing the form frames, we need to address the following three issues: (i) localization of form frames; (ii) separation between characters and form frames; and (iii) reconstruction of broken strokes introduced during separation. The form frame is automatically located by finding long straight lines based on a data structure, called block adjacency graph. Form frame removal and character reconstruction are implemented in this graph. When the same process is applied to a blank form, followed by the procedure of connected component extraction and clustering, a form structure-based template is automatically generated which includes form model, skew angle and preprinted data areas. Given the form template, our system can extract both handwritten and machine-typed filled-in data. Experimental results on three different types of forms demonstrate the performance of our system.

1 Introduction

Form processing is an essential operation in many business and government organizations. Forms are special documents typically used to collect or distribute data. A typical filled-in form consists of the following three components: (i) form frame, including black lines and blocks; (ii) preprinted data such as logos, large symbols and machine preprinted characters; and (iii) user filled-in data (including machine-typed and/or handwritten characters) which are located in pre-defined areas, called filled-in data areas, usually bounded by form frames and preprinted texts. We regard the first two components as preprinted entities. The useful data to be extracted from the forms are contained in the filled-in text. Distinguishing the filled-in data from the preprinted entities is one of the fundamental problems in form processing. Man-

ually keying in the filled-in data is a time consuming and tedious work because the number of forms to be processed in a typical application (e.g., tax forms) is usually very large. Interactive tools which provide a graphic interface for manually extracting filled-in data [1] do not solve this problem completely.

Machine understanding of form documents is an important problem in the advancement of office automation. Advances in handwritten character recognition have facilitated this problem to some extent. The major task in form dropout is to automatically extract filled-in data. In spite of the existence of some forms which are printed with dropout ink, a majority of the forms are printed without dropout ink for the sake of cost. Machine analysis of form documents should involve an intelligent form dropout procedure which removes preprinted entities while preserving the user filled-in data. This is a difficult task, especially when the filled-in characters are written or typed outside the data areas and touch or cross form frames or preprinted texts. We call this situation *field overlap* (between preprinted domain and filled-in domain).

Our goal is to extract filled-in data from a form in the presence of field overlap. Segmentation of individual characters is a very important problem in character recognition and document image processing. A number of methods have been presented to deal with this problem. Most of these methods, however, are applicable in situations where the machine typed or handwritten characters are touching each other. The segmentation problem of interest to us is to separate handwritten or machine-printed characters from preprinted entities such as form frames. Since the strokes of filled-in characters are either attached to or located across the form frames, the problem of character segmentation involves two issues: (i) separating characters from form frames and (ii) reconstruction of broken strokes introduced during separation. Because the pixels located in the areas where a character stroke crosses a form frame represent both a part of the stroke and a part of the frame, the stroke will be broken af-

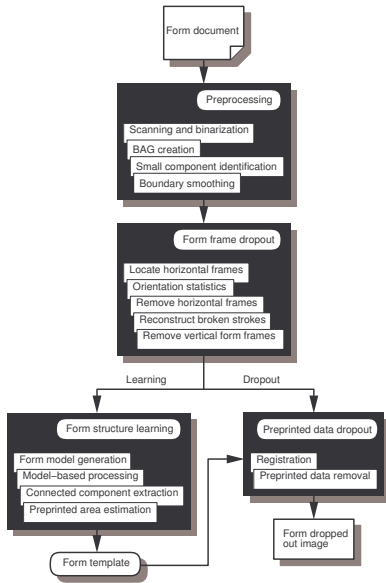


Figure 1: Block diagram of the proposed system.

ter removing the form frame. Therefore, we need to reconstruct the broken strokes. A few attempts have been made to solve this problem [2, 3, 4, 5].

We propose a system which can automatically capture form structure from a blank form and extract the filled-in data from filled-in forms. Forms are characterized by the presence of horizontal and vertical frames that delimit the usable space. A form frame can be either a simple straight line or a block area. In practice, filled-in characters often touch and cross the horizontal frames. Our system focuses on the horizontal frames which are the most obvious characteristics in a form. A binary image data structure, called the block adjacency graph (BAG), is used to locate the horizontal frames even when the form model is unknown and there is a modest amount of skew introduced during scanning. Figure 1 shows a block diagram of our form dropout system.

2 Data Representation

The key information that we utilize to initiate the form dropout is the location of horizontal frames. Therefore, the image representation should be chosen to facilitate extraction of horizontal frames. On the other hand, a typical A4 size document will produce a digital file of up to 1 MB after scanning it in a binary mode at 300 dpi. Therefore, the scanned image should be compressed and filtered.

Yu et al. [6] describe a block adjacency graph (BAG) data structure which is a graph-based representation of a binary image. Figure 2(b) is the BAG

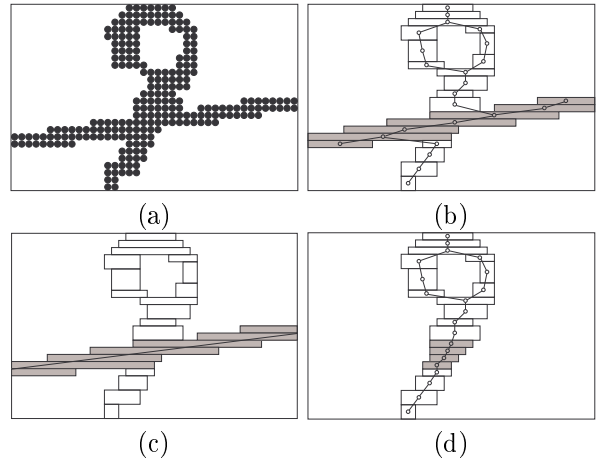


Figure 2: The BAG representation: (a) raster image; (b) BAG; (c) locating a horizontal form frame; (d) removing horizontal form frame and recording the connectivity.

representation of the raster image shown in Fig. 2(a). In this graph each node represents a block containing one or more connected run lengths with the same start and ending positions, and each edge gives a two-way connectivity information of nodes. With the use of BAG, we are able to find long horizontal frames by finding nodes (instead of conventional run length) whose length is longer than a threshold.

3 Form Dropout

Since most field overlap takes place between the horizontal form frames and filled-in characters, we deal with horizontal frames first. They are located and removed from the BAG. At the same time, connections between frames and character strokes are identified and the strokes broken during frame removal are reconstructed. After this process, some vertical form frame segments originally separated by those horizontal frames will become connected. However, it is easy to locate and then remove these long vertical lines.

Horizontal Form Frames Any horizontal straight line will produce some long horizontal run lengths in its digital representation even if it is slightly skewed (see Fig. 2(a)). We first detect and then remove a horizontal line by collecting a group of connected blocks in the BAG, each with (i) horizontal size larger than t_h , (ii) vertical size less than t_v , and (iii) the total length of the component formed by this group of blocks longer than t_l . Most horizontal form frames can be detected by this method. Figure 2(c) shows how a horizontal line is located using this method. The skew angles of the horizontal lines in the image are statistically averaged. When removing a detected

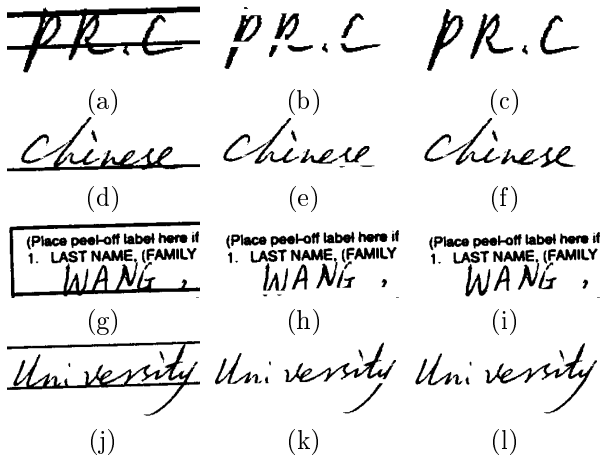


Figure 3: Examples of character reconstruction. First column shows segments of the input image, second column shows the result after the form frame removal and the last column shows the reconstructed characters.

horizontal frame, information regarding touching or crossing strokes and vertical frames connecting the horizontal frame is recorded and the connectivity information is stored.

Stroke Reconstruction In the areas where the characters touch or cross form frames, they share some pixels. During the removal of form frames, some of the strokes of these touching or crossing characters will be broken, which is likely to increase the error rate of the character recognition module. After removing the form frames, we perform character reconstruction to restore the broken strokes at the top or bottom of the characters. This reconstruction process is independent of the character recognition module. On the other hand, we regard a vertical form frame as a vertical segment between two horizontal form frames. To identify a vertical frame, we should link these segments when removing horizontal frames.

A character is said to touch the frame if no stroke from the character appears on both sides of the frame. If the width of the touching link between a stroke and a horizontal frame is less than the average stroke width W , then we do not have to reconstruct this stroke. Otherwise, it is extended by a distance equal to one half of the line width while the width of the stroke smoothly converges to $W/2$. A few examples of this reconstruction are given in Fig. 3.

The most difficult reconstruction is encountered when the broken strokes are caused by a stroke crossing the frame. We categorize these broken strokes into two classes. The first category is called “one-to-one” where the components in a pair match each other

across the frame (e.g., character ‘y’ in Fig. 3(j)). The other type of broken strokes is “multiple-to-one” or “one-to-multiple” where one component on one side of the crossed frame matches multiple entities on the other side. The character ‘W’ in Fig. 3(g) is an instance of this type of crossing.

For the first category of crossing, the broken strokes do not overlap horizontally because of the tilt of the handwritten stroke and the skew, e.g., the rightmost character ‘e’ in Fig. 3(e). Therefore, we should search in a local neighborhood to match the pair of broken strokes and interpolate the blocks between them (see Fig. 2(d)). In the second case (see, for example, the character W in Fig. 3(h)) one entity under the frame is related to two others on the frame. We first group these multi-strokes which touch the same frame and are close to each other. Then we restore the strokes by interpolating extra blocks to fill in the broken area. Vertical frames are processed as the first class of broken strokes. This allows us to link vertical segments into a longer one. By this processing step, major horizontal form frames are removed while keeping vertical frames intact.

Vertical Form Frames After removing horizontal frames, the long vertical frames are easily identified and removed. Some short vertical frames are difficult to distinguish from the long vertical strokes of characters such as numeral “1”. We use two sources of information in dealing with this problem. One is the linearity of the thin blocks representing a vertical segment. The other is the angle against a vertical reference line. For a vertical frame, this angle should be the same as the average skew angle detected during the process of locating horizontal frames. Most vertical frames can be detected and removed by this method.

Form Model Generation Most approaches to form processing are based on a pre-designed form structure [3, 7, 8]. The form structure can be obtained by one of the following methods: (i) a form structure file created by a word processing software, (ii) interactive learning, or (iii) scanning and automatic recognition of a blank form. For many applications, it is difficult to get a form structure file which provides the needed geometric information in a readable format. On the other hand, interactive learning is time consuming and tedious. Therefore, an automatic system for capturing form structure is very desirable.

A form structure in an image includes form model (consisting of form frames) and preprinted data (e.g., characters and logos) layout. During the detection of both horizontal and vertical frames, we are also

able to determine their positions. This establishes a rough form model. Based on the essential (domain-independent) knowledge in commonly encountered forms, the system can refine this rough model by (i) parallelizing all horizontal and vertical frames and letting the former be perpendicular to the later, and (ii) extending, retracting, connecting or aligning some frames to satisfy their geometric relationships. With the help of the extracted form model, we can locate and remove those small form frame components which were missed during the above processing by detecting the areas where the form frames are located.

4 Filled-in Data Extraction

To extract the filled-in data, the form processing system should have the form structure knowledge in advance. This information can be obtained by simply processing a blank form with the proposed system. First, the system needs to capture the specified form structure and record it as a template. Then this template is used for registering a given filled-in form.

Form Template We have noted that even preprinted objects often touch form frames in the digitized image. In such cases, the techniques mentioned above can be used to process a blank form image in order to automatically capture the form structure. After the removal of form frames, only the preprinted objects remain. We group or cluster the preprinted objects based on the distances between their corresponding bounding boxes. Together with the skew angle, these preprinted data areas and form frames are stored with the form template.

Form Registration Form registration is the process of aligning an input form with the form template. After locating and removing frames in an input filled form, the skew angle is estimated. All frames in the input form image are rotated in a clockwise direction by the estimated skew angle to have the same direction as those in the template. We use horizontal and vertical frames to get the translation values in vertical and horizontal directions, respectively. For details, refer to [9]. If no vertical form frames are available, then we obtain the horizontal translation value based on the average distance between the end points of long horizontal frames which are matched to each other after vertical translation correction.

Deleting Preprinted Data When mapping to the template space, an object in an input form image is called preprinted if its bounding box overlaps the preprinted data areas by more than 50%. These objects are deleted and the remaining objects in the image are assumed to be filled-in data and/or possibly noise. The filled-in data are fed to either a postpro-

cessing or a data compression module.

5 Experimental Results

Our system has been tested on three types of forms filled-in by hand (cursive and printed) and machine typed. They were scanned with a small amount of skew ($< 5^\circ$) at a resolution of 300 dpi. This level of resolution is required by most OCR software packages. Figure 4(a) is a form filled-in by hand with an image size of 2394×3157 . By automatically extracting the form template from a blank form shown in Fig. 4(b), the form dropout results are obtained in Fig. 4(c). Applying a handprinted OCR module [10] to these extracted filled-in data, we can obtain the identity of the filled-in characters as shown in Fig. 4(d). Note that the OCR results can be improved by some post-processing such as contextual analysis. The average CPU time on a SPARC 20 for this form is 8.28 seconds for the processing of form dropout. Other two examples of form dropout are shown in Fig. 5.

6 Conclusions

We have developed a form dropout system for automatic form processing. Our system has the following properties: (i) it can process a variety of forms and business reply mail cards that are either filled-in by hand or typed; (ii) it is able to accommodate a moderate amount of skew in the input images and no special skew detection [11] and deskew processes are needed; (iii) the system can dropout both horizontal and vertical form frames without using any particular knowledge of the form structure even when there is field overlap; (iv) it can correctly reconstruct most handwritten or machine-typed character strokes that are broken by the removal of form frames; (v) by processing a blank form, the system can automatically pick out the form structure which is then stored as a form template; (vi) based on the captured form template, the system can extract filled-in data for a class of input form images.

In practice, besides form frames, some of the filled-in characters may also touch preprinted text or objects. This is a more difficult problem to handle. Currently, we deal with it on a case by case basis. For example, if the filled-in data is a check mark which overlaps a preprinted box, then our system learns the positions of these boxes and saves the objects both inside and in the neighborhood of these areas. In more complicated cases, the system will capture the filled-in data, but will also extract some preprinted text at the same time. Some high-level or knowledge-based postprocessing needs to be employed to resolve these ambiguities.

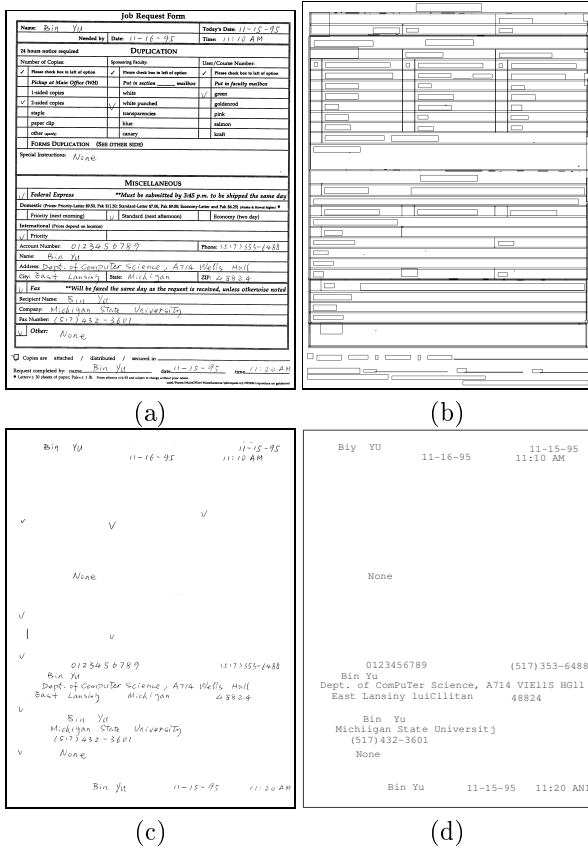


Figure 4: Form dropout: (a) input image; (b) form template; (c) results of dropout; (d) OCR results on (c).

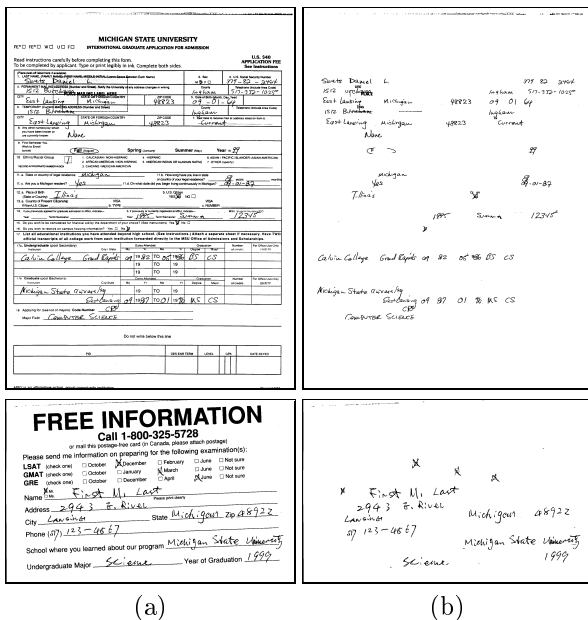


Figure 5: Form dropout examples: (a) input images; (b) dropout.

References

- [1] G. Leedham and D. Monger, "Evaluation of an interactive tool for handwritten form description," in *Proc. of the 3rd International Conference on Document Analysis and Recognition*.
- [2] G. Maderlechner, " 'Symbolic subtraction' from fixed formatted graphics and text from filled in forms," *Machine Vision and Applications*, vol. 3, pp. 457-459, 1990.
- [3] R. Casey, D. Ferguson, K. Mohiuddin, and E. Walach, "Intelligent forms processing system," *Machine Vision and Applications*, vol. 5, pp. 143-155, 1992.
- [4] D. Doermann and A. Rosenfeld, "The interpretation and reconstruction of interfering strokes," in *Proc. of International Workshop on Frontiers in Handwriting Recognition*.
- [5] Y. Y. Tang, C. Y. Suen, C. D. Yan, and M. Cheriet, "Financial document processing based on staff line and description language," *IEEE Trans. Syst. Man Cybern.*, vol. 25, pp. 738-754, 1995.
- [6] B. Yu, X. Lin, Y. Wu, and B. Yuan, "Isothetic polygon representation for contours," *CVGIP: Image Understanding*, vol. 56, pp. 264-268, 1992.
- [7] S. L. Taylor, R. Fritson, and J. A. Pastor, "Extraction of data from preprinted forms," *Machine Vision and Applications*, vol. 5, pp. 211-222, 1992.
- [8] D. Doermann and A. Rosenfeld, "The processing of form documents," in *Proc. of the 2nd International Conference on Document Analysis and Recognition*.
- [9] B. Yu and A. K. Jain, "A generic system for form dropout," to appear in *IEEE Trans. Pattern Anal. and Machine Intell.*
- [10] J. Mao, K. M. Mohiuddin, and T. Fujisaki, "A two-stage multi-network OCR system with a soft pre-classifier and a network selector," in *Proc. of the 3rd International Conference on Document Analysis and Recognition*.
- [11] B. Yu and A. K. Jain, "A robust and fast skew detection algorithm for generic documents," to appear in *Pattern Recognition*.