

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/220914403>

Extraction of Filled-In Data from Color Forms.

Conference Paper in Lecture Notes in Computer Science · September 1997

DOI: 10.1007/3-540-63460-6_105 · Source: DBLP

CITATIONS

3

READS

52

7 authors, including:



[Matsello Viacheslav](#)

Faculdade de Odontologia do Recife

10 PUBLICATIONS 25 CITATIONS

[SEE PROFILE](#)



[Dmitrij Schlesinger](#)

TUD Dresden University of Technology

35 PUBLICATIONS 341 CITATIONS

[SEE PROFILE](#)

Extraction of Filled-in Data from Colour Forms

I.Aksak¹, Ch. Feist², V.Kiiko¹, R.Knoefel²,
V.Matsello¹, V.Oganovskij³, M.Schlesinger¹, D.Schlesinger³, G.Stanke²

¹Institute of Cybernetics, Ukrainian Academy of Sciences, 40, Prospect Akademika
Glushkova, Kiev, Ukraine; e-mail: schles@image.kiev.ua

²Gesellschaft zur Foerderung angewandter Informatik, Rudower Chaussee 5,
Gebaeude 13.7, D-12484 Berlin; e-mail: stanke@gfai.de

³National Technical University, (Kiev Polytechnical Institute), 252056 Kiev, Ukraine,
Prospekt Peremohy, 37

Abstract. The paper describes an intelligent system for extraction filled-in data from colour conventional forms. Filled-in characters in these forms can touch or cross the form frames. The empty form description is acquired in this system by interactive learning followed by automatic processing and extraction filled-in data from input forms. Some steps of input form processing (form coding and form dropout) are carried out for any input form and other (form registration and broken characters reconstruction) are needed if the colour of the user filled-in data coincides with the colour of the empty form only. Experimental results showing the performance of the system are described.

1 Introduction

Intelligent form analysis is one of the most actual problems in document image processing. A typical filled-in form consists of pre-printed data (form frames, symbols and machine pre-printed characters) and user filled-in data (machine-typed or hand-written characters and check marks). Distinguishing of the filled-in data from the pre-printed ones and extraction these data from the input form is the major task in form processing.

Existing form processing systems can be categorised into two groups. The systems from the first group either use interactive tools for manually extracting filled-in data [1] or process specially designed forms [2]. The systems from the second group [3-7] are intelligent systems for processing of conventional forms, which have not been constructed and/or filled-in specially to facilitate reading by machine. As a result these forms usually contain filled-in data that touch or cross form frames or pre-printed entities. The problem of extraction filled-in data in these systems involves the following issues: empty form template, input form registration and pre-printed data dropout, reconstruction of broken strokes introduced during separation of form frames.

Form template means the capturing of empty form structure or description. The form structure knowledge is acquired by interactive learning in [4,5] and by automatic recognition in [6] using block adjacency graph (BAG) representation of this form.

Form registration is the process of aligning or calibrating an incoming form with the previously defined empty form template. During form registration the translational and rotational differences between the form template and the input form are minimized. Therewith, translation and rotation of input form image are carried out in [4,5] and the same operations but at the rectangles, describing pre-printed strokes in empty form, are carried out in [6].

Form dropout is the process of matching some objects or parts of input form as corresponding to pre-printed entities in empty form and deleting these objects.

Because some character strokes touch or cross form frames, these strokes will be broken after separating the form frame. The method of reconstruction of these strokes, based on skeleton, is introduced in [3]. In [4,5] the same process is performed by copying the pixels that lie just above and below the deleted line through the region where the line is being erased. In [6] form frame separation and character reconstruction are implemented by means of BAG.

The systems [3-7] have some distinctive features but are similar in that they deal with photocopied (in black and white) bilevel images of the forms only. At the same time beyond a doubt that the colour filled-in form has essentially more useful information when the colour of user pen differs from the colour of the empty form. In this case it is hopeful to extract filled-in data without need to reconstruct broken strokes and to handle the most difficult problem for [3-7], when filled-in characters touch or cross not only form frames but pre-printed text and objects also.

We propose a system for automatic extraction filled-in data from colour forms. In Section 2 the main parts of the system are described. Some of these parts (form colour coding and dropout) are used for every form and others (form registration and broken strokes reconstruction) are needed only if the colour of filled-in data coincides with the colour of the empty form. The performance of the system is shown in Section 3.

2 Filled-in data extraction

The applied content of filled-in data extraction problem is as following. There is a picture (in common case colour) of the empty (not filled) form. There is also a picture of this form with some inscriptions, added by a man, which are in general case coloured too. On the base of the certain information about the empty form the machine technology for extraction these inscriptions (filled-in data) is to be developed. In other words, it is necessary to transform a filled document into the picture, which would be produced, if the inscriptions would be written by a black pen on a white background. Processing steps of the system that solve this task are described in the following sections.

2.1 Form template

To extract filled-in data the following information about the empty form in general case must be acquired:

a). List C of colours, which exist on the document, and the coding table, that for every three-tuple (r, g, b) of bytes, defining the intensities of red, green and blue lights, defines the set $C' \subset C$ of colours, which this (r, g, b) can be represented by. It is supposed, that the variables r , g and b take their values from the interval $[0, 31]$, so the coding table consists of $32 \times 32 \times 32$ lists and each list has a length not more than 16 elements. On the base of this coding table the initial rgb -presentation of the empty form must be transformed to another one, in which for every pixel t the certain list $e(t)$ of the names of colours is defined.

b). List of horizontal and vertical lines of the empty form frame. These lines are represented by coordinates and the colour of the corresponding "bounding" rectangles in this form.

c). List of text strokes in the empty form. This list is represented in the same format like the list of lines.

d). List of rectangles covering the characteristic parts of the form. These rectangles are used while aligning the incoming form with the previously defined form template.

This information is generated interactively on the base of filled or empty form, that was scanned in *rgb*-codes. Form template is constructed only once, and then it is used for automatic processing of input filled-in forms. Separate steps of this processing are described in Sections 2.2-2.4. The principles of construction of the coding table from *rgb*-codes to colour codes are as follows.

Let r, g, b be variables that define the quantities of red, green and blue lights in pixels of the scanned picture. Let rgb be three-tuple of these variables and RGB be the set of all such three-tuples, i.e. the set of points with integer coordinates in the 3-D cube. For subsequent processing it is necessary to transform the rgb -representation of the scanned incoming form into the following one. Let C be the set of the names of colours, observed in the picture of the empty form, for example $C = \{\text{white, black, brown, yellow}\}$. For analysis of the filled-in form it is necessary to indicate for every pixel of this form not the rgb values but some list of the names of colours (for some pixels this list can be empty), i.e. some subset $C' \subset C$. For such transformation of rgb presentation to C' one it is necessary to define in the 3-D RGB cube for every colour $c \in C$ the region $RGB(c)$, containing all rgb -tuples, which this colour can be represented by. The region $RGB(c)$ is defined for every colour c , as the region of the certain type on the base of the learning information from user. Every region $RGB(c)$ is the polyhedron, every side of which is parallel to the one of the following thirteen planes:

$$\begin{aligned} r = 0, g = 0, b = 0, r + g = 0, g + b = 0, b + r = 0, r - g = 0, g - b = 0, b - r = 0, \\ r + g + b = 0, r - g + b = 0, r + g - b = 0, -r + g + b = 0; \end{aligned}$$

The learning information is received from the user who observes rgb -presentation of the picture and points out some pixels having by his opinion the certain colour c . On the base of this information the program defines the set $U(c)$ of indicated by user rgb -tuples and then constructs the region $RGB(c)$, which represents the minimal polyhedron of defined above type and includes the set $U(c)$.

2.2 Colour coding

Colour coding operation has the colour picture as input data, in which for each pixel the rgb -code is given. This representation of the picture is transformed to another one, in which for each pixel the list of names of colours, which are observed in this pixel, is given. On the next steps of the software the picture is presented only in such representation. The colour coding operation is based on previously constructed during form template coding table.

2.3 Form registration

The form registration consists of two parts: coarse and fine registration. The first part results in such displacement and rotation of the whole input form picture, that it will in the best way correspond to the previously defined set of lines in the empty form. The second part results in correction of the position and parameters of separate lines in empty form template for every incoming filled-in form, because the positions of lines and their thicknesses in the input form can be slightly different from the lines in the empty form.

The procedure of rotation of the picture is rather non-trivial part of aligning the input form with the empty one. It is known, that pixels coordinates after rotation are not integer and if to perform the rounding of these coordinates not correctly, the situation can occur, when either more than one pixels or neither can be mapped into the certain pixel of the picture after rotation. In this case the rotation turns to be irreversible that, firstly, contradicts to a reasonable understanding of the rotation and, secondly, destroys the initial information about the picture. To avoid these lacks we represent the rotation as the following three successive operations:

- 1) displacement of the rows of the picture: $x_{fir}=x+k1*y; y_{fir}=y;$
- 2) displacement of the columns of the picture: $x_{sec}=x_{fir}; y_{sec}=y_{fir}+k2*x_{fir};$
- 3) displacement of the rows of the picture: $x_{fin}=x_{sec}+k3*y_{sec}; y_{fin}=y_{sec}.$

Values $k1$, $k2$ and $k3$ are parameters of these operations. If the following relations

- a) $1+k2*k3 = \cos\varphi;$ b) $k1+k3+k1*k2*k3 = \sin\varphi;$
- c) $-k2 = \sin\varphi;$ d) $1+k1*k2 = \cos\varphi$

are hold, the equalities $x_{fin}=x_\varphi$ and $y_{fin}=y_\varphi$ are valid for every x and y . A solution of these relations is $k1 = \text{tg}(\varphi/2); k2 = -\sin\varphi; k3 = \text{tg}(\varphi/2)$ and implementation of image rotation as a superposition of specified three operations turns to be reversible for digitized images.

The procedure of fine registration of the filled-in form is as follows. Let C be a list of colours and ρ be a coloured picture. It means, that for every pixel t the list of colours $\rho(t) \subset C$ is defined. Let L be the following set of horizontal lines. Every line $l \in L$ is determined by the three-tuple (h_1, h_2, h_3) , such that $h_1 + h_2 + h_3 = h$, $h_1 > 0$, $h_2 > 0$, $h_3 > 0$. This three-tuple determines a picture of the line l that contains five horizontal stripes B_1, L_1, S, L_2, B_2 , as it is shown in Fig. 1a.

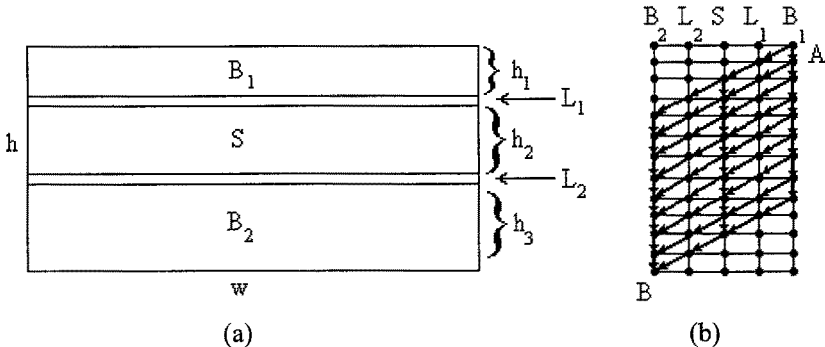


Fig. 1. Five stripes (a) of the horizontal line and the graph (b) representing the set of these lines.

The stripe $B_1(B_2)$ consists of $h_1(h_3)$ rows of pixels; for every pixel t from B_1 or B_2 holds $l(t) = B$, where B is the predefined list of background colours. Both L_1 and L_2 consist of one row of pixels and for every of these pixels t any colour is allowable, i.e. $l(t) = C$. The stripe S consists of h_2 rows of pixels and for every pixel $t \in S$ holds $l(t) = \{a\}$, where a is the predefined colour for the given line. The set of all possible pairs of positions of the stripes L_1 and L_2 defines the set L of pictures, i.e. the set of lines. For every line l and every picture ρ their dissimilarity $R(\rho, l)$ is defined

by the following way. A pixel t will be referred as a strange one, if it satisfies the condition $(\rho(t) = \emptyset) \vee (\rho(t) \not\subset l(t))$. The dissimilarity $R(\rho, l)$ is a total amount of the strange pixels in the picture.

The problem under solution is as following: for the given picture ρ the line $l^* \in L$ is to be found, which minimizes the dissimilarity $R(\rho, l)$, i.e. $l^* = \arg \min_l R(\rho, l)$. Time for direct calculation of this expression is of order

$h^3 \times w$, because an amount of lines in L is of order h^2 and calculation of R for every line requires $h \times w$ operations. Because the problem under consideration is reduced, as it will be shown below, to searching shortest path on the certain oriented weighted graph, the time for solution of this problem can be of order $h \times w$.

The set of lines L may be represented by the following oriented graph G , shown in Fig. 1b.

The nodes of this graph are represented by the nodes of two-dimensional grid. Every node has coordinates $(i, k), k \in \{B_1, L_1, S, L_2, B_2\}, 0 \leq i < h$. It means that every column of graph G corresponds to the certain stripe and every row corresponds to the certain row in the picture. The edges of the graph are displayed by the arrows in Fig. 1. Every path from the node A to the node B corresponds to the certain line from the set L . This isomorphism is defined by the following sentence: if the path on this graph passes through the node (i, k) , the i -th row of pixels belongs to the stripe k .

The dissimilarity function $R(\rho, l)$ must be represented by the weights of nodes, so that for any line $l \in L$ the value $R(\rho, l)$ is equal to the total weight of the path, which the line l is represented by. This requirement is satisfied if the weight $v(i, k)$ of the node (i, k) is defined by the following way: $v(i, k) = 0$, if $k=L_1$ or $k=L_2$, and $v(i, k)$ is the amount of the strange pixels in the i -th row for other k .

The calculation weights of all nodes takes a time of order $w \times h$ of the pixels in the picture. For the above described case of graph the search of the best path on the base of dynamic programming takes a time proportional to the number of graphs nodes, i.e. to the number h of rows in the picture. The program solves the above considered task for every line in the empty form and has as a result the new position for the line in this form.

2.4 Form dropout

This operation is carried out on the base of the following two pictures:

- the registered input form picture, in which for each pixel t the list $\rho(t)$ of colours is given, which are observed in this pixel;
- the empty form picture, in which for each pixel t the certain list $e(t)$ of names of colours is also given. This list consists of the names of the colours in the pixel before writing the inscriptions in the document.

Form dropout consists in the following transformation of the input form picture. The colour of every pixel t in this picture is changed to black colour, if the following condition

$$(\rho(t) = \emptyset) \vee (\rho(t) \not\subset e(t))$$

is satisfied, and to white colour otherwise. The first part in this condition means, that *rgb*-code in the pixel t of the picture under analysis does not correspond to any

colour, permitted in the empty form. The second condition means that in the pixel t of the input form some colour is observed, which is not allowable for the given pixel, although this colour is allowable for the form at all.

2.5 Broken stroke reconstruction

In the areas, where the inscriptions touch or cross form frames, they share pixels. During the form frame separation these pixels will be erased, if the colours of inscriptions and of form frames coincide, and as a result some of touching or crossing characters will be broken. In this case after the form frame separation we perform reconstruction of the broken strokes.

For every horizontal line of form frame this reconstruction consists of the following stages. The first stage results in noise removing on the upper and lower boundaries of erased line. During the second stage thicknesses and directions of line segments touching to upper and lower boundaries of this line are defined. These line segments consist of the runs of horizontal black pixels to the right and to the left of which white pixels are located. Two runs are called neighbouring if one of the pixels of the first run is neighbouring to some pixel of the second one. The direction of the line segment is defined over the sequence (from 2 to 5) neighbouring runs located either above of the upper or below of the lower boundaries of the line under processing.

The final and the most important stage consists in joining (filling the gap) of some of these line segments. At the beginning different pairs of line segments correspondingly above and below of the erased line are considered and two line segments of the pair are joined if they are crossed (taking in consideration their thicknesses) while extending one or both these line segments into erased region. Filling the gap between two line segments results in replacing of white pixels in black ones inside quadrangle between these line segments if their thickness are approximately the same. Otherwise joining is carried out by extending of the line segment with the lesser thickness. After joining line segments at upper and lower boundaries the pairs of neighbouring line segments both locating at upper or lower boundary are considered. These line segments are joined if certain conditions are satisfied, which depend on the directions, thicknesses and the distance between line segments.

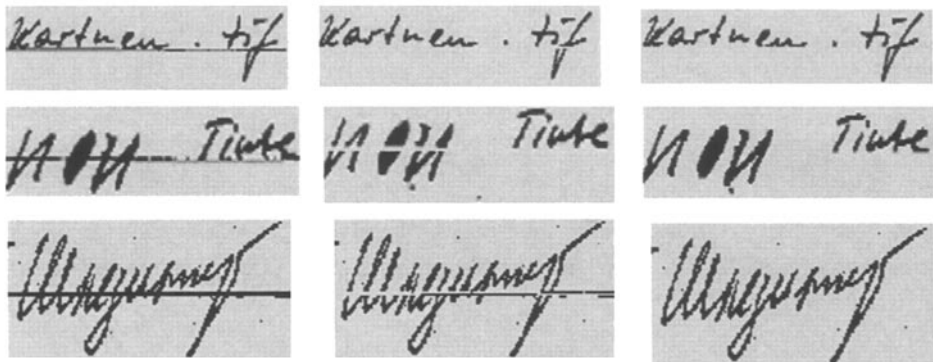


Fig. 2. Examples of character reconstruction.

A few examples of horizontal strokes reconstruction are given in Fig. 2. First column in this figure shows segments of the input image, second column shows the result after the form frame separation, and the last column shows the reconstructed characters. The reconstruction of characters having broken parts after separation of vertical lines of form frame is similar to considered above.

The resulted picture of extracted inscriptions contains inevitably the noise in the form of thin lines, whose thickness is one pixel, arising on the boundary of lines and inscriptions of the document. This noise is eliminated by rather obvious procedure.

3 Experimental results

The system has been developed for the OS MS Windows and tested on more than 30 forms of 6 different types filled-in by hand. These forms were filled in by different people, using ball-point pen, pencil or ink pen, and were scanned with a small amount of skew ($< 10^\circ$) at 240 dpi. The example of filled-in blue ink-printed form is shown in Fig. 3a. This form was filled in by blue pen in the upper part of the form, by red pen in the middle and by green pen in the lower parts of this form. The picture of the form is of size 1380 x 1900 (15 x 20 cm) and as it is shown in Fig. 3a the handwritten characters touch the form frames in all the four directions (top, bottom, left and right). Processing of this picture was carried out under the assumption that the colour of user's pen does not differ essentially from the colour of the empty form and because of that the whole number of processing operations were implemented. As a result it takes tens of seconds on a Pentium IBM PC to process this picture. Otherwise if the colour of the user's pen does not coincide with the colours in the empty form, the operations of input form registration and broken characters reconstruction have not to be used, and because of that the running time is drastically reduced.

The form dropout results for Fig. 3a are shown in Fig. 3b. These results are occurred to be better in comparison with presented in [6], because extracted filled-in data in Fig. 3b contain no pre-printed entities, but the system [6] takes lesser time for processing the input form. While the subsequent modification of the system we would like to continue developing of the methods of automatic coding and registering of coloured forms, and as a result to reduce the time of their processing.

<p>Radisson SAS WILKOMMEN WELCOME 1908 96 22 08 96 SNeisinger Michail Institut für Kybernetik Ukraine 327 Alles gute! tauglich 814</p>	<p>1908 96 22 08 96 Shlesinger Michail Institut für Kybernetik Ukraine 327 Alles gute! Y Y tauglich 814</p>
---	---

Fig. 3. Input form (a) and result (b) of extraction filled-in data.

Acknowledgement

This research was fulfilled in a multinational project and was sponsored by the Ministry of Education, Science, Research and Technology of Germany which is gratefully acknowledged.

References

- [1] Leedham and D. Monger, "Evaluation of an Interactive Tool for Handwritten Form Description", Proc. Third International Conf. Document Analysis and Recognition, pp. 1,185-1,188, Montreal, 1995.
- [2] S.L. Taylor, R. Fritzson and J.A. Pastor, "Extraction of Data from Preprinted Forms", Machine Vision and Applications, vol. 5, pp. 211-222, 1992.
- [3] G. Maderlechner, "Symbolic Subtraction from Fixed Formatted Graphics and Text from Filled In Forms", Machine Vision and Applications, vol. 3, pp.457-459, 1990.
- [4] R. Casey and D.Ferguson, "Intelligent Forms Processing", IBM Systems Journal 29(3), pp. 435-450, 1990.
- [5] R. Casey, D. Ferguson, K. Mohiuddin and E. Walach, "Intelligent Forms Processing System", Machine Vision and Applications, vol. 5, pp. 143-155, 1992.
- [6] Bin Yu and Anil K. Jain, "A Generic System for Form Dropout", IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 18, pp. 1127-1134.
- [7] K. Franke, "Unterschriftenverifikation mit SIC Natura", Bildanswertung für Handel, Banken und Behörden, Berlin, 19./20.02.97, Fraunhofer IPK - Workshop.