

## On recognition of low quality texts

I. Guzhavina, V. Denisyuk, F. Murzin, A. Palyanov, J. Trelewicz

**Abstract.** An acute problem for archives and museums is providing high-quality digitized text from low-quality or damaged originals. The majority of companies working in the field of optical character recognition (OCR) focus on digitized texts of high quality, which represent the majority of processed data and the market for these companies' products. As a result, the recognition of very low quality digitized texts, in general, is outside the scope of interest of such companies. In this paper, we analyze several algorithms for recognition of very low quality digitized images of texts and the results of testing the algorithms with the texts.

### 1. Introduction

Optical character recognition (OCR) is the well known problem of pattern recognition and digital image processing [1]. Investigations in the area of printed and hand-written text recognition carried out worldwide during the last several decades [2, 3] resulted in the development of various OCR algorithms which have been proposed to achieve better recognition results. Among the most well-known are template matching, image signatures, geometric features and shape-based image invariants. Recently, software for text recognition has achieved rather good quality due to progress in both methodology and computational performance. Commercial OCR packages are mostly oriented to the recognition of high-quality digitized texts and achieve almost no errors. As a result, it is possible with these packages to work with texts written in multiple languages, some of which contain multilingual dictionaries for additional post-correction after individual character recognition. For example, ABBYY Lingvo supports up to 180 recognizable languages and 33 dictionaries.

However, there are highly-specialized problems in this area that are of significant practical interest but which cannot effectively be solved by the usual methods. One of them is the recognition of old texts and documents from archives and museums, which are of significant historical value but often of a quite low original image quality. Such texts may have been manually typeset, so that the characters within one word can be of varying contrast and clarity. The original document may be old with partial physical damage, or the document could be a low-quality photocopy or mimeograph copy, which shows variations in toner density and character spread. Any of these factors can contribute to less than acceptable OCR [4, 5]. Tested with a set of examples of archive images of average low-quality, ABBYY FineReader was

able to recognize only about 30% of characters per page for the majority of them. After manual correction via brightness, contrast and gamma tuning and a slight blurring, this number increased up to 75%, which is still not sufficient for full-automatic word recognition.

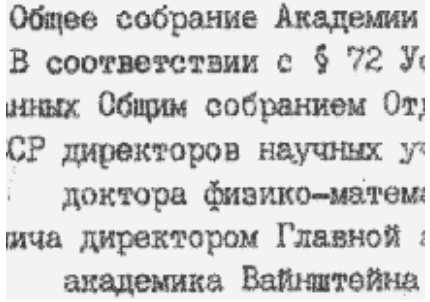
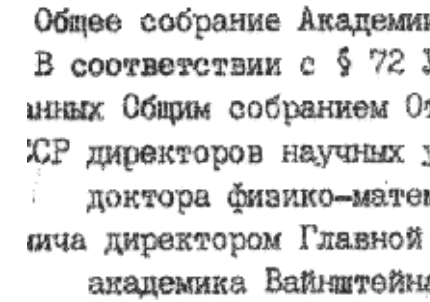
To solve this problem, we are working to develop a specialized software tool combining new approaches and known algorithms, maximally efficient for the spectrum of situations mentioned above, which should provide complete automatic recognition of texts in such documents. First we plan to implement pre-processing and denoising [6] and then hierarchical page segmentation [7] (identification of areas corresponding to lines, followed by words and finally by single characters boxes). This approach allows individual correction for characters with abnormal properties, e.g. very light or very dark, but usually recognizable. As a result, sometimes we can correctly recognize a character in such unclear condition as to be difficult to recognize even with the human visual system if considered as a single object, but which can be restored taking into account the word which contains it or a more global context. It is because of this feature of context-dependent restoration that we work also to realize dictionary-based correction after or combined with single character recognition. The typewritten characters of different types occurring in documents are used to produce a set of actual fonts, which should be detected automatically depending on its geometrical properties at the stage of recognition. We also provide minimal multilingual functionality for recognition of Russian and English texts in the same document – taking into account that some letters may be undistinguishable, such as Russian “p” and English “p”, and could be restored only from the context – recognition of punctuation marks and special symbols, and detection of areas containing non-text graphical objects or hand-written text.

## **2. Some image processing algorithms**

### **2.1. Linear contrast**

We use linear contrast to normalize the points of the initial image into the brightness range of the interval  $[0, 255]$ . If originally the image has a relatively narrow interval of grayscale gradation, for example, from 100 to 230, then the letters of the document will have poor contrast, especially in relation to noise in the background. After normalization processing, these disadvantages are significantly reduced.

Let us assume that the minimal and maximal brightness of the initial image are equal to  $x_{\min}$  and  $x_{\max}$ , respectively. If these parameters or one of them are essentially different from the boundary values of a brightness interval, then the image is low-contrast, which hinders recognition, even for a person.

	
$x_{\min} = 26, \quad x_{\max} = 243.$	$y_{\min} = 0, y_{\max} = 255$
<b>a)</b>	<b>b)</b>

**Figure 1.** Example of linear contrasting

We perform normalization by linear contrast using the following affine transformation of the brightness :

$$y = a \cdot x + b, \quad (1)$$

where the parameters  $a$  and  $b$  are defined by the desirable values of minimal  $y_{\min}$  and maximal  $y_{\max}$  target brightness. Thus it is necessary to solve the system of equations

$$\begin{cases} y_{\min} = a \cdot x_{\min} + b \\ y_{\max} = a \cdot x_{\max} + b \end{cases}, \quad (2)$$

to obtain the parameters of transformation  $a$  and  $b$ , and from (1) it is easy to obtain the following formula:

$$y = \frac{x - x_{\min}}{x_{\max} - x_{\min}}(y_{\max} - y_{\min}) + y_{\min}. \quad (3)$$

The result of linear contrast of the initial image represented in Figure 1.a is given in Figure 1.b.

A comparison of two images demonstrates much better visual quality of the processed image. Improvement is due to the use of a full dynamic interval of the screen after the contrasting process that is absent in the initial image.

## 2.2. Contrasting on the basis of a statistical threshold

When considering the histograms of various images of printed texts, we noticed that gradation of gray color corresponding to letters uses only a small part of the whole brightness range.

<p>11 марта в 15 часов  центра Академии наук СССР  по вычислительной технике  Новостка дня :</p>	Brightness	Percent
	10	0.0001%
	13	0.0001%
	82	0.006%
	102	0.03%
	121	0.4%
	141	1.5%
	161	2.2%
	181	2%
	200	1.7%
	220	1.7%
	240	2.7%
	255	87.6 %

**Figure 2.** Distribution of points of different brightness

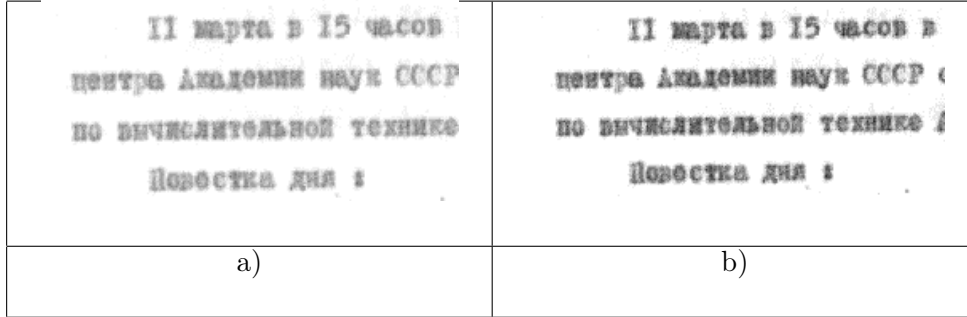
For these images, it is possible to use the following method of contrasting:

1. We determine a suitable threshold  $P$  for 2 %, for example,.
2. We break the range of gray in two parts:
  - (a) The first part is the values having brightness less than  $P$  (the Dark text).  
In the given example, this interval consists of values having brightness from 10 to 220.
  - (b) The second part is the values having brightness greater than  $P$  (the Light background).  
In the given example, this interval consists of colors having brightness from 240 to 255.
3. The background is brightened to white color.
4. The text remains unchanged.

We do not perform specific “denoising” of the background, since the low contrast of small letters and punctuation complicates the detection of noise. We have achieved much stronger results by simple affine transformation, followed by character detection.

### 2.3. Allocation of special points

By detecting the approximate contour of an object, it is possible to obtain information about its geometry. Using more or fewer special characteristic points on the contour, we carry out more or less detailed analysis of the image.



**Figure 3.** Example of linear contrast based on the described statistical analysis

Assume that there is a point with coordinates  $p = (i, j)$   $0 \leq i \leq n - 1$ ,  $0 \leq j \leq m - 1$ . We suppose that we have a grayscale picture given with the help of the brightness function  $S : M^2 \rightarrow R$ , which maps the value of its brightness for each point of the image. In this case,  $R$  is the set of integers from 0 to 255.

At the first stage, we carry out a transformation of a half-tone picture into a logical matrix  $T = (t_{ij})$ .

Let  $t_{ij} = \begin{cases} 0, & S(i, j) \geq \tau \\ 1, & S(i, j) < \tau \end{cases}$ , where  $\tau$  is the threshold of brightness.

Further we select a contour, leaving only boundary points and erasing all internal points. If  $\langle i, j \rangle$  is a boundary point, then

$$t_{i,j+1} + t_{i,j-1} + t_{i+1,j} + t_{i-1,j} < 4.$$

To describe the methods of allocation of characteristic points on a contour, we give some definitions.

Let us consider a point  $\langle i, j \rangle$ . This point admits various types of neighborhoods:

$$S_4(i, j) = \{\langle i \pm 1, j \rangle, \langle i, j \pm 1 \rangle\} \quad - \quad 4\text{-neighborhood},$$

$$S_D(i, j) = \{\langle i \pm 1, j \pm 1 \rangle\} \quad - \quad D\text{-neighborhood},$$

$$S_8(i, j) = S_4(i, j) \cup S_D(i, j) \quad - \quad 8\text{-neighborhood}.$$

Let us enumerate all elements around the given point. We assume that they are located on a circle with the center at this point, placed on the same angular distance  $\pi/4$  and numbered sequentially clockwise, starting from the top point of this circle. Instead of  $t_{kl}$ , let us write  $t_p$ , where  $p$  is the appropriate number,  $1 \leq p \leq 8$ , when  $\langle k, l \rangle$  is in the given environment of a point, i.e.  $\langle k, l \rangle \in S_8(i, j)$ .

Let us introduce the following functions:

1. The number of points in  $S_8$  и  $S_4$  of brightness equal to 1:

$$A_8(i, j) = \sum_{k=1}^8 t_k, \quad A_4(i, j) = \sum_{k=1}^4 t_{2k-1}$$

2. The number of triples in  $S_8$  of brightness equal to 1:

$$C_8(i, j) = \sum_{k=1}^8 t_{2k-1} t_{2k} t_{2k+1},$$

3. The similar characteristic number for  $S_4$ :

$$C_4(i, j) = \sum_{k=1}^8 t_{2k-1} t_{2k+1},$$

4. The number of 8-connectivity:

$$N_{C_8} = A_8(i, j) - C_8(i, j);$$

5. The number of 4-connectivity:

$$N_{C_4} = A_4(i, j) - C_4(i, j)$$

Then, by means of the numbers of connectivity, the characteristic points of a contour are defined as follows:

$$N_C(i, j) = \begin{cases} 0, & \text{isolated point;} \\ 1, & \text{final point;} \\ 2, & \text{binding point;} \\ 3, & \text{point of branching;} \\ 4, & \text{point of crossing;} \end{cases}$$

where  $N_C(i, j)$  means  $N_{C_4}$  or  $N_{C_8}$ , depending on the chosen type of connectivity.

Then the points  $\langle i, j \rangle$  such that  $N_C(i, j) \neq 2$  are informative, i.e. it is necessary to allocate them for the considered object. The condition  $N_C(i, j) = 2$  does not provide useful information.. In this case, we have usual binding points. If the contour has a break at the given point, then it is expedient to allocate it., Otherwise we do not allocate, since such points provide less information. As a result, some non-informative points are effectively deleted. These may be the separate isolated points, points forming ledges on contours, etc.

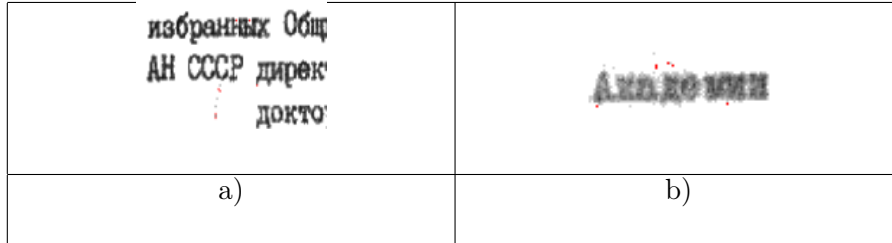


Figure 4. Examples of non-informative points

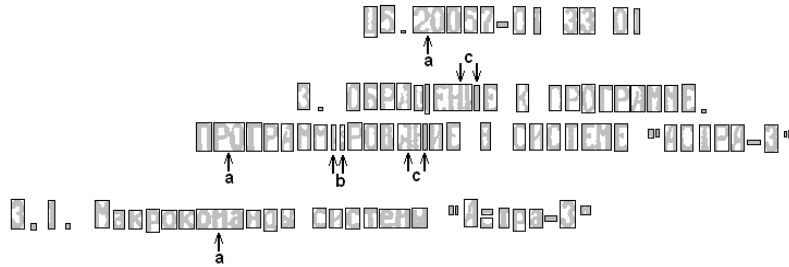
### 3. Optical character recognition

The research in OCR usually includes two stages, “low-level” and “high-level”. The low-level stage involves extracting features from images, such as the boundary of a character or areas with the same texture. The “high-level” stage involves recognition of these objects with the extracted features. In this work we deal with both “low-level” and “high-level” stages, focusing on the moments which are critical for a specific task – to reach a low-rate-of-error in recognition of low-quality images, containing texts written with a typewriter. Ideally, we work to approach the high level of recognition of these texts by a human vision system. As part of our preliminary work, we analyzed disadvantages of attempting to recognize such test images using commercial packages such as ABBYY FineReader after image preprocessing; e.g., brightness/contrast/gamma correction, slight blurring, noise filtration.

#### 3.1. Analysis

We tested the functionality of ABBYY FineReader, which processes good-quality images quite well, for recognition of our low-quality images. The attempts to recognize our examples without preprocessing showed less than acceptable quality for about 30% of correctly recognized characters, but after manual tuning of each image with a combination of brightness/contrast/gamma correction, noise filtration and sometimes a slight blurring, the quality of recognition reached the value of about 75%, which, unfortunately, is still insufficient for our purposes. Analysis of faults gave two most frequent cases: single letters, which were split into 2 parts, mostly vertical, and several consecutive letters merged into one (see Figure 5). It should be noted that many of our test images contain many letters which are significantly lighter or darker than the average brightness of their neighbors. FineReader has an internal threshold, below which all pixels are considered black, and others white. As a result, with FineReader some letters were completely lost, and others became thick black spots, though individual correction of brightness or manually-tuned threshold values allowed successful recognition of these letters. We noted that even after fixing disadvantages described above, we

still need to supply the algorithm with one more stage – correction of a preliminary recognized text using a dictionary to select most probable variant for each uncertain letter.



**Figure 5.** Two most frequent types of letter box detection errors. Merged letters (a), split letters (b) and combination of these cases (c)

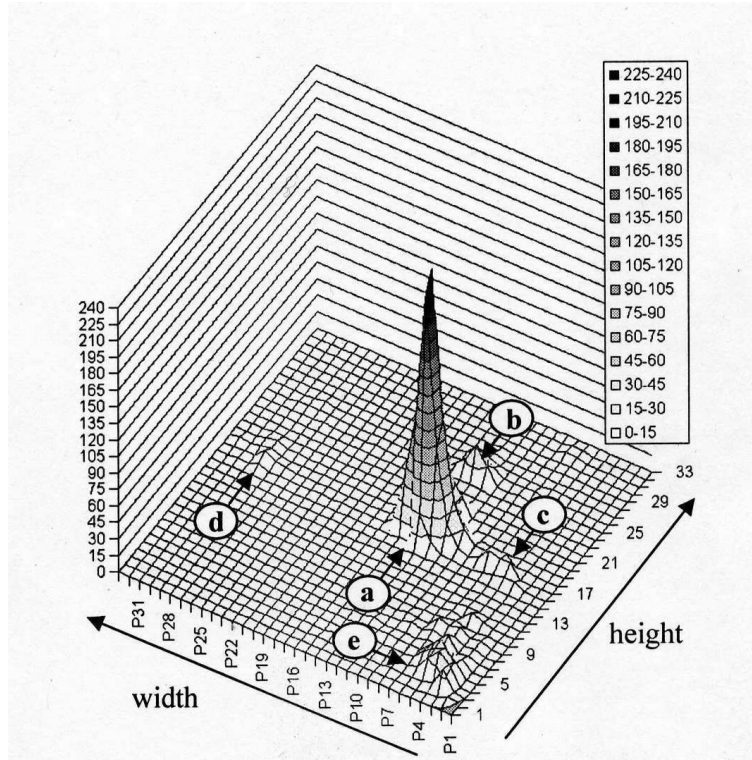
### 3.2. Low-level stage

In the general case of typeset text, the width of characters can differ even within the same font, the text can be a combination of different fonts of various sizes, upper- and lowercase characters are mixed and so on, but in the special case, when the text was created with a single typewriter, each symbol has a fixed size and we can try to detect it once on a page.

After studying the properties of typical texts to be recognized, we suggest the following algorithm for detection of symbol parameters, which proved to be precise and reliable. Defining a brightness threshold, e.g., some value of gray separating black and white, we can get a set of contours which in most cases (except for the errors earlier described) correspond to correct letters. Then for each contour we can get two simple parameters – width and height. Using all of the letters on a page, we build a surface  $F[w][h]$ , where every point reflects the number of times the box with a given width and height was found (Figure 6). Even on images of worst quality, the highest peak significantly overcomes all of the others, and its  $w$  and  $h$  correspond to the width and height of a lowercase letter (Figure 6a). All other peaks also have a certain significance, those with similar width to these letters and larger height correspond to capital letters (Figure 6b), and those with the similar height to these letters and narrow width – to characters such as “!”, “(”, “)”, “[”, “]”, “|” etc (Figure 6c). Next, the same height and more than twice larger width correspond to the case when two subsequent letters were merged into one box, which is as wide as 2 letters plus letterspacing (Figure 6d). Finally, the area with small values of width and height characterizes “.”, “;”, “’” and “?” and some small, rarely-occurring characters (Figure 6).

The next step is detection of the boxes corresponding to whole words, which allows us to solve the following problems:



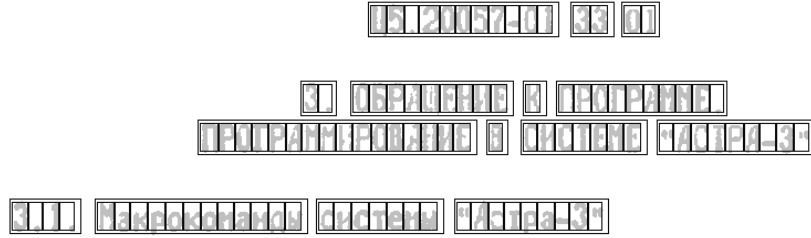


**Figure 6.** Preliminary distribution of width and height of letter boxes

1) For small-size characters corresponding to Figure 6e, we can identify their vertical position relative to the text line, so we can distinguish between “.” or “,” and “/” or “?”;

2) If we detect the start and end positions of a word, then knowing the character width and letter spacing, which is already determined, we can easily detect the number of letters in a word, and split it into a number of equal-size boxes independent to the size and type of a character at any position. Each of these boxes will correspond to the typewriter print hammer position, which makes high-level recognition much more precise.

The most simple, fast, and reliable way of whole word box detection which we have found is applying Gaussian blur with  $\sigma \sim 2 \dots 4$ . For this purpose, we have implemented the fast Gaussian blur algorithm [8]. After blurring, we determine the contours of areas with brightness lower than the threshold, giving areas corresponding to whole words instead of single letter contours, and then we split the words into letters as it was described above. An example is presented in Figure 7.



**Figure 7.** Successful detection of whole words and separate letter boxes for the same example which was shown in Figure 5

### 3.3. High-level stage

When the surrounding box for every letter is determined, the system has the input needed to recognize the characters. In the current version we have implemented the simple realization of the template matching method [9]. We compare the image inside the box with a set of standard letters  $\mathbf{F}$  (i.e., our extracted “font”), calculating the score reflecting the distance from the boxed image to each letter in  $\mathbf{F}$  (4). The variant with the best score is selected. The set  $\mathbf{F}$  was generated using the characters from the text, where each letter was built basing on several characters of a good quality, without significant defects. The set  $\mathbf{F}$  contains not only black and white pixels, but shades of gray, which makes score selectivity more precise:

$$Score_k = \max_{\delta=-30,-20,\dots,+20,+30} \left( \sum_{i=0..w-1, j=0..h-1} |M_{ij}^C + \delta - M_{ij}^{E,k}| \right), \quad (4)$$

where  $k$  is the index of a character in  $\mathbf{F}$ ,  $M_{ij}^C$  is the bitmap corresponding to the area defined by the current letter box and  $M_{ij}^{E,k}$  is the bitmap corresponding to the  $k$ -th letter of  $\mathbf{F}$ ,  $w$  is a box width and  $h$  is its height. The role of  $\delta$  is the correction of difference between brightness of the current letter and etalon. We also calculate the score for the current letter box position shifted by  $\pm 1$  in vertical and horizontal directions and take the best variant among all of these. Of course this is not an optimal algorithm in both speed and quality, but was implemented to check the efficiency of the low-level stage, which was extremely important, and the results of testing were quite encouraging.

We have tested the algorithm on the high-level stage; i.e. the result of all previous and the final step of text recognition at several most typical pages of texts from archive, at which it will be used in future. Currently, the low-level stage was tested on different typewriter fonts, and the high-level on  $\mathbf{F}$ , our most common font. We have processed several pages, calculating the quality of recognition for each page as the number of correctly recognized

characters divided by the total number of letters, and got values within the interval from 70 to 85%. The fragment of a typical source page, illustrating its properties and the result of text recognition, is shown in Figure 8. For this fragment the quality of recognition is 84.8%. Let us note that dictionaries in the given algorithm are not used yet. The usage of dictionaries is an additional possibility to improve the quality of recognition, which will be implemented in future iterations.

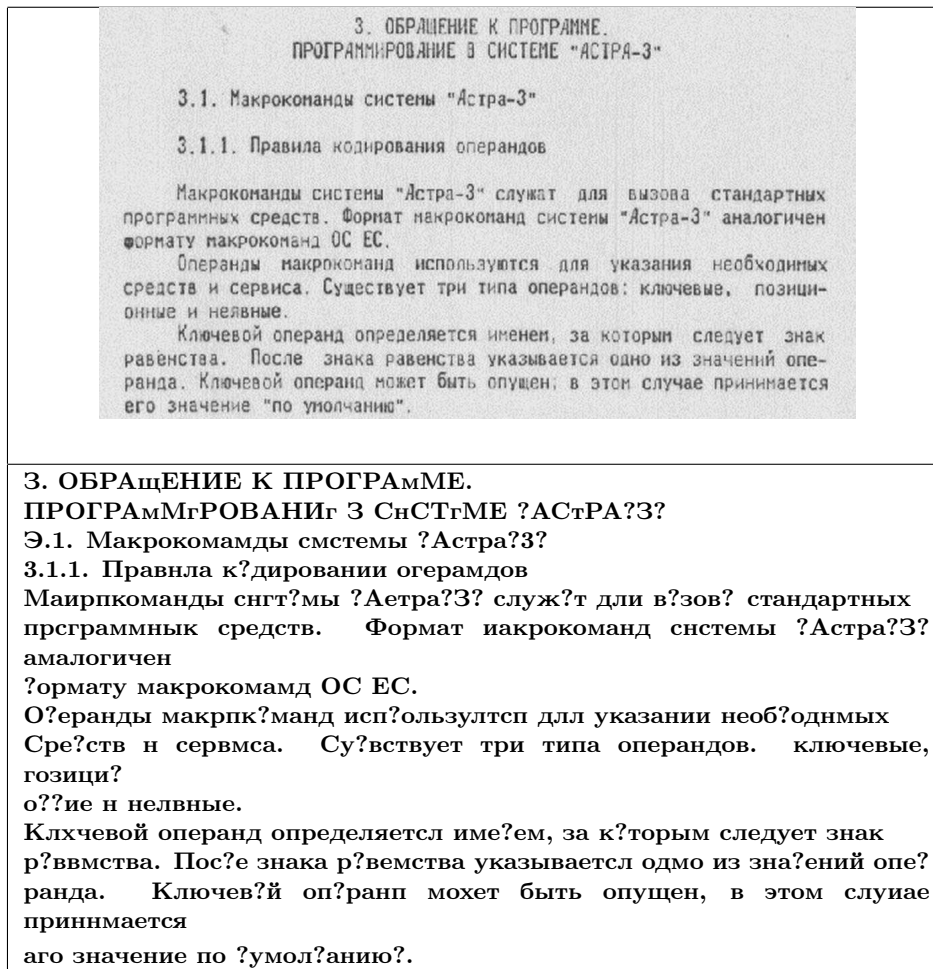


Figure 8. Example of a typical text fragment and the result of its recognition. The sign "?" corresponds to an unrecognized character

## Conclusion

In this work we have created a prototype of a program for text recognition intended for documents of low digitized image quality. This problem is important for recognition of old texts and documents from archives and museums, which may be manually typeset or, possibly suffering from physical degradation of the original paper (ageing of a material, partial physical destruction) or presented in the form of a photocopy or mimeograph copy with variations in toner density and character spread. This class of images is recognized by non-specialized OCR software with unacceptably low accuracy. Our purpose is to develop software intended to solve this problem, taking into account special features mentioned above. The program is written in C++, it is designed as a console application which can be compiled both at Windows and Linux; image loading is performed using FreeImage library which supports the most popular graphical formats. The output is generated in the form of a plain text.

One of the main results is that our program successfully detects text lines, whole words and single characters, and according to the test results, at this low-level stage we are almost free of errors; i.e., the level of errors is not a factor in accurate recognition of the words in the document. After the second, high-level stage of character recognition within the boxes, which is the final stage in the current version, we obtained accuracy of  $\sim 70\text{--}85\%$  correctly recognized characters. This is quite promising for the prototype, which will be tuned and optimized further, but for now this value prevents performing completely automatic processing. We should notice here that characters incorrectly recognized by our program are easily read by the human vision system within the whole word, but in most cases they cannot be restored correctly, if considered as a single symbol without context. As a result, our current work includes implementing post-correction of a processed text using a special dictionary, which will also help to distinguish between similar-looking letters in different languages, such as “p” in Russian and “p” in English and so on, because sometimes we meet the text written in both languages on one page. We plan to implement this functionality in the next version.

## References

- [1] Mori S., Suen C.Y., Yanamoto K. Historical review of OCR research and development // Proc. of the IEEE. – 1992. – Vol. 80. – P. 1029–1058.
- [2] Impedove S., Ottaviano L., Occhinegro S. Optical character recognition – a survey // Int. J. of Pattern Recognition and Artificial Intelligence. – 1991. – Vol. 5(1–2). – P. 1–24.

- 
- [3] Trier O.D., Jain A.K., Taxt T. Feature extraction methods for character recognition – a survey // *Pattern Recognition*. – 1996. – Vol. 29(4). – P. 641–662.
  - [4] Hartley R.T., Crumpton K. Quality of OCR for Degraded Text Images // *Comput. Res. Repos.* – 1999. – N 9902009.
  - [5] Flusser J., Suk T. Degraded image analysis: an invariant approach // *IEEE Trans. on Pattern Analysis and Machine Intelligence*. – 1998. – Vol. 20(6). – P. 590–603.
  - [6] Taghva K., Borsack J., Condit A., Erva S. The Effects of Noisy Data on Text Retrieval // *J. American Soc. for Inf. Sci.* – 1994. – Vol. 45(1). – P. 50–58.
  - [7] Pavlidis T., Zhou J. Page segmentation and classification // *Computer Vision Graphics and Image Processing*. – 1992. – Vol. 54(6). – P. 484–486.
  - [8] Young I.T., van Vliet L.J. Recursive implementation of the gaussian filter // *Signal Processing*. – 1995. – Vol. 44. – P. 139–151.
  - [9] Cowell J., Hussain F. Two template matching approaches to arabic, amharic and latin isolated characters recognition // *Machine Graphics & Vision*. – 2005. – Vol. 14(1). – P. 213–232.

