Published: Bogdan Savchynskyy, Sergiy Olefirenko. Partially supervised learning for text recognition problem. Control Systems and Computers, 2007(1), Kiev, pp.19-29

# Partially supervised learning for text recognition problem[*]

*Savchynskyy B.D., Olefirenko S.A.*

## Introduction

Despite its popularity, the problem of text recognition still contains considerable space for research. Recognition of images of text lines is, probably, one of the simplest examples in structural image recognition. That is why it is often used as a ground for implementation and experimental testing of new methods in structural recognition.

One of the fields of structural recognition that, in our opinion, needs further research, is an estimation of recognition algorithm parameters on the basis of a learning sample. A learning sample contains some number of images and corresponding recognition results. In case of text line recognition these results are not only corresponding character sequences, but also segmentations of the image into separate characters. Usually these segmentations should meet rather strict requirements: the same characters in different segments must be centered in the same way. It means that coordinates of corresponding pixels of the same characters in different segments should have the same values. Creation of such segmentations requires significant efforts and time of a teacher.

In this paper we will consider the problem of parameters estimation of a text recognition algorithm on a basis of the learning sample that contains only images and corresponding character sequences and does not contain segmentations of these images into separate characters. We will propose a formulation of the problem and an algorithm of its effective solution. This problem can be called a partially supervised learning because of partial information from the teacher in the learning sample. Formulation of the problem in such a way allows to significantly simplify the process of learning by reducing construction of the learning sample to typing of a text that corresponds to sample images.

The paper consists of four chapters, the first one is devoted to main definitions and formulation of the problem, the second one — to its solution. The third and the fourth chapters are devoted to experimental testing of algorithms and conclusions, respectively.

## 1 Definitions and formulation of the learning problem

Let's introduce notation, which will be used further in this paper.

[*] The paper was supported by the EU INTAS project PRINCESS 04-77-7347

A rectangular subset of two-dimensional integer grid, i.e. the set of image pixel coordinates, we will call a *field of view* $T$:

$$T = \{ (i, j) \mid i = \overline{0, W-1}, j = \overline{0, H-1} \}.$$

Value $W$ denotes *width* of the field of view, value $H$ denotes its *height*. Elements of the field of view $t \in T$ will be treated as two-dimensional vectors. In particular, addition operation is defined as a component-wise addition of corresponding coordinates.

Let $V$ be a set of pixel brightness values. We will call function $x\colon T \to V$ the *image*, width and height of the image equal to width and height of its field of view. We will distinguish two kinds of images: images given for a recognition and template images of characters. The second type of images will be discussed further. We will consider all the mentioned images having the same height $H$, but different widths.

Finite set $A_0$ will be called an *alphabet*. Its elements are characters of text. Sequence of alphabet elements $\bar{k} = (k_1, k_2, \ldots, k_L)$, $k_l \in A_0, l = \overline{1, L}$ we will call a *text line*. Notation $L_{\bar{k}}$ will stand for the length of a line $\bar{k}$.

A template image $e_k$ for each character $k \in A_0$ is defined on a field of view of height $H$ and width $d(k)$. Template widths $d(k), k \in A_0$ for all characters are fixed and known. The set of template images we will denote as $E$.

We will assume that an ideal, unnoised image, that corresponds to a given text line, is a horizontal sequence of character templates. These templates do not overlap and possible gaps between them including spaces between words in a text are filled with a background color.

To describe these gaps between character images formally we will introduce an additional element of the alphabet. We will call it an *insertion* and denote with $\kappa$. A template of the insertion is considered to have width $d(\kappa) = 1$, height $H$, and to belong to the template set $E$. The set $A_0 \bigcup \{\kappa\}$ we will denote with $A$ and call its elements $a \in A$ *symbols*. Thus symbol $a \in A$ is either an alphabet character $k \in A_0$ or the insertion $\kappa$.

A rectangular image fragment that contains an image of some symbol we will call a *segment*. Its height coincides with the height $H$ of an input image, its width is equal to the width of the corresponding symbol. Thus segment $s$ is defined by its left border coordinate $i \in \{0, \ldots, W - d(a)\}$ and a name $a \in A$. Left border coordinate and a name of a segment $s = (i, a)$ will be denoted with $i(s)$ and $a(s)$, respectively. Notation $S$ stands for the set of all possible segments on the input image. For a top-left corner of each segment $s$ notation $t_s$ will be used: $t_s = (i(s), 0)$.

Let us consider a sequence of segments $\bar{s} = (s_1, \ldots, s_N)$ of an arbitrary length $N$ such that these segments cover the whole field of view and are placed closely to each other:

$$\begin{cases} i(s_1) = 0; \\ i(s_{n+1}) = i(s_n) + d(a(s_n)), n = \overline{1, N-1}; \\ i(s_N) + d(a(s_N)) = W. \end{cases}$$

We will call such a sequence a *segmentation* of the image. Thus the segmentation can be regarded as a complete description of an ideal image. The set of all segmentations will be denoted with $\bar{S}$.

We have already described the process of construction of ideal text line image. Now we assume that an input image differs from the ideal one only by Gaussian noise with defined deviation $\sigma$, which is added in each pixel independently from others.

Now we will introduce some probability distributions and use for them a standard probabilistic notation, i.e. $p_{x|\bar{s}}(x|\,\bar{s};E)$ stands for the conditional probability of image $x$ under condition of known segmentation $\bar{s}$ (that determines an ideal image) parametrized by templates $E$; $p_{x,\bar{s}}(x,\bar{s};E)$ stands for the joint probability of the image $x$ and segmentation $s$; and $p_{\bar{s}}(\bar{s})$ stands for the a priori probability of segmentations. To simplify the notation we will omit the description indexes, i.e. we will write $p(x|\,\bar{s};E)$ instead of $p_{x|\bar{s}}(x|\,\bar{s};E)$.

According to the assumption, a probability $p(x|\,\bar{s};E)$, conditioned by segmentation $\bar{s}$, can be represented as a product of independent probabilities $p(x|\,s;E)$, conditioned by separate segments of this segmentation, and will be considered to be equal to:

$$p(x|\,\bar{s};E) = \prod_{n=1}^{N(\bar{s})} p(x|\,s_n;E) = \prod_{n=1}^{N(\bar{s})} \prod_{t\in T(s_n)} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{(x(t_{s_n}+t)-e_{a(s_n)}(t))^2}{2\sigma^2}\right\}, \tag{1}$$

where $N(\bar{s})$ is a number of segments in the segmentation $\bar{s}$ and $T(s)$ is a rectangular fragment of field of view which corresponds to the segment $s$.

The problem of an image $x$ recognition consists in search for the most probable segmentation $\bar{s}^*$ for the image $x$:

$$\bar{s}^* = \arg\max_{\bar{s}} p(\bar{s}|\,x;E) = \arg\max_{\bar{s}} p(x,\bar{s};E) = \arg\max_{\bar{s}} p(\bar{s})\cdot p(x|\,\bar{s};E).$$

As it is known [1], this problem can be solved by dynamic programming algorithm.

Unknown recognition parameters here are templates $E$ and a priori probability distribution $\{p(\bar{s}) \mid \bar{s}\in\bar{S}\}$ of segmentations. Learning of the recognition algorithm, which is actually a topic of this paper, consists in estimation of these parameters on the basis of a learning sample.

Before we proceed to formulation of the learning problem, let us note the connection between segmentations, symbol and character sequences. A symbol sequence $\bar{a}(\bar{s}) = (a_1,\ldots,a_N|a_n = a(s_n), n=\overline{1,N})$ corresponds to each segmentation $\bar{s} = (s_1,\ldots,s_N)$. From the symbol sequence we can obtain a character sequence by removing all insertions. Thus, for each character sequence $\bar{k} = (k_1,\ldots,k_N)$, $k_n \in A_0$ there is a set $\bar{S}[\bar{k}]$ of such segmentations that correspond to this sequence in a described way.

Now we proceed to the formulation of the learning problem. Let

$$D = \begin{pmatrix} x^1 & x^2 & \ldots & x^M \\ \bar{k}^1 & \bar{k}^2 & \ldots & \bar{k}^M \end{pmatrix}$$

be a learning sample that consists of $M$ input images and $M$ corresponding text lines.

Joint probability $p(x,\bar{k};E)$ of an image $x$ and a text line $\bar{k}$ is equal to a sum $\sum_{\bar{s}\in\bar{S}[\bar{k}]} p(x,\bar{s};E)$ of probabilities of all segmentations of the image $x$ that correspond to the text line $\bar{k}$. Thus, the probability of a learning sample $p(D;E)$ can be presented in the following form:

$$\begin{aligned} p(D;E) &= \prod_{m=1}^{M} p(x^m, \bar{k}^m; E) = \prod_{m=1}^{M} \sum_{\bar{s}\in\bar{S}[\bar{k}^m]} p(x^m, \bar{s}; E) = \prod_{m=1}^{M} \sum_{\bar{s}\in\bar{S}[\bar{k}^m]} p(\bar{s}) \cdot p(x^m |\, \bar{s}; E) = \\ &= \prod_{m=1}^{M} \sum_{\bar{s}\in\bar{S}[\bar{k}^m]} \frac{p(\bar{s})}{\sqrt{2\pi\sigma^2}} \exp\left\{ -\sum_{n=1}^{N(\bar{s})} \sum_{t\in T(s_n)} \frac{(x^m(t_{s_n}+t) - e_{a(s_n)}(t))^2}{2\sigma^2} \right\}. \end{aligned}$$

**Problem 1** *The problem of a recognition algorithm learning consists in finding such templates $E^*$ and a priori probabilities $p^*(\bar{s})$ of segmentations that maximize probability of the learning sample $D$:*

$$(E^*, p^*(\bar{s})) = \arg \max_{(E, p(\bar{s}))} \prod_{m=1}^{M} \sum_{\bar{s}\in\bar{S}[\bar{k}^m]} \frac{p(\bar{s})}{\sqrt{2\pi\sigma^2}} \exp\left\{ -\sum_{n=1}^{N(\bar{s})} \sum_{t\in T(s_n)} \frac{(x^m(t_{s_n}+t) - e_{a(s_n)}(t))^2}{2\sigma^2} \right\}. \quad (2)$$

The algorithm of an exact solution of the problem 1 is unknown to us. In this paper a non-supervised learning algorithm, described in [1], is used to solve the problem 1. As it is known, this algorithm guaranties finding only local extremum. But from practical point of view this is not a serious problem because quality of parameter estimation can be easily controlled visually.

However, the non-supervised learning algorithm, described in [1], can't be used for problem 1 solution directly, because it demands exponential on input image dimensions time and memory. In the next chapter it is described how this algorithm should be implemented to effectively solve the problem 1.

# 2 Learning task solution

First, we will formulate the non-supervised learning algorithm for problem 1 in the form as it is described in [1]. As it was already mentioned, this algorithm cannot be used directly in that form because of its considerable time and space complexity. After that we will transform it equivalently to decrease its complexity without altering the results of performed operations.

## 2.1 Base non-supervised learning algorithm

Before we proceed, let us introduce additional notations. We will use an equivalent notation $\bar{S}_m$ for the set $\bar{S}[\bar{k}^m]$ of all such segmentations that their symbol sequences after removing all insertions coincide with $m$-th text line $\bar{k}^m$ from the learning sample $D$. We will call segmentations from this set *allowable* for a given text line $\bar{k}^m$. Notation $\bar{S}_m(s)$ stands for such subset of the set $\bar{S}_m$ that consists only of the segmentations containing segment $s$.

The non-supervised learning algorithm is an iterative one. Upper index $r$ denotes values of parameters, which they take on the $r$-th iteration of the algorithm. Let $E^0$ be the initial values of symbol templates and $p^0(\bar{s})$, $\bar{s} \in \bar{S}$ be a priori distribution of image segmentations $\bar{s}$. Each iteration consists of two steps. At the first step (named *recognition*) a posteriori probabilities $\hat{\alpha}^r(x^m, \bar{s})$ of allowable segmentations $\bar{s} \in \bar{S}_m$ for each learning image $x^m$ are estimated:

$$\hat{\alpha}^r(x^m, \bar{s}) = \frac{p^r(\bar{s}) \cdot p(x^m|\bar{s}; E^r)}{\sum\limits_{\bar{s}\in\bar{S}_m} p^r(\bar{s}) \cdot p(x^m|\bar{s}; E^r)}, \quad m = \overline{1, M},\ \bar{s} \in \bar{S}_m. \quad (3)$$

At the second step (named *learning*) a priori probabilities of segmentations $p^{r+1}(\bar{s})$ and symbol templates $E^{r+1}$ are estimated according to formulas:

$$p^{r+1}(\bar{s}) = \frac{\sum\limits_{m=1}^{M} \hat{\alpha}^r(x^m, \bar{s})}{M}, \quad \bar{s} \in \bar{S}; \tag{4}$$

$$E^{r+1} = \arg\max_{E} \sum_{m=1}^{M} \sum_{\bar{s} \in \bar{S}_m} \hat{\alpha}^r(x^m, \bar{s}) \cdot \log p(x^m | \bar{s}; E^r). \tag{5}$$

Implementation of the non-supervised learning algorithm in the form (3)–(5) is impossible because values $\hat{\alpha}^r(x^m, \bar{s})$ should be calculated for all possible segmentations of the learning sample and their number increases exponentially with image dimensions. To optimize the algorithm we will modify it to deal with separate segments instead of segmentations.

## 2.2 Effective implementation of non-supervised learning algorithm

Let us assume that segments of any segmentation are independent, that is, the probability of the segmentation $\bar{s}$ is defined by the formula:

$$p(\bar{s}) = \prod_{n=1}^{N(\bar{s})} p(s_n), \tag{6}$$

where $p(s)$, $s \in S$ are a priori probabilities of segments. Obviously, the following equation is fulfilled:

$$p(s) = \sum_{\bar{s} \in \bar{S}_m(s)} p(\bar{s}), \quad s \in S. \tag{7}$$

Instead of probabilities of segmentations $\hat{\alpha}(x, \bar{s})$ we will calculate values $\alpha(x, s)$, $s \in S$, which are estimated a posteriori probabilities of segments:

$$\alpha^r(x^m, s) = \sum_{\bar{s} \in \bar{S}_m(s)} \hat{\alpha}^r(x^m, \bar{s}) = \frac{\sum\limits_{\bar{s} \in \bar{S}_m(s)} p^r(\bar{s}) \cdot p(x^m | \bar{s}; E^r)}{\sum\limits_{\bar{s} \in \bar{S}_m} p^r(\bar{s}) \cdot p(x^m | \bar{s}; E^r)}, \quad m = \overline{1, M}, \ s \in S. \tag{8}$$

By substitution of (1) and (6) into (8) we will obtain:

$$\alpha^r(x^m, s) = \frac{\sum\limits_{\bar{s} \in \bar{S}_m(s)} \prod\limits_{n=1}^{N(\bar{s})} p^r(s_n) \cdot p(x^m |\, s_n; e^r_{a(s_n)})}{\sum\limits_{\bar{s} \in \bar{S}_m} \prod\limits_{n=1}^{N(\bar{s})} p^r(s_n) \cdot p(x^m |\, s_n; e^r_{a(s_n)})}, \quad m = \overline{1, M}, \ s \in S. \tag{9}$$

It is clear, that calculations according to the formula (9) cannot be done directly, but later in the subsection 2.3 an effective algorithm of these calculations based on dynamic programming method will be proposed.

Let us proceed to the formula (4) and sum equation (4) over all segmentations that contain a fixed arbitrary segment $s \in S$. Then, according to (7) and (8), we will obtain:

$$p^{r+1}(s) = \frac{\sum\limits_{m=1}^{M} \alpha^r(x^m, s)}{M}, \quad s \in S. \tag{10}$$

Values $p^{r+1}(s)$ can be calculated directly according to this formula.

The set of templates $E$ consists of pixel values over templates of all symbols: $E = \{e_a(t) \mid t \in T_a, a \in A\}$. A condition of the local maximum of (5) is determined with the following system of equations:

$$\frac{\partial}{\partial e_a(t)} \sum_{m=1}^{M} \sum_{\bar{s} \in \bar{S}_m} \hat{\alpha}^r(x^m, \bar{s}) \cdot \log p(x^m|\bar{s}; E^r) = 0, \quad t \in T_a, a \in A.$$

After substituting values $p(x^m|\bar{s}; E^r)$ according to (1) and performing simple algebraic transformations we will consecutively obtain:

$$\frac{\partial}{\partial e_a(t)} \sum_{m=1}^{M} \sum_{\bar{s} \in \bar{S}_m} \hat{\alpha}^r(x^m, \bar{s}) \sum_{n=1}^{N(\bar{s})} \sum_{t \in T(s_n)} \frac{(x^m(t(s_n) + t) - e_{a(s_n)}(t))^2}{2\sigma^2} = 0,$$

$$\sum_{m=1}^{M} \sum_{\bar{s} \in \bar{S}_m} \hat{\alpha}^r(x^m, \bar{s}) \sum_{n=1}^{N(\bar{s})} \mathbf{1}_{\{a(s_n)=a\}} \cdot (e_a(t) - x^m(t_{s_n} + t)) = 0,$$

$$\sum_{m=1}^{M} \sum_{\substack{s \in S: \\ a(s)=a}} \sum_{\bar{s} \in \bar{S}_m(s)} \hat{\alpha}^r(x^m, \bar{s}) \cdot (e_a(t) - x^m(t_s + t)) = 0,$$

$$\sum_{m=1}^{M} \sum_{\substack{s \in S: \\ a(s)=a}} \alpha^r(x^m, s) \cdot (e_a(t) - x^m(t_s + t)) = 0.$$

It can be seen from the last equation that the extremum point is unique and in this point the global maximum of sum (5) is reached. Thus an optimal template for an arbitrary symbol $a \in A$ is constructed as a weighted average of corresponding image fragments with weights $\alpha(x^m, s)$ over all segments with a name $a$:

$$e_a^{r+1}(t) = \frac{\sum_{\substack{s \in S: \\ a(s)=a}} \sum_{m=1}^{M} \alpha^r(x^m, s) \cdot x^m(t_s + t)}{\sum_{\substack{s \in S: \\ a(s)=a}} \sum_{m=1}^{M} \alpha^r(x^m, s)}, \quad t \in T_a. \tag{11}$$

Thus, starting from the base algorithm (3)–(5), we obtained the algorithm that solves the same problem but estimates the different set of parameters, namely symbol template images and a priori probabilities of segments: $\{e_a, p(s) \mid a \in A, s \in S\}$. Unlike the base algorithm, this one can be effectively implemented: values $p(s)$, $s \in S$ and templates $e_a$, $a \in A$ can be calculated directly in accordance with formulas (10) and (11), and algorithm of calculation of $\alpha^r(x, s)$ values is proposed in the next subsection.

## 2.3 Algorithm of a posteriori segment probabilities $\alpha(x, s)$ estimation

In this subsection we will consider algorithm of segment probabilities $\alpha(x, s)$ calculation for an arbitrary image $x$ and corresponding text line $\bar{k}$. Let $L$ denote the length of line $\bar{k}$: $\bar{k} = (k_1, \ldots, k_l, \ldots, k_L)$. Probability $p(s) \cdot p(x|\, s; e_{a(s)})$ of a segment $s = (i, a)$ we will call a *penalty* for this segment and denote with $f(i, a)$. A penalty for a segmentation $\bar{s}$ is a product of penalties over all segments in this segmentation. We will distinguish segments with a name $a \in A_0$ from the alphabet, and segments-insertions. Segments of the first type will be called *significant*.

We will consider only segmentations which correspond to a text line $\bar{k}$, namely segmentations from the set $\bar{S}[\bar{k}]$. Such segmentations contain $L$ significant segments that correspond to text line characters, and arbitrary number of insertions. Let us introduce function $\operatorname{sgf}\colon \{1, \ldots, L\} \times \bar{S}[\bar{k}] \to S$, its value $\operatorname{sgf}(l, \bar{s})$

indicates $l$-th significant segment of the segmentation $\bar{s}$, namely the segment, which corresponds to the $l$-th character of the line $\bar{k}$. Furthermore, we will use two functions $b_L(l,\bar{s})$ and $b_R(l,\bar{s})$, $l=\overline{1,L}$, that point out left and right border coordinates of $l$-th significant segment, respectively.

For an arbitrary segmentation $\bar{s}$ the segment with left border coordinate 0 will be called the *beginning* of the segmentation $\bar{s}$, and segment with right border coordinate $W$ will be called its *end*.

From formula (9) follows that, for arbitrary segment $s=(i,a)$, value $\alpha(x,s)$ is a ratio of a total penalty for segmentations that contain segment $s$, and a total penalty for all segmentations. A numerator can be represented as a product of two sums: the sum of penalties for segmentation parts from the beginning of an image to the segment $s$, and the sum of penalties for segmentation parts from segment $s$ till the end. Segment $s$ can be either a significant segment or an insertion segment, and in these two cases calculation of $\alpha(x,s)$ differs. For the significant segment $s$ ($a(s)\in A_0$) the numerator of the formula (9) can be rewritten as follows:

$$\sum_{\bar{s}\in\bar{S}[\bar{k}](s)}\prod_{n=1}^{N(\bar{s})}p(s_n)\cdot p(x|\,s_n;e_{a(s_n)})=\sum_{\substack{1\le l\le L:\\ k_l=a}}\;\sum_{\substack{\bar{s}\in\bar{S}[\bar{k}](s)\\ \mathrm{sgf}(l,\bar{s})=s}}f(i_1,a_1)\cdot f(i_2,a_2)\cdot\ldots\cdot f(i_{N(\bar{s})},a_{N(\bar{s})})= \tag{12}$$

$$=\sum_{\substack{1\le l\le L:\\ k_l=a}}\left[\sum_{\substack{\bar{s}\in\bar{S}[\bar{k}](s)\\ \mathrm{sgf}(l,\bar{s})=s}}f(i_1,a_1)\cdot\ldots\cdot f(i,a)\right]\times\left[\sum_{\substack{\bar{s}\in\bar{S}[\bar{k}](s)\\ \mathrm{sgf}(l,\bar{s})=s}}f(i,a)\cdot\ldots\cdot f(i_{N(\bar{s})},a_{N(\bar{s})})\right]\times\frac{1}{f(i,a)},$$

where $\bar{s}=(s_1,s_2,\ldots,s_{N(\bar{s})}), s_n=(i_n,a_n)$. The penalty for the segment $s=(i,a)$ is included in both expressions bounded by parenthesis, so we divide the expression by this penalty to compensate this. Let us note here that conditions $\mathrm{sgf}(l,\bar{s})=s$, $b_R(l,\bar{s})=i+d(k_l)$ and $b_L(l,\bar{s})=i$ are identical, so later instead of the first condition we will use two others.

Expressions in square parenthesis depend on $l$, which is a number of $l$-th character in the text line, and segment $s=(i,k_l)$. We will denote the first of these expressions with $F_1(i+d(k_l),l)$ and the second one with $B_1(i,l)$, where $i+d(k_l)$ is right border coordinate of segment $s$:

$$F_1(i,l)=\sum_{\substack{\bar{s}\in\bar{S}[\bar{k}](s):\\ b_R(l,\bar{s})=i}}f(i_1,a_1)\cdot f(i_2,a_2)\cdot\ldots\cdot f(i-d(k_l),k_l), \tag{13}$$

$$B_1(i,l)=\sum_{\substack{\bar{s}\in\bar{S}[\bar{k}](s):\\ b_L(l,\bar{s})=i}}f(i,k_l)\cdot\ldots\cdot f(i_{N(\bar{s})},a_{N(\bar{s})}). \tag{14}$$

We will consider now all segmentations with $l$-th significant segment equal to $s(i,k_l)$ for an arbitrary pair $(i,l)$ of indexes. Value $F_1(i+d(k_l),l)$ equals to a total penalty for segmentation parts from the beginning to segment $s$ inclusive; $B_1(i,l)$ equals to a total penalty for segmentation parts from segment $s$ inclusive to the end.

In the case when $s$ is an insertion segment ($a(s) = \kappa$), formula (9) transforms into:

$$\sum_{\bar{s}\in\bar{S}[\bar{k}](s)} \prod_{n=1}^{N(\bar{s})} p(s_n)\cdot p(x|\, s_n; e_{a(s_n)}) = \sum_{l=0}^{L} \sum_{\substack{\bar{s}\in\bar{S}[\bar{k}](s)\\ b_R(l,\bar{s})\leq i,\\ b_L(l+1,\bar{s})\geq i+1}} f(i_1,a_1)\cdot f(i_2,a_2)\cdot\ldots\cdot f(i_{N(\bar{s})},a_{N(\bar{s})}) = \qquad (15)$$

$$= \sum_{l=0}^{L} \left[ \sum_{\substack{\bar{s}\in\bar{S}[\bar{k}](s)\\ b_R(l,\bar{s})\leq i,\\ b_L(l+1,\bar{s})\geq i+1}} f(i_1,a_1)\cdot\ldots\cdot f(i,a) \right] \times \left[ \sum_{\substack{\bar{s}\in\bar{S}[\bar{k}](s)\\ b_R(l,\bar{s})\leq i,\\ b_L(l+1,\bar{s})\geq i+1}} f(i,a)\cdot\ldots\cdot f(i_{N(\bar{s})},a_{N(\bar{s})}) \right] \times \frac{1}{f(i,a)}.$$

Again, we will denote the first expression in square parenthesis with $F_0(i+1,l)$ and the second one with $B_0(i,l)$:

$$F_0(i,l) = \sum_{\substack{\bar{s}\in\bar{S}[\bar{k}](s):\\ b_R(l,\bar{s})\leq i-1,\\ b_L(l+1,\bar{s})\geq i}} f(i_1,a_1)\cdot f(i_2,a_2)\cdot\ldots\cdot f(i-1,\kappa), \qquad (16)$$

$$B_0(i,l) = \sum_{\substack{\bar{s}\in\bar{S}[\bar{k}](s):\\ b_R(l,\bar{s})\leq i,\\ b_L(l+1,\bar{s})\geq i+1}} f(i,k_l)\cdot\ldots\cdot f(i_{N(\bar{s})},a_{N(\bar{s})}), \qquad (17)$$

where for uniformity is put $b_R(0,\bar{s}) = 0,\ b_L(L+1,\bar{s}) = W$.

Let us consider all such segmentations, which contain segment $s = (i,\kappa)$ and it is placed after exactly $l$ significant segments and arbitrary number of insertions. Then $F_0(i+1,l)$ equals to the total penalty for segmentation parts from the beginning to segment $s$ inclusive; $B_0(i,l)$ equals to the total penalty for segmentation parts from segment $s$ to the end inclusive.

Since the last segment (with right border coordinate equal to $W$) of any segmentation could be either $L$-th significant segment or such insertion segment that there are exactly $L$ significant segments between it and the beginning, the total penalty for all segmentations (denominator in formula (9)) equals to

$$Z = \sum_{\bar{s}\in\bar{S}[\bar{k}]} \prod_{n=1}^{N(\bar{s})} p(s_n)\cdot p(x|\, s_n; e_{a(s_n)}) = F_0(W,L) + F_1(W,L). \qquad (18)$$

Similarly, we can express $Z$ via values $B_0$ and $B_1$:

$$Z = B_0(0,0) + B_1(0,0).$$

Values $F_0$ and $F_1$ can be treated as penalties for paths on some graph. Let us consider a graph, which vertices are column coordinates $i = \overline{0,W}$ of image field of view, and edges correspond to segments on this image: for each segment $s = (i,a)$ there is an edge $(i \to i+d(a))$ with name $a$, which we will denote with $\varepsilon(i,i+d(a),a)$. The edges are present for those and only those segments that can be found in segmentations from $\bar{S}[\bar{k}]$. A penalty $f(i,a)$ is assigned to each edge $\varepsilon(i,i+d(a),a)$. Penalty for a path on the graph is a product of edge penalties on this path. An edge will be called *significant* if it corresponds to a significant segment. Then values $F_1(i,l)$ and $F_0(i,l)$ equal to total penalties for all paths on the graph from vertex 0 to vertex $i$ that contain $l$ significant edges and end up with significant and non-significant edge, respectively. Penalty for all segmentations $Z$ equals to the total penalty for all paths from vertex 0 to vertex $W$ that contain exactly $L$ significant edges.

In the similar way, values $B_0$ and $B_1$ can be treated as penalties for paths to the ending vertex $W$ from other vertices: $B_1(i,l)$ is a total penalty for all paths from $i$ to $W$ that contain $L-l+1$ significant edges and starts with a significant edge $\varepsilon(i, i+d(k_l), k_l)$; $B_0(i,l)$ is a total penalty for all paths from $i$ to $W$ that contain $L-l$ significant edges and start with non-significant edge $\varepsilon(i, i+1, \kappa)$.

Now, we will propose equivalent recursive definition of values $F_0$ and $F_1$ that determines an effective algorithm of their calculation.

Let $i$ take all possible values of image column coordinates including virtual column with coordinate $W$: $i = \overline{0, W}$, and let $l$ take all possible ordinal numbers of text line $\bar{k}$ character including the "empty-line" character with number 0: $l = \overline{0, L}$. Then values $F_0(i,l)$ and $F_1(i,l)$ can be defined by following recursive equations:

$$\begin{cases} F_0(0,0) = 1, \\ F_1(0,0) = 0, \\ F_0(i,l) = f(i-1,\kappa) \cdot (F_0(i-1,l) + F_1(i-1,l)), \quad l = \overline{0,L}, i = \overline{1,W}, \\ F_1(i,l) = f(i-d(k_l),k_l) \cdot (F_0(i-d(k_l),l-1) + F_1(i-d(k_l),l-1)), \quad l = \overline{1,L}, i = \overline{d(k_l),W}. \end{cases} \tag{19}$$

Obviously, values $F_1$ and $F_0$, defined in such way, coincide with values defined by (13) and (16).

Similarly, let us give recursive definition of values $B_0(i,l)$ and $B_1(i,l)$:

$$\begin{cases} B_0(W,L) = 1, \\ B_1(W,L) = 0, \\ B_0(i,l) = f(i,\kappa) \cdot (B_0(i+1,l) + B_1(i+1,l)), \quad l = \overline{0,L},\ i = \overline{0,W-1}, \\ B_1(i,l) = f(i,k_l) \cdot (B_0(i+d(k_l),l+1) + B_1(i+d(k_l),l+1)), \quad l = \overline{0,L-1},\ i = \overline{0,W-d(k_l)}. \end{cases} \tag{20}$$

Values $B_1$ and $B_0$, defined by (20), also coincide with values, defined by (14) and (17).

Expressions (9), (12) and (15) result in the final expression for the a posteriori probability $\alpha(x,s)$ of an arbitrary segment $s = (i,a)$:

$$\alpha(x,s) = \begin{cases} \frac{1}{Z} \sum\limits_{\substack{1 \le l \le L: \\ k_l = a}} \frac{F_1(i+d(a),l) \cdot B_1(i,l)}{f(i,a)}, & \text{if } a \in A_0; \\ \frac{1}{Z} \sum\limits_{l=0}^{L} \frac{F_0(i+1,l) \cdot B_0(i,l)}{f(i,a)}, & \text{if } a = \kappa, \end{cases} \tag{21}$$

where $F_1$ and $F_0$ are calculated according to (13) and (16), $B_1$ and $B_0$ are calculated according to (14) and (17), and $Z$ is calculated according to (18).

## 2.4 The algorithm of recognition problem learning

Now methods for the computing of all values necessary for algorithm's effective implementation are described and we will formulate it again as a whole. But first we will point out initial values of symbol templates $E^0$, a priori segment probabilities $p^0(\bar{s})$, $\bar{s} \in \bar{S}$ that are set on the first step of algorithm, and the algorithm's stopping criteria. Their variations and their influence on results are described in a section dedicated to experiments.

**Algorithm 1** Recognition problem learning

```
1: Define initial values of parameters $(E^0, p^0(\bar{s}))$.
2: Using obtained parameters $(E^r, p^r(\bar{s}))$ as a ground truth, calculate values $F_0$, $F_1$ and $B_0$, $B_1$ according to (19) and (20).
3: Using $F_0$, $F_1$ and $B_0$, $B_1$, calculate $\alpha^r(x^m, s)$ according to (21).
4: According to (10) and (11), estimate new parameter values $(E^{r+1}, p^{r+1}(s))$.
5: If stopping criteria is fulfilled, stop. Otherwise, go to step 2.
```

# 3 Experiments

## 3.1 Test samples

Experiments were carried out in the following way: the input to the algorithm was a learning sample that consisted of text line images and corresponding text lines. Widths of symbol templates and deviation $\sigma^2$ of the Gaussian distribution (1) were considered known. Experimental samples were divided into two parts: the learning one and the test one. The first part was used for estimation of symbol templates as it was described in the paper, the second part served for quality testing of recognition using constructed templates.

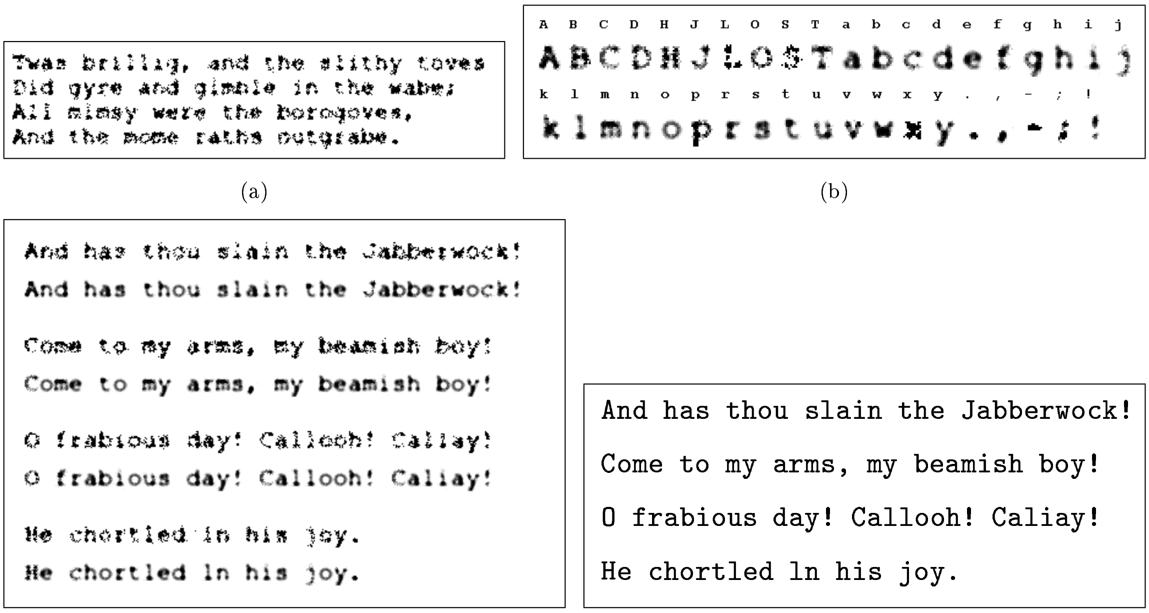
In this subsection we will consider four different samples. The first one was created artificially, others are natural with different types of image degradation.

### 3.1.1 "Jabberwocky" sample

In this example images were generated artificially using document image degradation model described in [2]. This model represents degradation of black and white text images after numerous printing and copying operations. Degradation of such kind fits rather well the statistic model of this paper. In a fig. 1 input images and algorithm work results are presented. Type of degradation can be seen in samples of an input image (fig. 1(a)). Fig. 1(b) contains symbol templates, which were estimated during the learning. Image pairs in fig. 1(c) demonstrate text recognition results. In each pair the first line is an initial image from the test sample, and the second one is a line constructed as a concatenation of templates of recognized symbols. At last, recognized text is shown on fig. 1(d).

Recognition error for the test sample, which contained 240 characters, was about 1%. In fact, there were 2 mistakes in recognition results and in both cases characters "l" and "i" were mischanged. Templates of these characters are very similar and noise easily makes wrong character more probable.

The learning sample contained about 600 characters and was recognized without errors.


Twas brillig, and the slithy toves
Did gyre and gimble in the wabe;
All mimsy were the borogoves,
And the mome raths outgrabe.
A B C D H J L O S T a b c d e f g h i j
k l m n o p r s t u v w x y . , - ; !
And has thou slain the Jabberwock!
Come to my arms, my beamish boy!
O frabious day! Callooh! Caliay!
He chortled ln his joy.

(a) (b) (c) (d)

Figure 1: "Jabberwocky" sample. (a) — input image example, (b) — constructed templates, (c), (d) — recognition results

### 3.1.2 "Book Intro" sample

Another example contains scanned images of a book page. Results of algorithm's work are presented in fig. 2. Both the learning and test samples contained about 550 characters. Test sample recognition error made 0.8%. Typical error in this example was mischange of cyrillic characters "п", "н", "и", which again have similar templates.

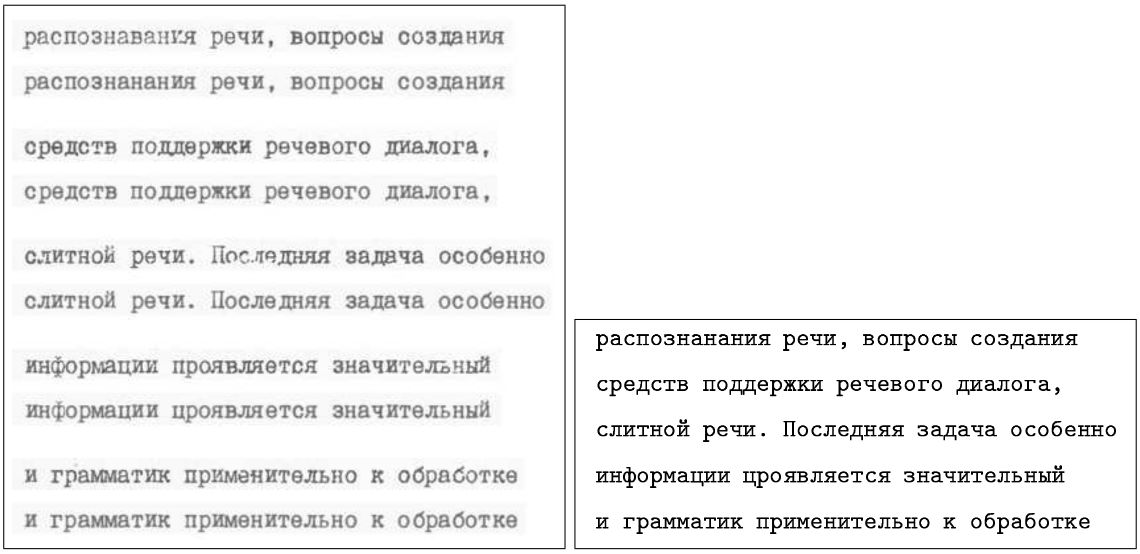
During recognition of the learning sample one error was made (0.8%), which was caused by significant local noise on the image.

### 3.1.3 "Gothic" sample

Next example is a real image with rather strong degradation, in fact, it is not easy to read the text on the images. The test sample contained 550 characters. Results of its recognition are presented in fig. 3. There was 3% recognition error, however, about the half of errors appeared due to lack of necessary characters in the learning sample. Recognition of the learning sample, which was 350 characters long, resulted in 2% error rate.

### 3.1.4 "BigBrother" sample

The last example was obtained by scanning of text, which was previously printed on defected laser printer. As it can be seen from fig. 4(a), type of noise considerably differs from the assumed model, mostly in a non-uniformity of background brightness. However, by manipulating the deviation $\sigma^2$ recognition error rate of 5% was reached for the test sample, which was 650 characters long. In this example the learning sample contained 1250 characters and was recognized with 5.3% of errors. The fact that the test sample



распознаванкя речи, вопросы создания
распознанания речи, вопросы создания

средств поддержки речевого диалога,
средств поддержки речевого диалога,

слитной речи. Последняя задача особенно
слитной речи. Последняя задача особенно

информации проявляется значительный
информации цроявляется значительный

и грамматик применительно к обработке
и грамматик применительно к обработке

(a)

распознанания речи, вопросы создания
средств поддержки речевого диалога,
слитной речи. Последняя задача особенно
информации цроявляется значительный
и грамматик применительно к обработке

(b)

Figure 2: "Book Intro" sample. Recognition results

was recognized with less errors can be explained with selection of less noisy lines into the test sample.

### 3.2 Initial values of algorithm parameters

On the first step of algorithm initial values of symbol templates and segment a priori probabilities are to be set. Moreover, the algorithm requires widths of all templates along with deviation of the Gaussian distribution (1).

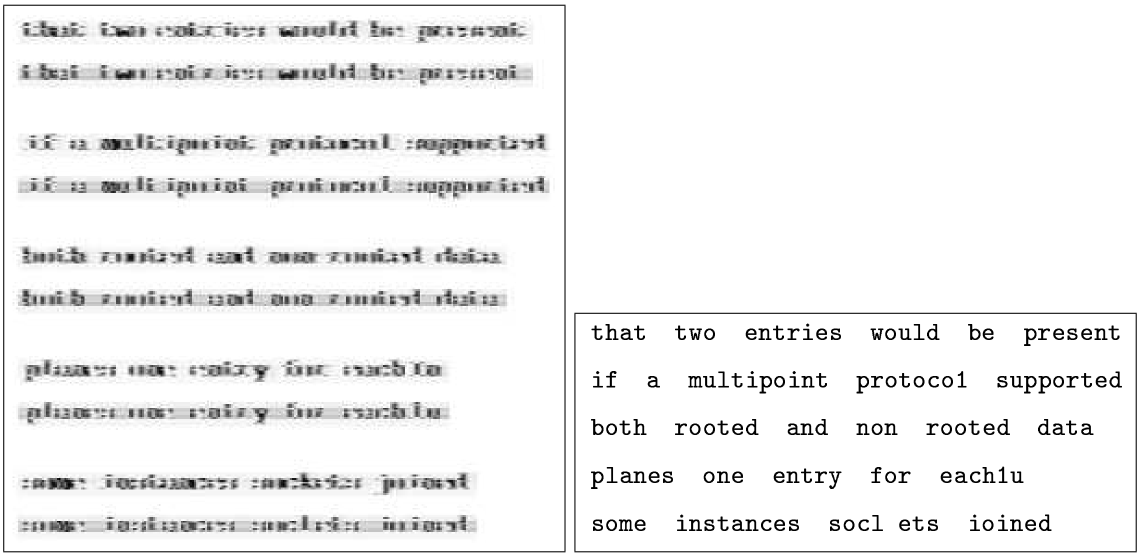
The choice of initial values has some influence on algorithm's output. For instance, setting initial character templates to solid foreground color and insertion template to solid background color results in more confident algorithm functioning, namely the algorithm converges faster and builds templates of insignificantly better visual quality. However, setting of templates in that very way is not obligatory. For example, randomly generated templates are also fully acceptable.

An influence of initial segment a priori probabilities $p^0(s)$ is not so obvious. For instance, setting probabilities of some segments to zero forbid the use of these segments. But, in general, the most suitable initial values of a priori probabilities are equal values for all segments.

In considered experiments initial values $(E^0, p^0(\bar{s}))$ were set as follows: probabilities were equal for all segments, character templates were solid black and insertion template was solid white. Such templates are natural for dark text on light background.

Besides described initial templates, experiments with randomly generated templates were carried out. Difference from black and white templates lies mainly in an increase of iteration number needed to obtain the same result and is only dimly visible when comparing qualitatively learning and recognition results.

Deviation $\sigma^2$ is a characteristics of an input image degradation. An increase of this parameter results in diminishing of a difference between penalties for different characters on one rectangular fragment. A decrease leads to an increase of influence of a noise non-uniformity. Optimal deviation value depends on an input image, but for experimental images these optimal values differed slightly, namely the same value



```
that  two  entries  would  be  present
if  a  multipoint  protoco1  supported
both  rooted  and  non  rooted  data
planes  one  entry  for  each1u
some  instances  socl ets  ioined
```

(a) (b)

Figure 3: "Gothic" sample. Recognition results

was satisfactory for all of them.

Experiments showed that small deviations of parameter values from empirically optimal ones do not result in significant influence of learning results. Instead, the algorithm is very sensitive to fluctuations of character template widths. Setting them to values, that are larger than correct ones, can considerably worsen learning and recognition results. This can be explained with particularities in recognition problem formulation, where overlapping of neighbour templates is forbidden.

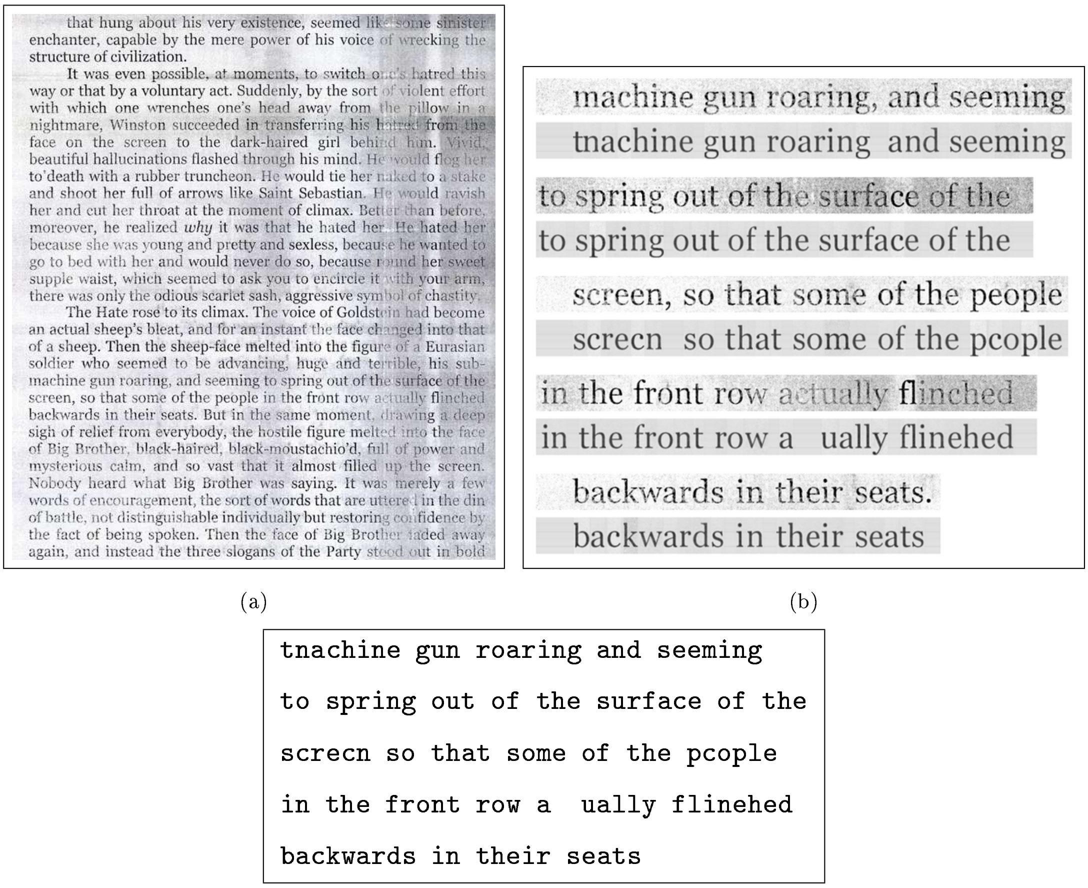
### 3.3 Learning algorithm stopping criteria

An obvious criterion of algorithm stopping is a correct recognition of all images in the learning sample. But the learning problem in this paper is formulated as a search of such parameters that maximize a likelihood function. Such a formulation does not guarantee convergence to the set of parameters that ensures correct learning sample recognition. So this criterion, at least in its pure form, cannot be used.

The simplest stopping criterion is reaching of a certain iteration count. But such criterion is quantitative and does not guarantee obtaining of some qualitative result. An example of another simple criterion could be a decrease of relative difference of likelihood function to some threshold.

Experiments showed that the algorithm converges rather quickly. In general, about 3 to 5 iterations are needed to obtain result that cannot be improved, in the sense that likelihood function increases slightly from iteration to iteration. However, obtained result could happen to be only local maximum of the likelihood function, not the global one. That leads to incorrectly built templates of some symbols and, as a result, in large error rate of the learning sample recognition. In such cases the algorithm can be thrown from this local maximum by a reinitialization of incorrectly built templates and segment a priori probabilities.

A heuristic algorithm of parameter reinitialization was developed on the basis of these observations.



that hung about his very existence, seemed like some sinister enchanter, capable by the mere power of his voice of wrecking the structure of civilization.

It was even possible, at moments, to switch one's hatred this way or that by a voluntary act. Suddenly, by the sort of violent effort with which one wrenches one's head away from the pillow in a nightmare, Winston succeeded in transferring his hatred from the face on the screen to the dark-haired girl behind him. Vivid, beautiful hallucinations flashed through his mind. He would flog her to'death with a rubber truncheon. He would tie her naked to a stake and shoot her full of arrows like Saint Sebastian. He would ravish her and cut her throat at the moment of climax. Better than before, moreover, he realized *why* it was that he hated her. He hated her because she was young and pretty and sexless, because he wanted to go to bed with her and would never do so, because round her sweet supple waist, which seemed to ask you to encircle it with your arm, there was only the odious scarlet sash, aggressive symbol of chastity.

The Hate rose to its climax. The voice of Goldstein had become an actual sheep's bleat, and for an instant the face changed into that of a sheep. Then the sheep-face melted into the figure of a Eurasian soldier who seemed to be advancing, huge and terrible, his sub-machine gun roaring, and seeming to spring out of the surface of the screen, so that some of the people in the front row actually flinched backwards in their seats. But in the same moment, drawing a deep sigh of relief from everybody, the hostile figure melted into the face of Big Brother, black-haired, black-moustachio'd, full of power and mysterious calm, and so vast that it almost filled up the screen. Nobody heard what Big Brother was saying. It was merely a few words of encouragement, the sort of words that are uttered in the din of battle, not distinguishable individually but restoring confidence by the fact of being spoken. Then the face of Big Brother faded away again, and instead the three slogans of the Party stood out in bold

(a)

machine gun roaring, and seeming
tnachine gun roaring  and seeming
to spring out of the surface of the
to spring out of the surface of the
screen, so that some of the people
screcn  so that some of the pcople
in the front row actually flinched
in the front row a   ually flinehed
backwards in their seats.
backwards in their seats

(b)

```
tnachine gun roaring and seeming
to spring out of the surface of the
screcn so that some of the pcople
in the front row a  ually flinehed
backwards in their seats
```

(c)

Figure 4: "Big Brother" sample. (a) — input image, (b), (c) — recognition results

It consists in the following. On its each step the learning algorithm makes several iterations and after that the learning sample is recognized. Recognized text lines are compared with ground truth lines from the sample, and templates of characters, which were not guessed correctly by the algorithm, are declared "guilty" and are reset to initial values. After that the new step begins. Algorithm works until all learning images are recognized correctly or given iteration count is reached.

### 3.4 Algorithm usage as an auxiliary for parameter tuning algorithms

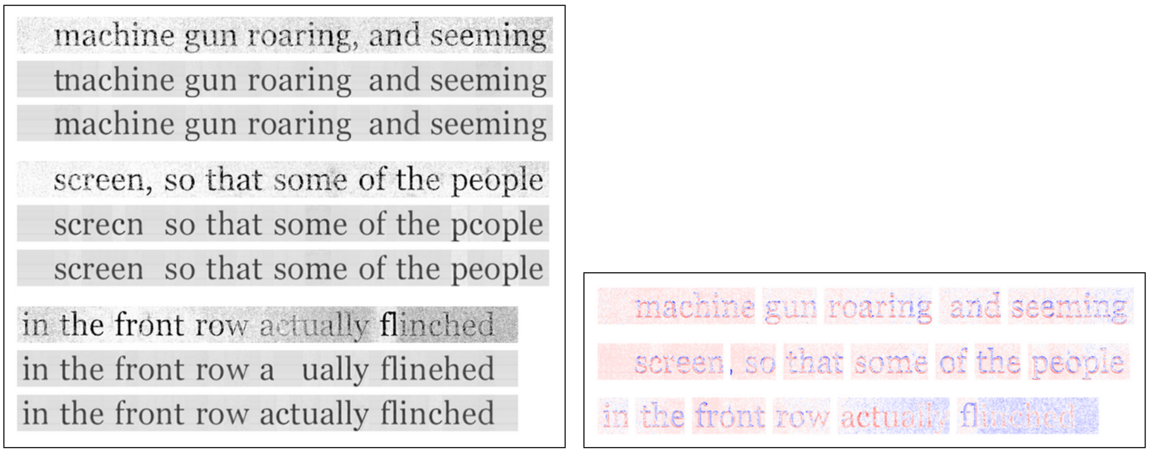
An approach described in the paper can be used not only as an independent method of template statistical estimation but also as an auxiliary method for other learning algorithms, for example tuning (see [3]).

Tuning algorithms depend in lesser way, comparing to described here, on image degradation type, therefore they give better results for images with considerably different noise model.

A disadvantage of tuning algorithms is a requirement of exact segmentations for all images from the learning sample. As it was repeatedly stated, construction of such segmentations usually requires significant operator efforts and time.

However, on the basis of described approach even for strongly degraded images we can obtain segmentations of enough precision. Segmentation construction consists of two steps: at the first one, using input images and text lines as a learning sample, templates are estimated as it is described in the paper. At the second step, using estimated templates, we search for the most probable segmentations among those that correspond to input text lines from the first step. These segmentations along with input images are used as an input to the tuning algorithm.

Below an example of such combined template tuning usage is shown, based on the ”Big Brother” sample. Recognition results are presented on fig. 5(a). The first line of each group is an input image, the second one is an ordinary recognition result, the third one is the most probable segmentation among those that correspond to a text line from the learning sample. Fig. 5(b) shows pixel-wise differences of the first and the third image from each group. Precision of constructed segmentations can be examined in the following figure.


machine gun roaring, and seeming
tnachine gun roaring and seeming
machine gun roaring and seeming
screen, so that some of the people
screcn so that some of the pcople
screen so that some of the people
in the front row actually flinched
in the front row a ually flinehed
in the front row actually flinched
machine gun roaring and seeming
screen, so that some of the people
in the front row actuall flinched

(a)

(b)

Figure 5: Automatic construction of precise segmentations. (a) — results of ordinary and modified recognition, (b) — the same results displayed as a pixel-wise subtraction of the first and the third lines form each group of previous image

# Conclusion

An approach to character templates estimation for text recognition problems, proposed in the paper, allows to significantly reduce amounts of manual work during preparation of learning samples, because instead of exact segmentations of learning images into separate characters algorithm’s input consists only of text lines that correspond to these images.

Natural direction of following research in this area is such a modification of a learning algorithm that allows to automatically estimate not only colors (shades of gray) of character templates but also sizes of these templates.

# References

[1] Michail I. Schlesinger and Václav Hlaváč. *Ten lectures on statistical and structural pattern recognition.* Kluwer Academic Publishers, Dordrecht/Boston/London, 2002.

[2] Prateek Sarkar, Henry S. Baird, and Xiaohu Zhang. Training on severely degraded text-line images. In *IAPR 7th International Conference on Document Analysis and Recognition (ICDAR03)*, pages 38–43, Edinburgh, Scotland, August 2003.

[3] Bogdan Savchynskyy and Olexander Kamotsky. Character templates learning for textual images recognition as an example of learning in structural recognition. In *Second International Conference on Document Image Analysis for Libraries (DIAL'06)*, pages 88–95, Lyon, France, April 2006.