# CHARACTER SHAPE PRECLASSIFICATION IN MIXED SCRIPT OCR FOR MACEDONIAN LANGUAGE

Dejan Đorđević[1], Ljupčo Josifovski[2], Dragan Mihajlov[1]
1 Faculty of Electrical Engineering, Sts. Kiril & Metodij University
2 Faculty of Mechanical Engineering, Sts. Kiril & Metodij University
Karpoš II b.b., 91000 Skopje, Macedonia
{dejan,dragan}@cerera.etf.ukim.edu.mk, ljupco@ereb.mf.ukim.edu.mk

**Abstract:** *Despite of the presence of many commercial OCR programs on the market, the recognition accuracy is still a difficult problem, especially when handling multilingual documents. In absence of an OCR program which can concurrently recognize mixed script documents, we have developed an OCR program for recognizing mainly Macedonian text with words in Latin script. In this paper an adaptation of preclassification based on character shape, for Macedonian characters along with standard Latin characters is presented. Preclassification is performed using the line parameters such as baseline and upper-baseline, as well as the dimensions of the character boxes and their horizontal position within the text line.*

**Keywords:** OCR, character preclassification, mixed script text recognition

## 1. Introduction

Although speech is a sign system that is more natural than writing, writing is considered to have made possible much of the culture and the civilization. Different writing systems, or scripts, represent linguistic units, words, syllables and phonemes, at different structural levels. Some languages ignoring minor differences in orthography share the Latin script, and some also ignoring minor differences in orthography share the Cyrillic script. Each script has its own set of icons, known as characters or letters, that have certain basic shapes. In addition to linguistic symbols, each script has a representation for numerals, and some special symbols.

The task of text recognition is to transform the image data containing textual information into its "usual" representation as a strings over a given alphabet (e.g. ASCII). Today's commercial OCR systems reach highly reliable results on good quality documents. However, in the case of multiple scripts, current devices are far from recognition accuracies demanded for subsequent automatic text analysis. Systems that recognizes characters from a scanned image can be divided into three main operational steps: document layout analysis, character recognition, and contextual postprocessing.

Document layout analysis consists of image filtering and thresholding, skew detection and possibly correction, and character segmentation. In most existing OCR systems, character recognition is performed on individual characters. Therefore, prior to character recognition it is necessary to isolate individual characters from the text image. The technique which partitions the image of lines and words in individual characters is called character segmentation.

Approaches for segmenting document image components can be divided in top-down and bottom-up approaches. Top-down techniques divide the region into major regions which are further divided into subregions, while bottom-up methods first extract small elements (typically connected components) and then combine them to form larger areas (Saitoh *et al.*, 1994). There are also some hybrid methods that combine global and local strategies.

Most of the segmentation and recognition algorithms are extremely sensitive on even a small skew, so the image is usually rotated to eliminate the skew prior to recognition. Number of techniques for text skew angle detection have been reported (Baird, 1987).

The task of the character recognition is to assign the digitized image of the character into its symbolic class. A pattern recognition algorithm is used to extract shape into the appropriate class. Number of different methods like: template matching, statistical, syntactical, and structural approaches exist for character recognition. Artificial neural networks have emerged as the perfect choice for implementing classifiers for OCR.

In the contextual postprocessing step a variety of different approaches exist, from simple dictionary lookup on a word level to implementation of linguistic syntax knowledge.

## 2. Layout Analysis

The documents which have to be recognized are scanned at a resolution of 300 dpi, producing a bilevel TIFF file. As we use the top-down approach, we have to eliminate the skew prior to segmentation. Our approach in layout analysis can be divided in four major steps:

- image size reduction
- detecting the skew angle
- rotating the image to correct the skew
- segmentation

Although high resolution is needed for character recognition, it is not necessary for skew detection. Thus, we first reduce the image. An automatic skew detection is implemented using Hough transform approach on the reduced image. The factor of reduction is calculated according to the original image size. Searching for the global maximum in the Hough transform domain, we obtain the dominant skew angle up to accuracy of 0.25°, allowing angles from –5° to +5°. After obtaining the skew angle the whole image is rotated around the center for skew elimination.

The first step in determining the layout structure of the document is the block segmentation. We use an adaptation of the recursive X-Y cuts technique (Nagy *et al.*, 1985) to decompose the document image into a set of blocks. At each step of the recursive process, the projection profile is computed along both horizontal and vertical directions. Then subdivision along the two directions is accomplished by making cuts corresponding to deep valleys, with width larger then a predefined threshold in the projection profile. A "local" peak detector is applied to horizontal and vertical "profiles" to detect local peaks at which the cuts are placed. Based on the assumption that printed documents are primarily made up of rectangular blocks, the image is recursively cut into rectangular blocks. The depth of the recursion of the recursive X-Y cuts procedure is controlled not to allow further dividing of blocks whose height or width is less then a predefined threshold.

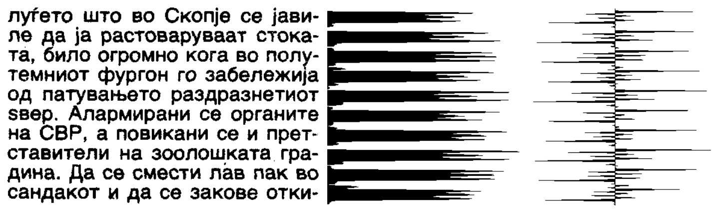


Figure 1. Horizontal profile of a text image and its first difference

After the block segmentation, a textline segmentation is performed. Horizontal projection profile is used for textline segmentation (Fig. 1). Positions of local minima in the projection profile are used as places where cuts are placed. The average line height is calculated from the projection profile, and used to correct the lines whose height is too small or too big. The small lines are being joined to the next or previous line, when joining results in line which height is closer to the average line height. Also, an attempt to resegment very high lines is made. Similarly, location of the words in every textline is performed using the vertical projection profile.

## 3. Character segmentation and preclassification

Character segmentation is fundamental to character recognition approaches which relay on isolated characters. It is a critical step because incorrectly segmented characters are not likely to be correctly recognized. While segmentation of uniformly spaced characters is relatively easy, segmentation of proportionally spaced and low quality texts can be rather difficult. Various character segmentation techniques have been developed to cope with proportionally spaced, kerned, broken and touching characters (Lu, 1995).

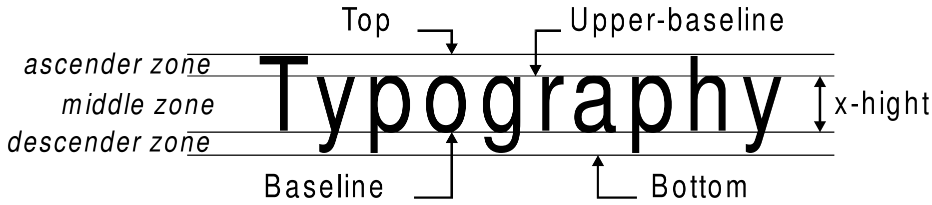


Figure 2: A text image showing the text line parameter positions: Top, Upper-baseline, Baseline and Bottom

In our approach, character segmentation is done on a word level, using some general information like line height, and estimated x-height of the characters being segmented. Four horizontal lines define the boundaries of three significant zones on each textline (Fig. 2). However, in alphabets used by some languages, certain letters even higher than the top line appear. That is the case for some upper case Croatian letters (Č, Š, Ž, Ć), German letters (Ö, Ü), Macedonian letters (Ѓ, Ќ), etc. Such letters can be decomposed in four, instead of three zones used in the standard approach (Depowska *et al.*, 1995).

Prior to character segmentation, the positions of the baselines are found. The local maxima in the first difference of the horizontal projection profile is considered to be the position of the upper-baseline, while the local minima the position of the baseline (Fig. 1). This procedure is extremely sensitive on skew, and may fail in longer textlines with a skew larger than 0.5°.

Character segmentation is performed using the vertical projection profile and the break cost function. After the segmentation, every character box is assigned to one of the character shape classes. Suspicious character boxes which are too wide or too thin for their class are reexamined. Thin character boxes are considered broken characters, and an attempt was made to join them to neighboring character boxes. Wide character boxes are resegmented using the top and bottom profiles, also considering minimum and maximum character box width for particular classes. After correcting character boxes, positions of the baselines are recalculated, only this time on a word level and using character box positions and dimensions which yield more accurate results. Characters are then reclassified using the baseline positions and character box aspect ratio.

The task of the preclassification is to prune the set of classes to a smaller number. Assigning the character into a shape class yields to: better performance, distinguishing capital letters for the characters that has similar shapes of lowercase and uppercase letters, and simplifying the recognition engine.

Most of the characters in the Latin and Cyrillic alphabet can be readily mapped from their positions relative to the baselines to a small number of classes shown in Table 1. Since all character preclassification is based on character box vertical location, these techniques tend to be largely font independent, but there are some characters that can be classified in more than one class, depending of the font (e.g. Q, Q).

Table 1:

| Class Code | Spread through | | | Character |
|---|---|---|---|---|
| | asc. | mid. | desc. | |
| **A** | ✓ | ✓ | | A-ZĐbdfhikltđćčžš 0-9%#$&/+?! А-ГЃ-ХЧШбѓќ |
| **Q** | ✓ | ✓ | ✓ | Qj ( ) ДЦЏфј |
| **x** | | ✓ | | acemnorsuvwxz - =.: авге-ик-пстхчш |
| **y** | | ✓ | ✓ | gpqy ,; друцџ |
| **`** | ✓ | | | " ' ` |

Considering the aspect ratio of the characters this classes can be further divided. In our approach we use 10 character shape classes shown in Table 2. The preclassification process is implemented as a decision tree. An outline of the decision process is given on Fig. 3. The top and the bottom of the character box are labeled with Ctop and Cbottom, the upper-baseline with ubl, the baseline with bl, the width of the character box with Cwidth, and cc denotes the character class.

$$\mathtt{asc} = \frac{\mathtt{Ctop} - \mathtt{ubl}}{\mathtt{ubl} - \mathtt{bl}}$$

$$\mathtt{desc} = \frac{\mathtt{bl} - \mathtt{Cbottom}}{\mathtt{ubl} - \mathtt{bl}}$$

$$\mathtt{asp} = \frac{\mathtt{Ctop} - \mathtt{Cbottom}}{\mathtt{Cwidth}}$$

Isolated character images are scaled to binary square matrix of constant dimensions (16×16) which are forwarded to appropriate classifier according to their class membership.

```
if(asc>0.15) {
  if(desc>0.15) {
    if(asp>3)          cc='j';
    else               cc='Q';
  }
  else if(desc>-0.2) {
    if(asp>3)          cc='!';
    else               cc='A';
  }
  else if(desc<-0.5)   cc='`';
}
else {
  if(asc<-0.6)         cc='.';
  else if(desc>0.2)    cc='y';
    else if(asc<-0.15 && desc<-0.15
           && asp<0.8) cc='-';
      else if(asp>2.5) cc=':';
        else           cc='x';
}
```

Figure 3: Deciding the character shape class

## 4. Character Recognition

Character recognition is a process in which the isolated character image is being assigned to a class representing a letter. In our approach we use an adaptive logic network (ALN) as a classifier (Armstrong, 1975). The binary matrix (16×16) holding the scaled image of the sample to be recognized is used as a feature vector. Our implementation of ALN uses one or more trees per output class (Đorđević *et al.*, 1995). Every tree is trained to recognize a single class, i.e. to answer whether the sample belongs to its class or not. Several trees (usually an odd number) are provided for every class and they vote for decision with the trees of the other classes. The maximum selector decides in favor of the character with largest number of votes. We have tried networks with 512 to 2048 leaves and realizations of 1 to 9 voters.

## 5. Postprocessing and script determination

Latin and Cyrillic alphabet share some symbols among each other. Some of this symbols represent the same letter (e.g. AEOMTj), while some other represent different letters (e.g. Cyrillic B represents Latin V). In our approach we assume that a word can be consisted of letters from only one script. In order to correctly recognize a word, one must distinguish how to recognize the shared symbols: as a Cyrillic or as a Latin letters. Therefore, characters in every class are separated in 5 categories:

1. symbols that represent the same letter in both alphabets labelled as **C&L**;
2. symbols that represent different letters in both alphabets labelled as **C|L**;
3. symbols specific for Latin script labelled as **Lo**;
4. symbols specific for Macedonian Cyrillic script labelled as **Co**;
5. symbols representing digits and special marks labelled as **S**;

The final classes with their categories are given in Table 2.

Table 2:

| Class | Categ. | Characters |
|---|---|---|
| **A** | **C&L** | AEJKMOT |
| | **C\|L** | BSHPCX |
| | **Co** | БГЃЖЗИЛЉЊП ЌУФЧШ бѓќ éйô |
| | **Lo** | DFGLNQRUVWYZ ĐČŠŽĆ bdhkfiltđćčžš |
| | **S** | 0123456789 ? % / $ # & |
| **Q** | **Co** | ДФЦЏф |
| | **Lo** | Q |
| **y** | **C\|L** | py |
| | **Co** | дцџ |
| | **Lo** | gq |

| Class | Categ. | Characters |
|---|---|---|
| **x** | **C&L** | aeко |
| | **C\|L** | sxc |
| | **Co** | вгжзилљмнњптчш |
| | **Lo** | nmruvwz |
| | **S** | + ? |
| **j** | **C&L** | j |
| | **S** | ( ) / |
| **!** | **Lo** | Iilft |
| | **S** | 1 ! / |
| **.** | **S** | . , _ |
| **'** | **S** | " ' ` o ‘ ’ “ ” |
| **-** | **S** | - = ~ |
| **:** | **S** | : ; ! |

After the recognition process all the characters in the word are examined for their category membership, and decision is made how to treat the word: as written in Cyrillic or in Latin script. It is obvious that if some of the letters in the word are unique for the first script, and there are no letters unique for the second script, that word should be recognized as written in the first script, and vice versa (e.g. Beca**u**se, рас**т**ура). If the word consists only of characters whose symbols are present in both scripts and represent same letters, it can be recognized either way. For the words that consist only of characters whose symbols are present in both scripts but some of them represent different letters in the Latin and the Cyrillic script, it can not be distinguished how they should be read (as written in Latin or Cyrillic script). Most of the such words have valid meaning in only one way, and the recognition problem for them can be overcome by a simple dictionary look up (e.g. руксак, sjaj, co, ce). There are some words that have valid meaning if read either way (e.g. KOCKA, read as written in Latin script meaning cube, and read as written in Cyrillic script meaning bone). For this words the ambiguity can not be solved on a word level. Only examining the context on a sentence level can solve such ambiguities. Words in which letters unique for the Latin script exist along with other letters unique for the Cyrillic script, signal an error in recognition, since interword script changing is not allowed. In such cases in our approach the suspicious characters are reexamined and second alternatives are taken into account in attempt to correct the error.

Script determination is implemented as a simple decision tree. For every word the number of characters belonging to categories Co, Lo, and C|L are counted and labeled as Co.n, Lo.n, and C|L.n respectively. The outline of the decision procedure is given on Fig. 4.

```
if(Co.n==0) {
  if(Lo.n==0) {
    if(C|L.n==0)
      Register_either_way;
    else
      Dictionary_lookup;
  }
  else
    Register_as_Latin;
}
else
  if(Lo.n==0)
    Register_as_Cyrillic;
  else
    Reexamine;
```

Figure 4: Word script determination process

## 6. Concluding remarks

In this paper an optical character recognition system that can handle mixed script documents has been presented. A special font independent character preclassification based on character shape for the letters in the Cyrillic and Latin alphabet has been developed. The preclassification error has shown to be very low, after treating the special cases of textlines consisting of only **A** class (titles) or only of **x** class letters (end of a sentence from the previous line).

A method for script determination on a word level has been described. This approach uses the hard decisions obtained from the character recognizer for all the characters in the word to determine the script it has been written in. In the future we plan to use the soft decisions from an appropriate character recognizer in word script determination which we hope will give even better results, especially in cases where a character was misrecognized.

## 7. References


Armstrong, W. and Godbout, G., (1975), "Properties of Binary Trees of Flexible Elements Useful in Pattern Recognition", IEEE Int. Conf. on Cyb. & Soc., San Francisco, pp. 447-449.

Baird, H. S., (1987), "The skew angle of printed documents" Proc. SPSE 40th Conf. Symp. Hybrid Imaging Systems, Rochester, NY, pp. 21-24.

Depowska, D. and Dabrowska, I., (1995), "Modification of the Character Preclassification Method for Four Striped Letters", Proc. 17th Int. Conf. ITI'95, Pula, Croatia, pp. 247-252.

Đorđević, D., Mihajlov, D., and Josifovski, Lj., (1995), "Computer System for Support of Humans With Damaged Sight: Subsystem for Optical Character Recognition of Printed Cyrillic Text", Simpozijum o računarskim naukama i informatici, YU Info'95, Brezovica, Srbija.

Lu, Y., (1995), "Machine Printed Character Segmentation – an Overview", Pattern Recognition, Vol. 28, no. 1, pp. 67.

Nagy, G., Seth, C. S. and Stoddard, S. D., (1988), "Document analysis with an expert system", Proc. Pattern Recognition in Practice II, Amsterdam, pp. 19-21.

Saitoh,T., Yamaai, T. and Tachikawa, M., (1994), "Document Image Segmentation and Layout Analysis", IEICE Trans. on Inf. and Syst., Vol 77, no. 7, pp. 778-888.