

Oversegmentation Methods for Character Segmentation in Off-Line Cursive Handwritten Word Recognition – An Overview

MAGDALENA BRODOWSKA

Faculty of Physics, Astronomy and Applied Computer Science,
Jagiellonian University, Reymonta 4, 30-059 Kraków, Poland

e-mail: *m.dudek@uj.edu.pl*

Abstract. Character segmentation (i.e. splitting the images of handwritten words into pieces corresponding to single letters) is one of the required steps in numerous off-line cursive handwritten word recognition solutions. It is also a very important step, because improperly extracted characters are usually impossible to recognize correctly with currently used methods. The most common method of character segmentation is initial oversegmentation – finding some set of potential splitting points in the graphical representation of the word and then attempting to eliminate the improper ones. This paper contains a list of popular approaches for generating potential splitting points and methods of verifying their correctness.

Keywords: recognition, segmentation, character, handwriting, cursive, overview.

1. Introduction

Off-line cursive handwriting recognition can be employed in a variety of real world tasks, such as: processing handwritten bank cheques, reading addresses from envelopes, analysis of handwritten forms and, generally, in converting scripts like notes, historical documents, letters etc. to a fully digital (searchable, easily navigated and manipulated) form. The broad range of significant real world applications and the considerable difficulty of the task both contribute to the fact that there are many known methods of performing cursive word recognition – one suited better for some applications and some for the others.

Existing methods of cursive handwritten word recognition can be assigned to four major categories based on the usage of character segmentation [1, 2, 3, 27]:

1. Holistic Approach

Systems based on this approach (also called a global approach) try to recognize entire words, without splitting them into single characters. Most popular methods among this group are based on analysis of the number and order of ascenders, descenders, loops and vertical strokes. They often rely on heavy dictionary searching that is costly and prone to be misled by spelling errors. The detailed description and examples can be found in [26].

2. Classical Approach

In this approach character segmentation strictly precedes character classification and hence word recognition. The image of a word is divided into segments that should represent single characters and are passed to next steps. No feedback from those later steps is used.

3. Recognition-Based Segmentation Approaches

Character segmentation and character classification steps are not totally separate. Degree of connection between them vary from a solution to a solution. However, the most common general approach is to split words into segments that should be characters, pass each segment to a classifier and, if the classification results are not satisfactory (e.g. some measure of belief requirements is not met), call segmentation once more with the feedback information about rejecting the previous result.

This and previous (2) groups are often jointly referred to as analytical approaches [26, 29].

4. Mixed Approach

Systems that belong to this group contain elements from at least two groups mentioned above. Some of them use holistic recognizers to lexicon reduction or verification of most probable hypotheses obtained by segmentation base techniques [27]. Other examples include systems based on human reading models [27, 28]. Their recognition phase, in general, can be divided into two parts. In the first one, global features (related to a word shape – e.g. word length, ascenders, descenders and loops) are extracted from the image and holistic recognizer uses them to select a short list of elements with a compatible shape from lexicon. If the analysed word can be identified as one of them with a satisfying degree of confidence, recognition is completed. Otherwise, analytical approaches are employed to extract single characters from the word image.

The variations in writing styles and flaws in individual handwriting skills cause correct performing of the segmentation without classification feedback almost implausible. The most accurate analytical systems belong mainly to the third group above. Most of algorithms constructed in recent years fall to this group as well.

In methods belonging to the second group, the segmentation achieved after the segmentation step is final, and finding it requires finding the correct number of segments. However, in methods from the third group it is not necessary. Moreover – generating more candidate points might be necessary if some of them are rejected by a classifier. Those candidates can be generated as needed – if some previous ones were rejected. Alternatively some pool of potential points might be initially generated. Eventually, only the most probable or plausible points will be selected with the help of a classifier. This selection may take different forms: either implicit (e.g. when using Hidden Markov Models [27]) or explicit. In the latter case the tentative evaluation of the candidate points at the segmentation step is useful to find the order in which the points should be presented to the next step. In all those cases decision on which candidate split points will be finally used is taken together by a segmentation and classification module. The last case is employed to great extent in recent publications.

Main steps of the methods described as the last case above take often the form of:

1. find many possible candidate split points,
2. tentatively evaluate those points,
3. reject those that are very unlikely to be the correct split points and, if necessary, add missing ones,
4. present the candidate points to the classification step in some order taking into account their preliminary evaluation.

This paper presents algorithms based on those steps with emphasis on generating a preliminary pool of candidate split points and verification of their validity.

2. Finding candidate split points

2.1. Methods based on the projection analysis

2.1.1. Methods based on the vertical histogram

A vertical histogram (also called a vertical projection) is a function assigning to each column of an image the number of black pixels found in that column (or just pixels not in background colours in case of not binarized images). Vertical histogram analysis was one of the earliest techniques used to find possible splitting points. This method allows for easy detection of empty columns, which very probably might be white spaces between successive letters. It can be also helpful in detection of vertical

strokes and parts with multiple lines (unfortunately a vertical histogram does not provide enough information to undoubtedly distinguish those two cases).

Early simple methods assumed segmentation points to be in places, where histogram values were below some given threshold. That threshold might be calculated based on a mean line width (pen thickness) or, even simpler, as a percentage of a mean histogram value. Simplicity and low computational complexity (especially in the case, where the pen thickness was not calculated) are unquestionable advantages of this solution, as well, as being able to detect horizontal ligatures between adjacent letters very well. However, it also has several serious shortcomings, such as:

1. dividing characters that contain valleys (i.e. handwritten k, l, m, n, r, u, v, w, y),
2. dividing loops (straight in the middle of the loop is the most common case),
3. inappropriate separation of fragments that are minor in size, but still have major influence on recognition,
4. sensitivity to a threshold value (often generates great numbers of candidate points or – in the opposite case – omits proper splitting points),
5. a single real separating place can appear as many points,
6. unsuitable for slanted cursive writing.

Drawbacks 1 and 2 are not specific to this solution but persist also in many more sophisticated ones. General difficulties with eliminating them are one of the reasons why an oversegmentation approach is applied. Problem from point 5 manifests itself as all columns with single non-vertical strokes being marked as candidate splitting points. It can be eliminated by simply choosing from the group of adjacent candidate splitting points the one with the least histogram value (as illustrates Fig. 1). In case of slanted writing, columns with ligatures are often crossed by strokes from two or sometimes even three characters, which increases their histogram values significantly and makes their recognition as splitting points impossible with this method. The slant correction is then essential preprocessing for all methods based on vertical histogram analysis.

Detection of steep changes of a histogram is a little more sophisticated yet still intuitive method for finding candidate split points. Steep increase of the histogram value means a beginning of the character, while steep decrease, its end. Such a rule would suggest that restricting the method to just one of the cases would be enough. However, this technique leads to omission of many real split points. For example – if the method only detected decreases, it would not detect a split point between a character that ends with a soft histogram value decrease and the next one. Analogical problem would occur if we were detecting only the steep increases of the histogram density, as shows Fig. 2.

Doubling some of the split points is an intrinsic trait of this method. This happens, for example, in the case of empty columns (e.g. without foreground pixels) and most horizontal ligatures between two letters (the latter could also be treated as an advantage, because it allows the ligatures removal and, in consequence, simplifies character recognition). However, this is not nearly as troublesome as producing



Fig. 1. Oversegmentation based on a vertical histogram. (a) An original image. (b) An image histogram. (c) Potential split points: columns with the histogram value less than 34% of a mean value are in light grey, dark grey lines indicate histogram minima

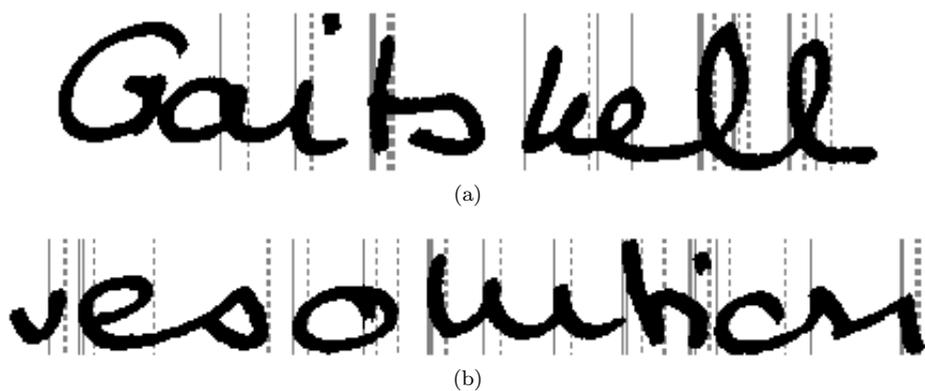


Fig. 2. Oversegmentation based on a vertical histogram. (a), (b) Solid lines mark steep increases of the histogram, dashed – steep decreases

multiple candidate split points per one real split point that occurred in the previous method. With this solution also valleys are split less frequently – only if their edges are very steep, near-vertical lines. Loops are still sometimes split, not in the middle, but near their inner borders (a contour). The main drawback of this method is possibility of cutting small horizontal fragments of letters such as E, F, L, T and separating distinct vertical strokes from the character body. It happens more often than with the previously described method. Despite (still persisting) numerous shortcomings this solution is still applied for preliminary and redundant recognition of split points [4, 5].

Richard G. Casey and Eric Lecolinet in [1] reported also some other methods based on the vertical projections, like finding maxima of the second difference of the projection or analyzing projection of an image after applying an AND operator to its columns, but this methods are applicable mainly to systems processing machine typed words.

2.1.2. Methods based on the angled histogram

Split points lookup based on vertical histogram analysis fails when used on slanted handwriting. Slant of even minor angle causes the methods from point 2.1.1 to not detect many inter-character connections. Of course, slant of the image can be corrected during preprocessing, which is common in OCR systems, also those not using a vertical histogram. Examples of such algorithms can be found in [6, 7]. A slant corrections process is, however, a major change of an input image and thus can alter information contained there and mislead further steps. One of the solutions where additional preprocessing can be avoided while preserving the simplicity of using density analysis, is the angled projection. The angled projection is similar to a vertical histogram – foreground pixels are counted along a line. Those lines are not vertical, but make an angle comparing to an image edge. Friday and Leedham ([8]) proposed a split points detection method where they examined 17 histograms made with different angles (every 2 degrees from -16 to $+16$ from vertical). Yanikoglu and Sandon [9] also used a constant set of angles. They broadened the range of angles resulting in -30 to 40 degrees, while increasing step to 10 degrees. In the case of both methods the cuts were performed along the chosen projection angles. Suggestions on which angle to choose can be found in the article of Frank de Zeeuw [10], where a slant correction method based on the angled projection was introduced. In Zeeuw’s work angled projections for angles between -45 and $+45$ (which is the range containing most of the cases in handwriting) were found. Next, for all of them, the heights of histogram peaks were measured. The one with maximum heights was chosen. The angle step was variable – initial was 5 degrees and decreasing with peaks heights increase. The approach is based on the assumption that a histogram corresponding to a word written straight up should have more pronounced peaks and valleys (i.e. extrema). Example of this can be seen in Fig. 3.

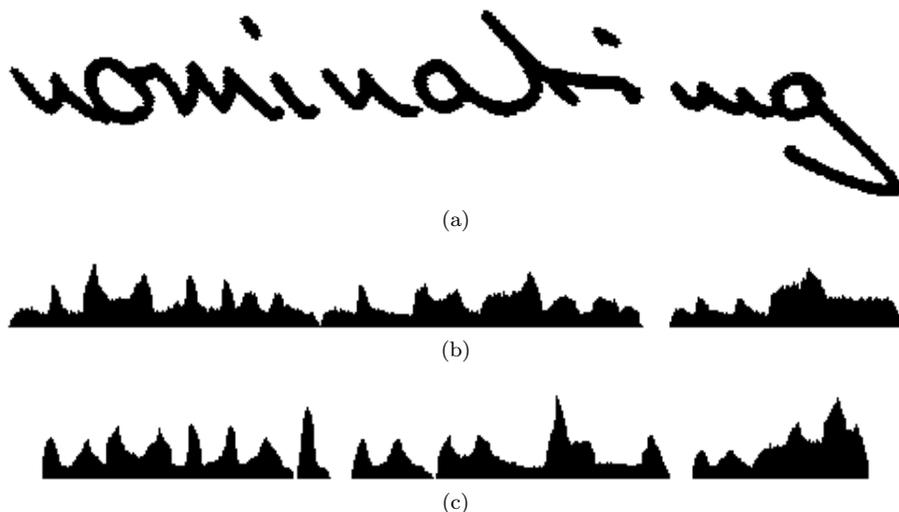


Fig. 3. Projection analysis of slanted words. (a) An original image. (b) A vertical image histogram. (c) An angled image histogram corresponding to the dominant slant

2.2. Methods based on the profile and contour

A profile is a 2D image made from a binarized connected component by leaving only outermost foreground pixels in each column. The sets of points placed at highest coordinates (so called an upper profile) and those placed on lowest coordinates (a lower profile) are examined separately. This feature is very easy to obtain and to analyze in order to determinate candidate split points. The local extrema of each part of the profile are considered as the potential split points. The minima of the upper profile are most intuitive. Splitting the component in these places corresponds to the idea on which "drop falling" algorithms are based. The technique used there is splitting the characters in a place where a "drop" released at the topmost black pixel of the left-side character would "fall". In this way centers of lower convex ligatures, which are very common in cursive handwritten words, can be detected. As well, as starting points of ligatures going up from left to right characters (which is the common shape of connections coming out of a loop). Unfortunately determining cuts depending on upper profile minima leads to cuts in valleys of letters m, n, u, w, v, y, separating the upper short line of r and dissecting of vertical strokes from the body of a character in letters b, d, h, k, p. On the basis of maxima of the lower profile a letter can be easily cut off the ligature that comes from the preceding character and approaches from below (which is also a common case). However, such points occur also in many places where cuts would intersect single letters, such as separating hooks from a, u, d, cutting letters containing arcs pointed up (like in handwritten h, k, n, m) in the middle and splitting the letter w. Lower profile minima are sometimes interchangeable with upper profile minima, as both appear at the centers of lower ligatures. However, the former are considerably more frequent in handwriting. They occur at

the lower ends of vertical or nearly vertical strokes (as in letters f, h, i, k, m, n, p, r, w), where they can sometimes be used for determining a leftmost bound of the letter. Unfortunately they also appear in the centers of the loops and characters with a curved bottom (like a, b, c, d, e, g, k, l, o, p, s, u, w, y) where they are completely spurious. The upper profile maxima are rarely used as they commonly appear in the center parts of letters rather than in connection points. Fig. 4 illustrates the contour based segmentation.

In cursive handwritten 3 or even 4 lines can cross a single column. By examining a profile only connections made by uppermost or lowermost lines can be detected. Any connection that is hidden from the profile by such lines will be missed. That is why extrema of the image contour of the word can be analyzed instead of or additionally to profile extrema. Likewise for the profile, lower and upper contours are distinguished. Together they form a so called external contour. The upper contour is found by tracing the contour on the upper side of the writing strokes from the leftmost to rightmost black pixel. The lower contour – analogically but on the lower side of the line. There are other definitions of the upper and lower contour. In [11] the external contour is traced counterclockwise. Fragments where it runs mostly from right to left are marked as the upper part and those when it runs mostly in the opposite direction are marked as the lower part. Regardless of the chosen definition the split points are looked for in local extrema of contours. The methods rarely make use of all kinds of extrema (with notable exceptions as [11]). Usually only pairs or subsets are used. In numeric strings segmentations (postal codes on envelopes or numbers on bankchecks) a very popular solution is to use so called min-max algorithms, where upper profile minima and lower profile maxima are used. The idea that stands behind them is that in the touching points of two characters the distance of the upper and lower contour should be rather small (preferably equal to the average stroke thickness). This method is less satisfying in the case of overlapping characters or characters with more than one touching point. Nicchiotti and Scagliola ([12]) made use of minima of three contours: the upper, lower and additional, "median" contour, obtained only for columns with 3 black runs. Minima were considered to be correct split points only if it was possible to obtain the appropriate contour in every point of their neighborhoods. Cuts were performed along straight but not necessary vertical lines. Vertical lines were preferred, but if the number of crossed foreground pixels was greater than some limit, which was set based on the width of the core region (a region between the baseline and upperline, that is without ascenders and descenders), other direction angled from -45 to 45 deg. from vertical lines were tested. The angle minimizing the number of intersected pixels was chosen. In [13] only minima of the upper contour have been used. Bozekova ([14]) proposed a method for checking if the authors of the two documents are, in fact, the same person, where upper contour minima placed near lower contour minima (approximately the pen thickness apart or closer) were considered as segmentation points. Mao, Sinha and Mohiuddin designed an address recognition system, in which for finding candidates splitting points in the oversegmentation step horizontal stretches and high curvature points of these contours were used additionally to lower and upper contours minima.

Oliveira in [15] proposed an interesting method of numeral string segmentation in which contour analysis is supported by the skeleton analysis (methods for obtaining

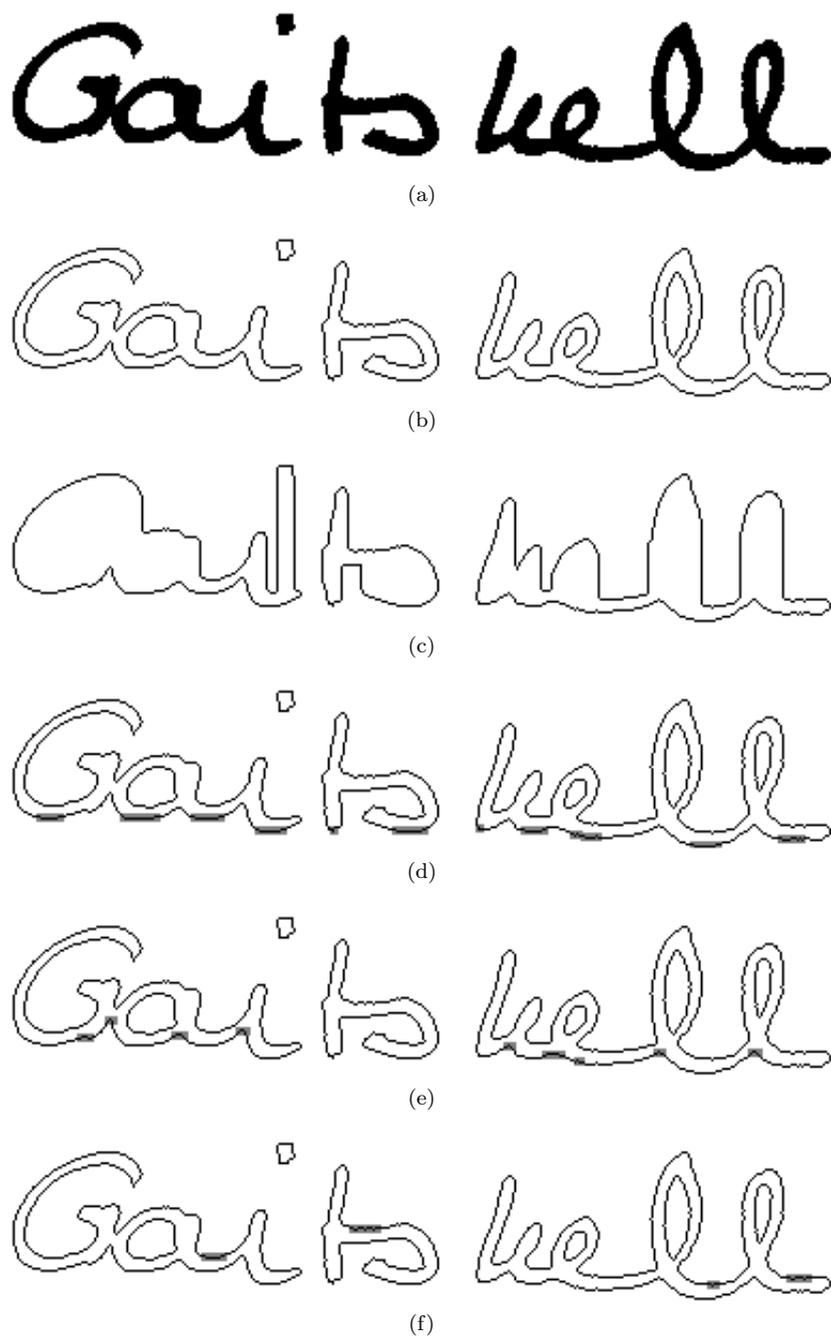


Fig. 4. Oversegmentation based on the word profile and contour. (a) An original image. (b) A word contour. (c) A word profile. (d–f) Potential split points: (d) lower contour minima; (e) lower contour maxima; (f) upper contour minima

a skeleton can be found, among others, in [20, 17]). Apart from profile and contour extrema, also intersection points of the skeleton (defined as black pixels with at least 2 black neighbors on the skeleton) were searched for. Skeleton intersection points are often located near the stroke connections, therefore they can reveal ligatures between characters, also in the case of strongly skewed or overlapped characters, which is more difficult for contour or profile analysis. Moreover, skeleton paths are very useful for determining a line of the cut. Oliveira's method made two kind of cuts: parallel to a skeleton path and orthogonal to it. The decision which to choose was made based on analysis of placing skeleton intersection points relative to profile or contour extrema. Skeleton analysis is highly supportive in classification of handwritten Chinese characters (which is a consequence of the specific shape of that writing). An example of that method used for segmentation can be found in [18].

Oversegmentation methods based on the structural features like the contour, profile and skeleton produce less spurious split points than methods based on the projection analysis and miss less of the true split points. They are also not that vulnerable to the slanted writing. However, neither profile or contour extrema nor intersection points of the skeleton will consider natural empty spaces between disconnected characters as split points – in particular empty columns are not detected, which is certainly a drawback. That is why such methods have to cooperate with connection component analysis. Cut line selection is essential here: it should not be a straight line coming through a found point, as this can easily lead to cutting off fragments of a single character, especially when establishing a segmentation point based on the lower profile or lower contour minima. It is even more important in the case of systems that do not implement a slant correction as a preprocessing step.

2.3. Methods based on the background analysis

Segmentation paths should lead between the characters, that is – in the background, only occasionally intersecting written lines. Because of that, an approach focusing on the image background while finding the splitting paths may be plausible. Methods based on the background can be divided into two groups:

1. based on the background skeleton,
2. based on background regions analysis.



Fig. 5. A word image and its background skeleton

2.3.1. Methods based on the background skeleton

In methods of this group a skeleton of the word background is found first (Fig. 5). A regular thinning algorithm used normally to produce a foreground skeleton can be used here. Because some solutions require a word to have a small empty space at each side (that is – it should not cross the image border), adding a thin background-coloured frame around the word before starting the thinning process can sometimes be necessary. In simplest applications a background skeleton is used for separation of non-touching characters [18, 19]. In contrast with methods based on histogram analysis, they can unambiguously separate characters that, although without crossing strokes, have intersecting bounding boxes. Background skeleton lines form complete segmentation paths, that would not separate fragments of processed characters, which is a considerable advantage especially towards the straight-line cutting paths.

Comprehensive analysis of the background skeleton allows for extracting information useful also when separating touching characters. A system recognizing connected handwritten digit strings described in [20] is one of the examples. In the aforementioned system skeleton fragments were divided into several classes called segments: base segments – lines formed between the foreground and the upper image boundary (called upper-base segments) and between the foreground and the lower image boundary (called lower-base segments), side segments – between the foreground and the left or right boundary (not used in segmentation), hole segments – formed inside background regions enclosed totally within foreground strokes and – most interesting for segmentation purposes – branch segments – i.e. segments starting from the base segments and directed towards the interior of the image (a branch segment is named the upper or the lower branch segment depending on whether it is connected with the upper or the lower base segment). Splitting points were searched for in proximity of certain characteristic points (feature points) of the background skeleton. Such points are: end points (points with only one black neighbour), fork points (points connecting branches, with at least three black neighbours) and corner points (points of abrupt changes of a line direction). Potential cutting lines were made along lines connecting a feature points placed on different segments or vertical lines connecting a feature point located on the upper segment (the upper base or upper branch segment) with the lower base segment or a feature point located on the the lower segment (the lower base or lower branch segment) with the upper base segment. Liang and Shi in [19] used a similar strategy in segmentation of touching Chinese characters. To reduce the cost of searching potential segmentation paths, they restricted the method only to fork and end points (called further background skeleton feature points) of so called major branch segments (i.e. fragments of the background skeleton from a fork point in the upperbase segment or the lowerbase segment to the end point or the last fork point in upperbranch segments or lowerbranch segments). Those segments are in general branches that extend furthest towards the basesegment opposite to the one of their origin. For determining segmentation paths fork points and corner points of the foreground skeleton (called further foreground skeleton feature points) were used as an aid. Two types of segmentation paths were created. First consisted of paths formed by

connecting feature points from the upper major branch with the ones on the corresponding lower major branch. Second contained all paths formed by connecting background skeleton feature points with nearby foreground skeleton feature points. Determining segmentation paths based on both background and foreground skeleton feature points appears also in [18]. However, the foreground skeleton points were the primary ones there, and the background skeleton points – secondary ones.

The main advantage of using the background skeleton in oversegmentation is that the determined complete or at least partial segmentation paths have a useful trait of intersecting only selected lines while avoiding others that could stand in their way. The possibility of cutting off the small fragments of characters does not exist here. Easiness of loop detection (loops are indicated by small isolated fragments of the background skeleton) is another quality of this method. Disregarding such fragments in the process of segmentation point searching will secure the algorithm against unwanted loop segmentation. Slanted strings of touching characters can be segmented as well as the straight ones. A background skeleton feature point of some kind is found near almost every correct segmentation point, so thorough analysis of the skeleton should yield segmentation with only very low percentage of undetected split points. However, like methods described before, this one can also separate hooks or tails from letters like a, e, d, split valleys or cut off the vertically orientated strokes from the character body in less neatly written characters b, d, h, k, p, q, r.

2.3.2. Methods based on the background regions

Background regions are the white areas between two successive black strokes (orientated more or less vertically). They are detected by scanning all image rows from top to bottom and from left to right searching for connected groups of background pixels. It is worthwhile to notice that, in this way, also regions between lines belonging to non-neighbouring characters will be found. Because such regions are irrelevant to segmentation, regions wider than twice a mean character width should not be considered. Also too small regions should not be taken into account, as they can be a result of noise or be created inside lines due to imprecision of a writing tool. For handwriting recognition purposes, four categories of regions are distinguished:

1. Face-up: the lower part is enclosed by the foreground strokes and the upper part is open to exterior of the word.
2. Face-down: the upper part is enclosed by the foreground strokes, the lower is open.
3. Loop: the whole region is enclosed by foreground strokes.
4. Other: any region not fitting into any of the above categories (for example connected regions not enclosed neither from below nor from above).

Examples of regions from main categories are shown in Fig. 6. Determining a category that a given region belongs to is done with vertical scan of all its columns



Fig. 6. Background regions: L – Loop, FU – Face-up region, FD – Face-down region

and counting the number of transitions from the background to foreground color (details can be found in [21]). The most interesting from the segmentation algorithm creator’s point of view are the two foremost. In cursive handwriting, ligatures between two adjacent characters most often form valleys or concavities, which, from the background region analysis point of view, represent the face-up regions. Some letters (like *m*, *n*, *r*) connect to others with a line placed above the vertical coordinate of the centre of gravity. In this case a ligature creates a face-down background region. Sometimes letters are connected by lines that are directed from the lower part of the left character to the upper part of the right character. Mostly in cases where a descender of the first letter (in letters *g*, *j*, *y*) is directly connected to an ascender of the second one (in letters *b*, *f*, *h*, *k*, *l*, *p*). In such case a pair of background regions can be formed that share a common line: a face-up region followed by a face-down region. This situation can be utilized for detecting the beginning as well as the end of the auxiliary line connecting two neighbouring characters, which allows for discarding it in purpose of making further character classification easier. Loops are mostly integral parts of letters (*a*, *b*, *d*, *e*, *f*, *g*, *o*, *p*), so not good candidates for split points. The background open from both sides – the upper and lower, are common between characters that were originally completely separated. Whether to trace a segmentation paths inside background regions from the 4th category depends then on the kind of input for the whole segmentation solution – images of whole words, or connected components.

The idea of the background regions based segmentation is to trace a segmentation path inside the face-up or face-down region. However, there is no universal rule stating exactly how such a dividing line should be formed. Xiao and Leedham in [21] recognized additionally two subcategories of each face-up and face-down regions categories: wide and narrow regions. A shape of the region was the main factor when determining region membership to one of the subcategories. Wide and obtuse regions come into first one, narrow ones ended with sharp valleys or peaks into second one. Because usually wide regions are formed by the additional strokes connecting two letters, segmentation points were found in the middle of the lower (in case of face-up regions) or upper (face-down) bounding lines. With narrow regions, their convexes were used as the segmentation anchorage. Segmentation paths were traced along straight lines directing one of the 6 constant directions: left-down, downward and right-down for the face-up regions and left-up, upward or right-up in the case of the face-down regions. The decision which direction to choose was made to minimize the number of foreground pixels passed by the line.

The oversegmentation algorithm in the segmentation method described in [21] analyzed consecutive connected components obtained from the image of the word. If the component was two times wider than a mean width of the character in the examined text, it was made divided by a path going through every face-down background region. It led to creation of smaller sub-components, the number of which was greater by 1 from the number of all face-down background regions in the divided component. Each sub-component, with width more than two times greater than a mean character width was divided into smaller ones traced through its all face-up background regions. Zheng, Hassin and Tang used background regions defined similar to those described in [21] in segmentation of Arabic characters [5]. However, background regions were used to assess the validity of candidate splitting points generated with the vertical projection analysis method.

3. Evaluation of split points correctness

The goal of oversegmentation is to minimize the risk of segments containing more than one character appearing as output of the segmentation process. This goal is approached by generating greater number of candidate points than the expected number of true split points. The final decision concerning selection of appropriate splitting points is influenced or simply made by the classifier. However, because classification is a relatively time-expensive process, it is desirable that the number of image segments undergoing classifier evaluation was not too much greater (though still greater) than the number of characters in the word being recognized. That is why a preliminary verification of splitting points validity, aiming at rejecting those points which have comparatively small chance of being the right ones, is a necessary step in oversegmentation based segmentation methods. The methods for preliminary verification of candidate split points mainly fall into three general categories:

- rule based approaches,
- cost function optimization based approaches,
- machine learning based approaches.

There are also many solutions combining different approaches (in most cases this means that an evaluation by the cost function or neural nets is preceded by checking simple rules). Examples of systems using each of the approaches are given in the following paragraphs.

3.1. Rule based approaches

Rule based solutions are founded on the observation that both characters and the connection points expose some universal (even if only in the area of one script) traits, lack of which causes immediate qualification of the examined object (a character or a connection between two characters respectively) as invalid. Rules utilized for assessment of potential segmentation points or paths are often intuitive, as they usually emerge directly from analysis of a characters shape, their typical positioning relative to a baseline and how the different characters can be connected to each other. The rules for addition of missed splitting points or adjusting the existing ones (by slight transposition) are also found among commonly applied rules.

One of the most often utilized rules forbids segmentation paths traversing loops. In a neat handwriting loops should be found only in characters a, b, d, e, f, g, o, p and q, although in somewhat less neat handwriting they are sometimes found in the upper parts of letters b, h, k and l. In all those cases loops are integral parts of characters and, indeed, should not be divided among different segments. However, applying this rule can block detection of inter-character connections in even less neat handwriting, where accidental loop forming can occur (for example in pairs with first character c: ca, cb, cc, etc.). A rule forcing the segmentation points directly before and after loops is another one with common application. This one can lead to erroneous separating a loop from a vertical stroke in letters b and d, if those parts are not very tightly attached. Additionally, this rule is best restricted to the loops positioned between upper and lower baselines, as then it does not separate upper and lower parts of letters b, h, k or l, if the loop is formed in their ascenders.

Rules examining the size and placing of a potential segment are also often used. Segmentation paths that cause the segments to be smaller than a certain threshold, as well as those that will cause the segment to be completely above the upper or below the lower baseline are rejected. Segmentation points, that are too close to each other are combined into one (a threshold value for their distance is calculated from a mean stroke thickness in the analyzed word).

A method introduced by Nicchiotti and Scagliola in [12] can serve as an example of a segmentation method in which applying set of rules is the only form of initially proposed points assessment. A set of possible segmentation points in this method was created from local minima of the upper, lower and median contours. All rules described above were used to verify correctness of these points. Verma in [22] engaged a similar set of rules for checking a pool of points made with vertical projection analysis. The splitting points inside loops were removed, while points after the end of the loop were added. So called hat shapes (shapes \vee and \wedge appearing in letters m, n, r, v and w) were treated in a similar manner. The additional rule was responsible for generating new splitting points between two splitting points that were more distant than some threshold value calculated from a mean distance between all splitting points. Verma, however, did not restrict the method to verification with rules, and utilized neural networks to find and remove points lying too close to each other.

Apart from those for removing spurious and adding missing, also rules for improving initially formed segmentation paths are used. In a system designed to rec-

ognize handwritten dates in Brazilian bank cheques described in [13], rules have been used for moving segmentation paths determined initially as the vertical lines going through the local minima of the upper contour of the binarized word image. Transposition was made when: the segmentation path crossed an inner loop before reaching the lower contour, was tangent with the lower contour or crossed to many foreground pixels. A new point was looked for in the proximity of the original point first right, and when no satisfying segmentation point was found, the search continued left to the original point. Additionally, points being too close to each other, or being too close to left or right boundaries of the connected component being examined were eliminated.

In [23] rules were used directly for establishing a lower and upper bound for the number of processed segments. As it is pointed out in the [23], most characters that fit between upper and lower baselines (a, c, e, etc.) have the width similar to their height. Other letters expose similar characteristic after removing their ascenders and descenders. The only exceptions in English alphabet are characters m and w, which width, in general case, can be approximated as $\frac{3}{2}height$ and characters i, j and l, where the width is more or less $\frac{1}{2}height$. The *height* appearing in those formulas is an estimated height of the main body of the word (without considering ascenders and descenders). Vertical projection minima are considered the segmentation points, their number, however, is restricted to L_w/H_m , where L_w is a word length, and H_m – height of the main body of the word. In addition, the distance between every two adjacent points cannot be less than $\frac{1}{3}H_m$. On the other hand, if the distance between two successive segmentation points is greater than the main height, the algorithm searches for a minimum of the vertical histogram in a desired area between them to place there a new segmentation point.

Aforementioned rules are based on very general properties of characters and ligatures in handwritten words. In more complicated approaches the rules are created based on more specific traits of certain groups of characters. For example Xiao and Leedham ([21]) for the purpose of verification of the split points achieved with background regions analysis, divided analyzed fragments among four classes: components covering only the middle zone of the word (a space between upper and lower baselines) such as lowercase a, c, o or m (Class One), components covering the upper zone and the middle zone of the word, such as most uppercase characters and lowercase characters with ascenders (Class Two), components covering the middle and the lower zone of the word, that is creating characters with descenders, i.e. g, j, p and y (Class Three) and components covering all three zones, such as lowercase f (Class Four). Pairs of adjacent segments were analyzed for verification if they belonged to one or more characters (which is equivalent with checking validity of the segmentation point separating the segments). There were four most common types of cases: both sub-components belong to the same class (Type 1), one is of Class Two (Type 2), one belongs to Class Three (Type 3) and the last one: one of the components is of Class Four. Based on the four component classes, four types of pairs of components and the spatial relationships of components and background regions that were crucial in separating given components, a set of rules was defined, some of that rules forbidding or allowing for deletion of a given splitting point from the candidate splitting points pool. Rules based on detailed analysis of specific alphabet characters have been also used for splitting points verification in

the automated system for segmentation of strings of Arabic characters, described in [5].

3.2. Approaches based on cost function optimization

Verification is often made based on a cost function analysis. The exact form is created mostly with regard to certain features of split points or split paths. Sometimes resulting segments characteristics are also taken under consideration, as well as general characteristics of the word like the line thickness. When the function formula is ready, potential split points are given as arguments, and those minimising (or, in some cases, maximising) the function value are selected as proper ones. In [9] Yanikoglu and Sandon proposed a segmentation method in which verification was conducted with a cost function based on writing style parameters that are characteristic for a particular text author: the pen thickness, average letter width and height of the strokes. The dominant slant also considerably influenced a segmentation process. Potential points splitting connected components were determined using angled projection analysis made in five directions, depending on the dominant slant. The image was split among lines drawn at angles for which the projection was calculated. Segmentation paths were therefore formed by a pair: an angle of the separator line and a point on the baseline (a horizontal coordinate of the point at which this line intercepts the baseline). Proper splitting points were selected iteratively from the left border of the image. Given the segmentation point and its corresponding separator line angle, a point minimising the cost function within the pool of potential splitting points being on the right from the last one, was selected. In the case of several points sharing the minimum, the closest was chosen. The cost function was calculated according to the formula:

$$C(a, p) = w_1 \left(\frac{p - s}{W} \right)^2 - w_2 \left(\frac{p - s}{W} \right) + w_3 \left(\frac{t}{T} \right) + w_4 \left(\frac{h}{H} \right), \quad (1)$$

where p is the point on the baseline, a is the separator line angle, s is a given previous segmentation point, t is the number of foreground pixels cut by the separator line, h is the height of the highest placed one of these foreground pixels and W , T and H are estimated values of style parameters: the average word width, pen thickness and total height of the line. Coefficients w_i are weights differentiating influence of particular elements on the cost function value. They are found using linear programming described in [9]. The same cost function was applied for resolving the problem of multiplied segmentation points. The pool of potential splitting points was in this case the union of points achieved separately for each angle with projection analysis. Therefore many splits were represented by several close but not identical splitting points. Before the actual verification, one point was selected from each such a group, the one minimising the cost function value. The writer independence is, undoubtedly, a positive trait of this cost function.

An example of another approach can be found in [19]. Liang and Shi used a cost function formed by detailed analysis of characteristics describing segmentation paths

for segmenting handwritten Chinese character strings. They used a background skeleton analysis for generation of potential splitting paths. Next, for each such path a feature vector was created based on a set of spatial characteristics of the path such as: spatial coordinates of the end points and outermost points on the path, its horizontal centre or the number of foreground pixels through which the path is passing. The cost function was the mixture probability density function:

$$P(x | \Theta) = \sum_{i=1}^m a_i p_i(x | \Theta_i), \quad (2)$$

where

$$p_i(x | \mu_i, \Sigma_i) = \frac{e^{-\frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i)}}{\left((2\pi)^{\frac{d}{2}} |\Sigma_i|^{\frac{1}{2}}\right)}, \quad (3)$$

$$\Theta = (a_1, \dots, a_m; \Theta_1, \dots, \Theta_m), \quad (4)$$

$$\sum_{i=1}^m a_i = 1. \quad (5)$$

Parameters were established using the expectation-maximization (EM) algorithm. It was trained on feature vectors obtained from the correct segmentation paths. When the cost function was ready, those points among the potential splitting points, that maximised its value were selected.

3.3. Machine learning based approaches

In machine learning based approaches separate classifiers, designed specially for splitting points verification, are used. Each point from the potential splitting points set has a feature vector assigned, which in turn makes the input of a classifier (a neural network, for example). The result of classification decides whether the candidate points should be removed from the pool or left there. The ability to cooperate with any method of generating candidate points is a remarkable benefit of this approach. Feature vectors are usually created based on properties of close proximity of splitting points under analysis.

In [24] the classifier used for this task was a neural network with supervised learning. Splitting points for the training set of the network were found with several techniques (projection analysis, contour analysis, loops detection). Next, they underwent manual classification into two classes: "correct" and "incorrect". Only the validity of the x-coordinate of splitting point was assessed (in contrast to verifying exact segmentation points or segmentation paths). The feature vector was made as follows. From the binarized image of the word, a rectangle of some small, preset width, containing the candidate point, was chosen. Afterwards, this rectangle was normalized in size, and divided into square-shaped windows of some set and small

size. For each of the windows a quotient of the number of foreground pixels found in the window to the window size was computed. These numbers were the elements of the feature vector. The class value was encoded as 0.1 for "incorrect" points and 0.9 for points classified as "correct". In the experiments, vectors of length 42 were used (each splitting point was represented as 14×3 windows). After training vectors representing points generated for words that were meant to be automatically segmented were given as an input for the network. Only those verified by the network as correct were considered further as possible splitting points.

An interesting approach was presented in [25]. Authors introduced a filter reducing the number of unnecessary cut points generated with an oversegmentation approach. As it was emphasized in the article, spurious cut points generate oversegmented characters (digits in the case of the presented method). Because of that, cuts were not analysed with respect to some structural characteristics, as it is with most of the earlier methods, but with respect to properties of segments generated by them. It introduces an additional context, and so increases the information volume taken into account while verifying segmentation points. Because filtration was viewed as a two-class problem, (necessary and unnecessary segmentation cuts), SVM was used to model it. For each of the cut points undergoing filtration, a pair of components that this cut point partitions the segment to, as well as the whole, unpartitioned segment (that is the case where a cut point was not taken into account) were analyzed. Each background pixel within all three components (two after split, and one before) had a label assigned. This label was a number of foreground pixels accessible from that pixel in the 4-Freeman directions (those flags were described as concavity levels). Pixels that were inside loops were labelled separately. Next, the algorithm searched for pixels with labels different in full (i.e. not splitted component) from those in sub-components. Such pixels achieved another label, distinct from previously described kinds. The whole component was divided into the constant number of frames. In each frame a number of pixels marked with each label was counted. Values achieved in such a manner made a feature vector representing a given cut point. Because regions were split into 2×3 frames, and 7 different labels were used, vectors had 42 values (the same as in the solution described previously). At the cost of decreased granularity (a smaller number of frames), a broader spectrum of information was given as a SVM input. Authors justify such selection of features by observation that when digits are segmented correctly, there is greater difference in concavity levels between the unsegmented part and segmented parts, than if the digits are over-segmented. That is why pixels with those changes – marked here with a special label – are so important. They also state, that those changes apart from having different amount occur in different areas of the considered piece when the fragment is split correctly than they occur in over-segmented digits. This behaviour is detected by having those 6 frames, covering different areas of the considered image fragment. This filter was designed for filtration of cut points generated for strings of digits. Authors described experiments done with cut points generated by two algorithms performing oversegmentation: the contour and profile based Fenrich's algorithm and the skeleton based algorithm proposed by Chen and Wang. For the first one the 83% decrease in the number of unnecessary segmentation hypothesis was reported.

4. Conclusions

In recent years methods using oversegmentation outnumbered other methods of segmentation. It comes from a fact that proper segmentation of handwriting is often impossible to separate from recognition of single characters. In the case of digit strings the problem is significantly easier to solve, but also here combining steps of segmentation and classification by using oversegmentation leads to considerable improvement of results. Generating of potential split points itself is done in various ways. Each has its drawbacks and advantages, none is universal. Selection of a particular technique should be dependent on the task, primarily on the kind of alphabet the character strings are built over. For example, methods based on the contour are especially suitable for the Latin alphabet and its derivatives, while for Chinese characters methods based on the skeleton analysis (both foreground and background) perform much better.

Because in the case of systems based on oversegmentation omitting some proper splitting points in the stage of generating candidate splitting points has great negative impact on accuracy of the whole solution, often two or more methods described in Section 2 are combined to minimise danger of such error occurring. However, such a solution comes with increased computational complexity and need for solving a problem of single candidate splitting points being seen as multiple: some of the possible splitting places are detected by multiple methods. Points representing the same split, but generated with different methods will often have slightly different positions. The problem of selecting the best point is not trivial, because quite often moving a segmentation path even only by a few pixels can significantly influence ability to properly recognise the character. Additionally, in case of letter "i" it is possible, that two correct segmentation points are only 1 pixel apart. That fact causes the decision, if two candidate points are in fact the same, to be impossible to make based only on the distance between points. In solutions using single techniques the ones that allow to detect most characteristic points that can correspond to connected characters are chosen. They should be relatively resistant to errors that come from slant or overlapping lines. Such requirements are, as for now, best matched by contour based approaches, therefore such solutions are most commonly used.

The greater number of analysed segments is the greater chance for proper character classification and recognition of the word. But classification is a time-costly process, that is why reducing the final number of splitting points generated in oversegmentation is important. Currently attention is focused on designing new and improving existing verification methods rather than on developing methods generating potential split points. Using preset rules allows to reject points having some clearly defined and easy to check traits, that in most cases disqualify them as split points. However, variety in handwriting styles and its frequent deformations may cause strict rules to reject proper splitting points. That is why fuzzy rules are used more frequently. They are used not to make a binary decision but only to assess the possibility of a given point to represent a connection between two separate characters. Such activity brings rule based approaches closer to the cost function based

approaches. The cost function, while evaluated on some point, takes into account more characteristics of a point than a single rule does. This allows for detection of some dependencies harder to spot with simple visual analysis. This property is exposed even more in machine learning approaches. There are also solutions that compile different technique results in the stage of verification. In such cases usually preliminary assessment of splitting points is done with rule based expert systems but the actual evaluation is done with neural networks. Great numbers and variety of emerging solutions suggest that splitting the word image into segments corresponding to single characters is still quite a challenge in handwritten word recognition.

5. References

- [1] Casey R.G., Lecolinet E.; *A Survey of Methods and Strategies in Character Segmentation*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 18(7), 1996, pp. 690–706.
- [2] Plamondon R., Srihari S.N.; *On-Line and Off-Line Handwriting Recognition: A Comprehensive Survey*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(1), 2000, pp. 63–84.
- [3] Lu Y., Shridhar M.; *Character segmentation in Handwritten Words – An Overview*, Pattern Recognition, 29, 1996, pp. 77–96.
- [4] Verma B.; *A Contour Code Feature Based Segmentation For Handwriting Recognition*, Proceedings of the Seventh International Conference on Document Analysis and Recognition, 2, 2003, pp. 1203.
- [5] Zheng L., Hassin A.H., Tang X.; *A new algorithm for machine printed Arabic character segmentation*, Pattern Recognition Letters, 25, 2004, pp. 1723–1729.
- [6] Nicchiotti G., Scagliola C.; *Generalised Projections: a Tool for Cursive Handwriting Normalisation*, International Conference on Document Analysis and Recognition, 5, 1999, pp. 729.
- [7] Kavallieratou E., Fakotakis N., Kokkinakis G.; *A slant removal algorithm*, Pattern Recognition, 33, 2000, pp. 1261–1262.
- [8] Leedham C.G., Friday P.D.; *Isolating individual handwritten characters*, Proc. IEE Colloq. Character Recognition and Applications, 1989, pp. 4/1–4/7.
- [9] Yanikoglu B., Sandon P.A.; *Segmentation of off-line cursive handwriting using linear programming*, Pattern Recognition, 31(12), 1998, pp. 1825–1833.
- [10] Zeeuw G. de; *Slant Correction using Histograms*, Bachelor's thesis, University of Groningen, 2006. Available via http://www.ai.rug.nl/~axel/teaching/bachelorprojects/zeeuw_slantcorrection.pdf.

- [11] Madhvanath S., Kim G., Govindaraju V.; *Chaincode Contour Processing for Handwritten Word Recognition*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 21(9), 1999, pp. 928–932.
- [12] Nicchiotti G., Scagliola C., Rimassa S.; *A Simple And Effective Cursive Word Segmentation Method*, Proceedings of the Seventh International Workshop on Frontiers in Handwriting Recognition, 2000, pp. 11–13.
- [13] Morita M., Lethelier E., Yacoubi A. El, Bortolozzi F., Sabourin R.; *An HMM-based Approach for Date Recognition*, Proceedings of the Fourth IAPR International Workshop on Document Analysis Systems, 2000.
- [14] Bozekova M.; *Comparison of Handwritings*, Diploma thesis, Comenius University, 2008.
- [15] Oliveira L.S.; *Automatic Recognition of Handwritten Numerical Strings*, PhD thesis, Ecole de Technologie Superieure, 2003.
- [16] Jang B.K., Chin R.T.; *One-Pass Parallel Thinning: Analysis, Properties, and Quantitative Evaluation*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 14(11), 1992, pp. 1129–1140.
- [17] Huang L., Wan G., Liu C.; *An Improved Parallel Thinning Algorithm*, Proceedings of the Seventh International Conference on Document Analysis and Recognition, 2003, pp. 780–783.
- [18] Zhao S., Chi Z., Shi P., Yan H.; *Two-stage segmentation of unconstrained handwritten Chinese characters*, Pattern Recognition, 36, 2003, pp. 145–156.
- [19] Liang Z., Shi P.; *A metasynthetic approach for segmenting handwritten Chinese character strings*, Pattern Recognition Letters, 26, 2005, pp. 1498–1511.
- [20] Lu Z., Chi Z., Siu W., Shi P.; *A background-thinning-based approach for separating and recognizing connected handwritten digit strings*, Pattern Recognition, 32, 1999, pp. 921–933.
- [21] Xiao X., Leedham G.; *Knowledge-based English cursive script segmentation*, Pattern Recognition Letters, 21, 2000, pp. 945–954.
- [22] Verma B.; *A Contour Code Feature Based Segmentation For Handwriting Recognition*, Proceedings of the Seventh International Conference on Document Analysis and Recognition, 2003, pp. 1203.
- [23] Kavallieratou E., Fakotakis N., Kokkinakis G.; *An unconstrained handwriting recognition system*, International Journal on Document Analysis and Recognition, 4(4), 2002, pp. 226–242.
- [24] Blumenstein M., Verna B.; *A New Segmentation Algorithm for Handwritten Word Recognition*, Proceedings of the International Joint Conference on Neural Networks, 1999, pp. 2893–2898.
- [25] Vellasques E., Oliveira L.S., Britto A.S. Jr., Koerich A.L., Sabourin R.; *Filtering segmentation cuts for digit string recognition*, Pattern Recognition, 41(10), 2008, pp. 3044–3053.

- [26] Madhvanath S., Govindaraju V.; *The Role of Holistic Paradigms in Handwritten Word Recognition*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 23(2), 2001, pp. 149–164.
- [27] Vinciarelli A.; *A survey on off-line Cursive Word Recognition*, Pattern Recognition, 35, 2002, pp. 1433–1446.
- [28] Guillevic D.; *Unconstrained handwriting recognition applied to the processing of bank cheques*, PhD thesis, Concordia University, 1995.
- [29] Liu Z.-Q., Cai J., Buse R.; *Handwriting Recognition, Soft Computing and Probabilistic Approaches*, Springer, 2003.

Received February 9, 2010