Analysis and Understanding of GIRO Check Forms *

Thien HA MINH and Horst BUNKE
University of Berne Institut für Informatik und Angewandte Mathematik
Länggassstr. 51, CH-3012 Bern, Switzerland

November 18, 1992

Abstract

Check forms are used by many people in daily life for money remittance. Surprisingly, the processing of these forms at banks and post offices is only partly automated. In this report, we focus on a particular kind of form, viz., the GIRO checks used in Switzerland. We will describe a fully automatic system which is able to recognise the following items on a GIRO check: the financial institution, the name and address of the receiver and the account number. The complete analysis and understanding of a GIRO check is performed in two phases. In the first phase, the system performs a layout analysis in order to localise regions corresponding to various items on the check. The input gray-level image is first binarised and segmented using the X-Y-tree decomposition algorithm. This gives us the atomic parts of a check, for example, individual characters, special symbols, and formatting lines which are present on a form. Each entity is then interpreted as a part of an item (e.g., receiver's name, account number), according to the knowledge about possible layouts of a form. All atomic entities belonging to the same item are grouped together and yield the location of that item. In the second phase, the localised items are separately analysed by a local binarisation and the binarised sub-images are submitted to an OCR engine to obtain the streams of characters. We have tested the system on a large number of checks and the results are promising in terms of both computation time and recognition accuracy.

CR Categories and Subject Descriptors: I.4.6 [Image Processing]: Segmentation; I.4.7 [Image Processing]: Feature Measurement; I.5.1 [Pattern Recognition]: Models.
Additional Key Words: GIRO check, Optical Character Recognition (OCR), document image recognition, graph matching.

*This work is part of the joint project betwen the UBS (Union Bank of Switzerland) Laboratory and the University of Berne. The authors wish to thank the UBS Laboratory for the cooperation and the financial support.

Contents

1 INTRODUCTION

This report describes the results achieved throughout the second year of the joint project between the UBS Laboratory (UBILAB) and the University of Berne on "Document Image Analysis and Understanding". In this project, we focus on a particular kind of document, viz., the GIRO checks used in Switzerland. The emphasis has been put on a complete analysis and understanding of the left and bottom parts of these checks.

Up to the time of writing this report, we have implemented a software system capable of recognising the following items on a GIRO check: the financial institution, the name and address of the receiver, the account number on the left, and the 'OCR lines' at the bottom of the check. The obtained results are promising in terms of both computation time and recognition accuracy.

Section 2 presents the methodology of model-based recognition applied to the considered problem. Section 3 discusses the segmentation of the input image into atomic entities and Section 4 the labeling of these entities. Some results are presented in Section 5 and the last section gives a perspective of future research and concludes the report.

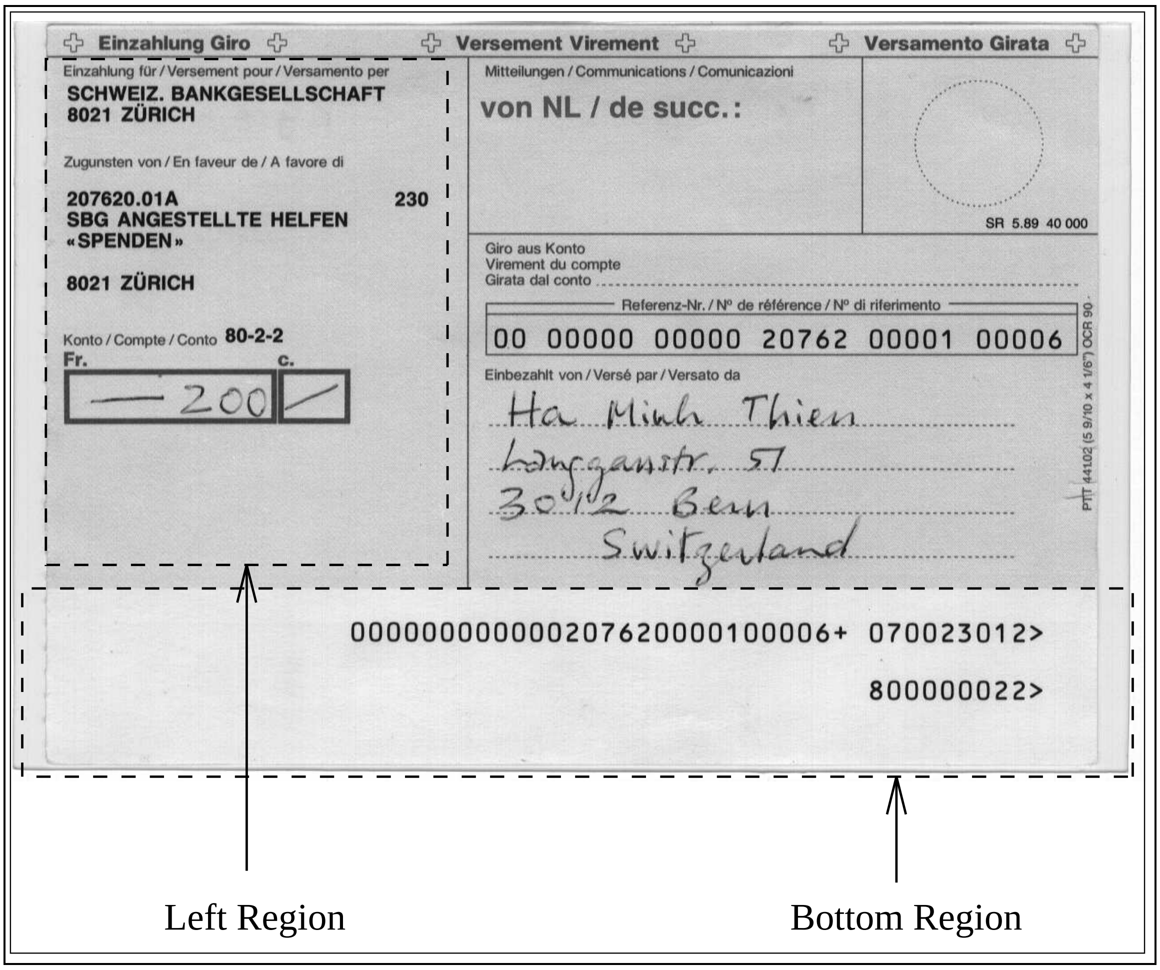
2 METHODOLOGY

GIRO checks are documents printed according to a set of rules specified by the Postes Telegraphes et Telecommunications (PTT) of Switzerland. This means that the recognition process may and should take advantage of these rules to guide the analysis and understanding phases. In other words, the recognition will be model-based. In this application, we are mainly interested in two regions of GIRO checks, namely, the left and bottom ones. For an illustration, see Fig. 1. The left region contains various kinds of information such as the name and address of the receiver, account number, etc. On the other hand, the bottom region, also called 'OCR lines' region, contains numerical codes including few specific symbols (e.g., '>', '+'). The main difference between these two regions is that the left one has a flexible layout whereas the structure of the bottom one obeys very strict rules. Therefore, the analysis of the latter is much easier and in fact can be achieved independently of the former. Note that there already exist commercial systems that can read the bottom region. Consequently, we will present only the recognition of the left region of GIRO checks in this report.

Section 2.1 introduces the document model based on graph representation and 2.2 describes the recognition method.

2.1 Document Model

Graphs are a powerful tool to represent knowledge. A graph is composed of nodes and edges linking the nodes together. In our application, nodes are associated with


Left Region
Bottom Region

Einzahlung Giro Versement Virement Versamento Girata

Einzahlung für / Versement pour / Versamento per
SCHWEIZ. BANKGESELLSCHAFT
8021 ZÜRICH

Zugunsten von / En faveur de / A favore di
207620.01A 230
SBG ANGESTELLTE HELFEN
«SPENDEN»

8021 ZÜRICH

Konto / Compte / Conto 80-2-2
Fr. c.
—200 /

Mitteilungen / Communications / Comunicazioni
von NL / de succ.:

SR 5.89 40 000

Giro aus Konto
Virement du compte
Girata dal conto

Referenz-Nr. / N° de référence / N° di riferimento
00 00000 00000 20762 00001 00006

Einbezahlt von / Versé par / Versato da
Ha Minh Thien
Langgasstr. 57
3012 Bern
Switzerland

PTT 44102 (5 9/10 x 4 1/6") OCR 90

000000000000207620000100006+ 070023012>
800000022>

Figure 1: A GIRO check form.

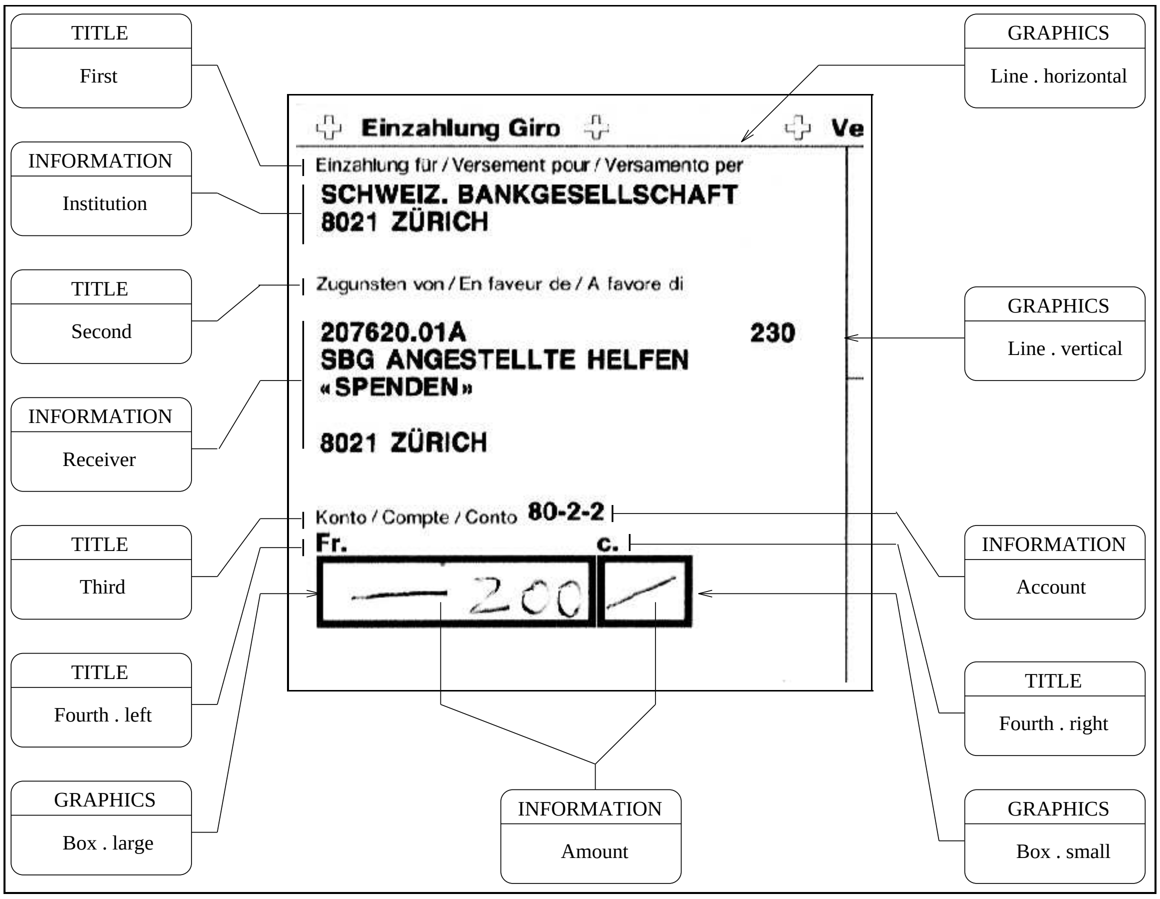


Figure 2: Objects on a GIRO check form.

various objects in the document (e.g., lines, boxes, strings of characters, etc.) and edges represent spatial relationships (e.g., above, left-to and inside) between objects.

Building a model graph is highly dependent on the particular application. Indeed, the building process involves 1) the choice of a set of objects and 2) the definition of their relationships. There are usually many possible combinations of objects and the choice of one over the other is a priori not obvious. Nevertheless, there exist some basic rules that guide an appropriate choice:

- The total number of objects should not be too large (to limit the time complexity of the matching process). This can be achieved through a hierarchical organisation.
- The choosen objects should be easily identifiable by some known procedures.
- The way human readers perform the task may sometimes be used as a cue.

Note that the first and the second rules may be contradictory under certain circumstances and thus can be resolved only by a trade-off.

In our application, the objects we have choosen to be associated with the nodes of the model graph can be classified into three categories (see Fig. 2):

- GRAPHICS (Lines and Boxes).

- TITLE (Predefined strings of characters).
- INFORMATION (Financial institution, receiver, account number and amount).

We will see later that this classification helps determining the 'best' order of recognition. There are three kinds of spatial relationships between the considered objects:

- Above.
- Left-of.
- Inside.

Thus, an object A linked to an object B by an 'above' edge means that A *is above* B. Relations between two objects are qualified as *binary*. *Unary* relations as well as attributes on the other hand describe intrinsic properties of an object (e.g. length of a GRAPHICS line, sizes of a GRAPHICS box, contents of a TITLE object, sizes of characters in an INFORMATION object). Fig. 3 gives the model graph of the GIRO checks. For simplicity, only binary relations are shown in this figure; unary relations and attributes have been omitted.

2.2 Recognition Task

The overall goal of the recognition process is to receive an input image and extract the INFORMATION objects of interest by using the document model. In this section, we will discuss how this goal can be achieved in an efficient manner. More specifically, we will show that the recognition process can be decomposed into a number of tasks, namely, segmentation, identification of conspicuous objects and finally recognition of characters within certain zones of interest.

Given the above model, one possibility is to 1) use an OCR engine to recognise all possible characters in the image and b) match these characters against the document model. However, some preliminary tests have shown that there are many drawbacks associated with this approach:

- Before any OCR engine can be applied, the document image has to be binarised. However, as the image contrast is not uniform - GRAPHICS and TITLE objects have low contrast whereas INFORMATION objects often have high contrast - a global binarisation does not yield a satisfactory result and a local binarisation is difficult because we do not know where the objects of interest lie. A local adaptive binarisation could be imagined but would be too complicated for this application.
- TITLE objects are of small size and can be reliably recognised by an OCR engine only at very high resolution. Even when this is possible, it is very time consuming.

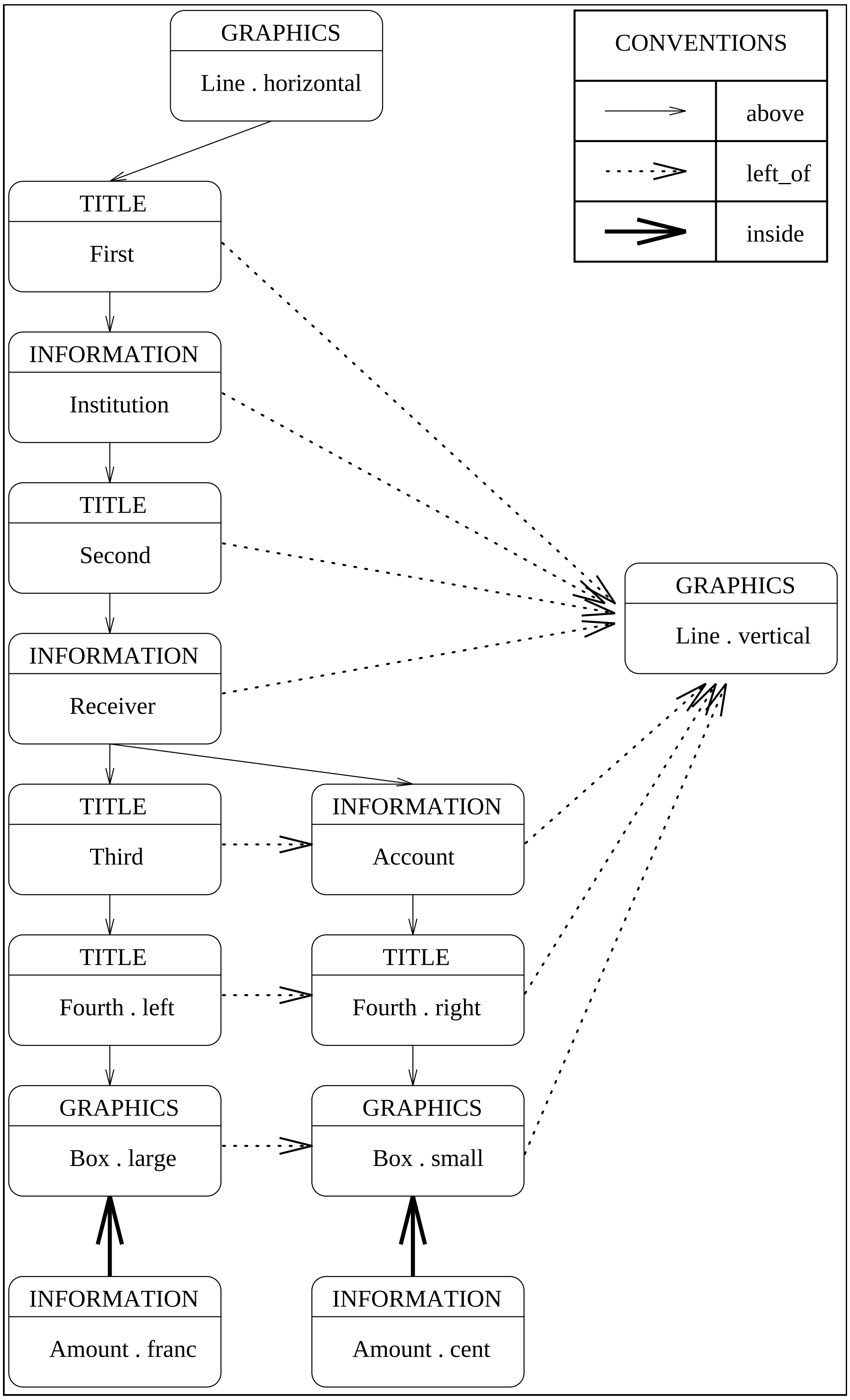


Figure 3: Model graph of GIRO check forms.

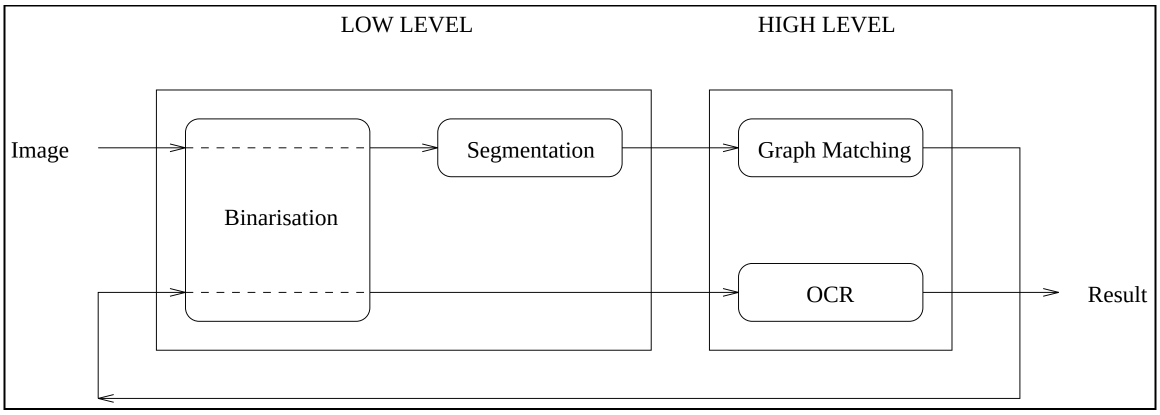


Figure 4: Block diagram of the recognition process.

- The presence of GRAPHICS objects makes the OCR engine work more slowly and may even disturb the correct localisation of characters.

In summary, this approach is time consuming and not reliable. Therefore, we have adopted another approach which consists of two phases (see Fig. 4). In the first phase, the whole image is segmented into atomic entities. Based on the document model, the matching algorithm groups these enties into regions according to the model objects in decreasing order of conspicuousness. A one-to-one correspondence between regions and objects is performed thus identifying regions that contain INFORMATION objects. In the second phase, the INFORMATION regions are separately binarised again and their binary bitmaps are fed into the OCR engine yielding the final result.

Low-level processing comprises binarisation and segmentation. The binarisation consists of determining a threshold that separates the graphics and characters (dark) from the background (bright). This operation reduces the amount of information and thus eases subsequent operations. The segmentation divides the binary image into indivisible zones, called atomic entities. Each atomic entity normally corresponds to a character or a graphics part. Both binarisation and segmentation algorithms will be discussed in the next section.

High-level processing consists of graph matching and the OCR engine. The graph matching algorithm in our application is restricted to the identification of GRAPHICS and TITLE objects, leaving out the INFORMATION ones. In other words, we use GRAPHICS and TITLE objects as a kind of islands in order to localise the INFORMATION objects. This is possible because GRAPHICS and TITLE objects can be identified by using only their intrinsic properties. This observation allows us to reduce the time complexity of the graph matching process from $\mathcal{O}(N^P)$ to $\mathcal{O}(NP)$, where N is the number of nodes in the model and P the number of data items. The graph matching algorithm will be described in Section 4 together with the application of the OCR engine.

Note that we have defined the decreasing order of conspicuousness of objects as:

GRAPHICS, TITLE and INFORMATION. The GRAPHICS objects are the most conspicuous ones because they have large size and well-defined form. The TITLE objects are more conspicuous than the INFORMATION ones because they have fixed contents, although they are smaller in size.

The main advantage of this two-phase approach is that it is less sensitive to the non-uniform contrast levels encountered in real documents. Additionally, the separation between segmentation and OCR eases the calibration and tuning of the whole system.

3 LOW LEVEL PROCESSING

This section addresses two topics. The first concerns the binarisation and the second a segmentation algorithm, called X-Y-Tree decomposition, to which we have added some improvements leading to a reduction of computation time.

3.1 Binarisation

In this project, we use the automatic threshold selection algorithm proposed by Otsu [Otsu 79]. This algorithm first computes the histogram of the gray-level image and then determines the binarisation threshold. The threshold determination is based on the maximisation of the inter-class variance assuming that the image is composed of only two classes, viz., foreground and background pixels. Since the algorithm makes use of the histogram alone, it is purely statistical; no spatial information is taken into account. However, in the context of our application, this approach has proved sufficient. The details of the implementation of the algorithm may be found in [Spider 83].

The alternative to automatic binarisation is to use a fixed threshold value. Thus, all documents would be binarised using the same threshold. It turns out in our application that the gray-level and contrast of GIRO checks vary over a large range of values. For instance, the value 150 of a pixel (0 and 255 represent Black and White respectively) may correspond to the background in a dark document and to some character in a bright document. Figs. 5 and 6 show the gray-level histograms of two GIRO checks. It is clear from these histograms that a fixed threshold is not suitable for our application.

3.2 X-Y-Tree Decomposition

The X-Y-tree decomposition, also called Iterative Projection Profile Cuttings method, is a popular segmentation algorithm in the context of Document Image Analysis and Understanding [Nagy 84, Srihari 86, Casey 90]. In our application, the X-Y-tree decomposition is purely considered as a low-level algorithm in the sense that no high-level knowledge of the document is required to perform the segmentation. Previous works often, but not exclusively, make uses of high-level knowledge to guide

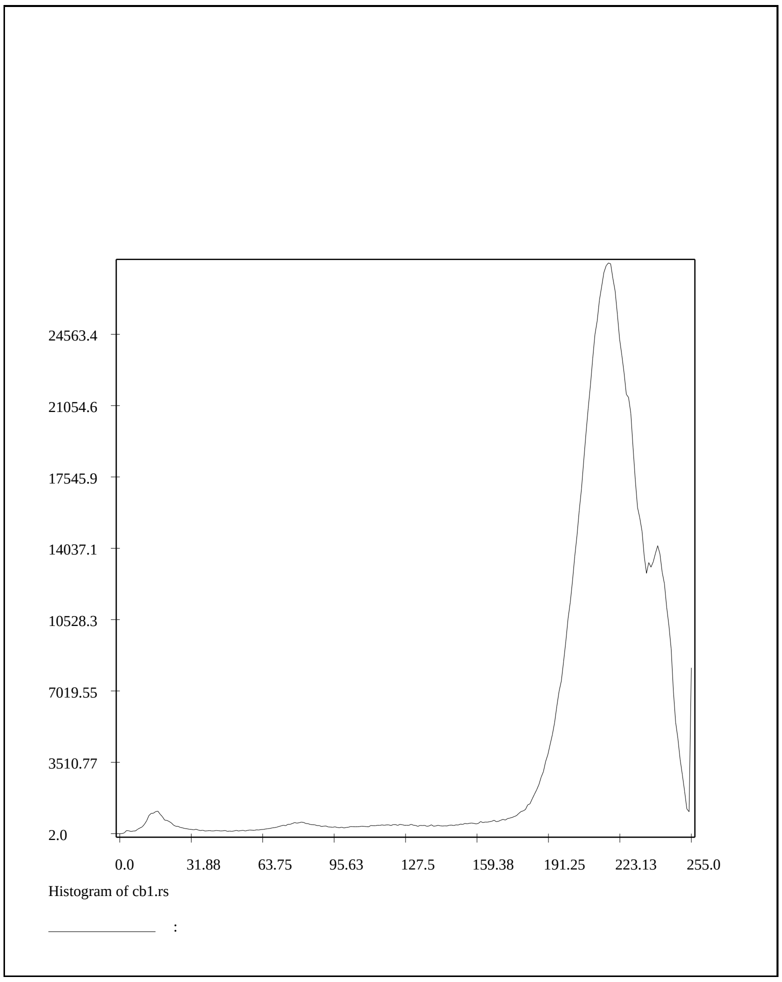


Figure 5: Gray-level histogram of a GIRO check.

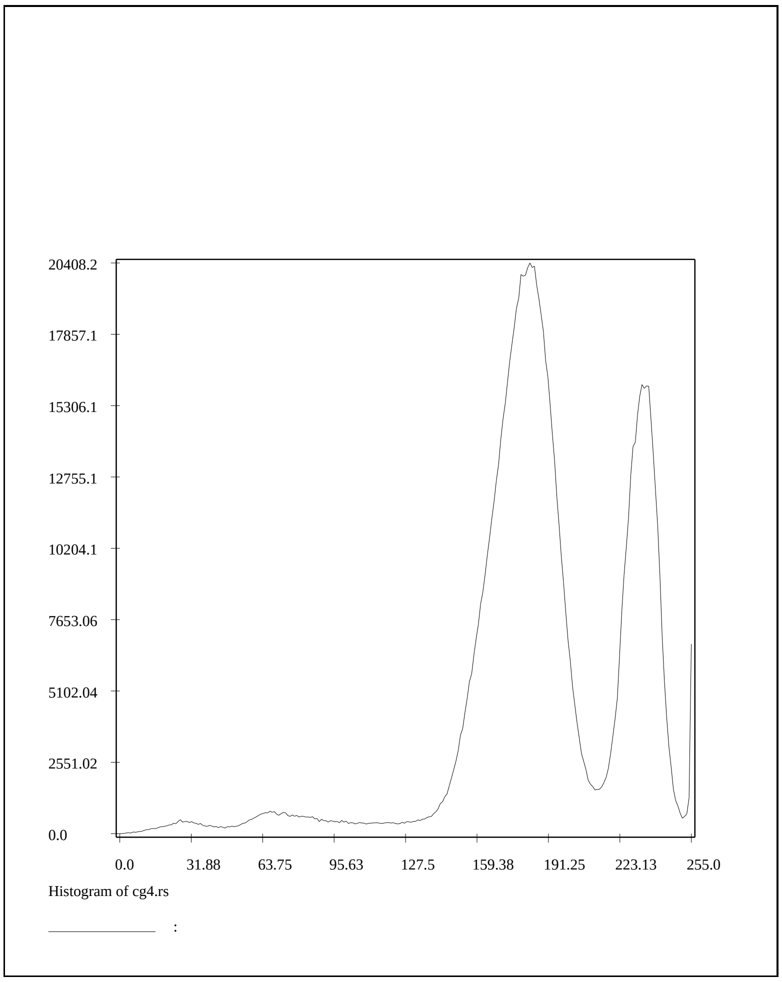
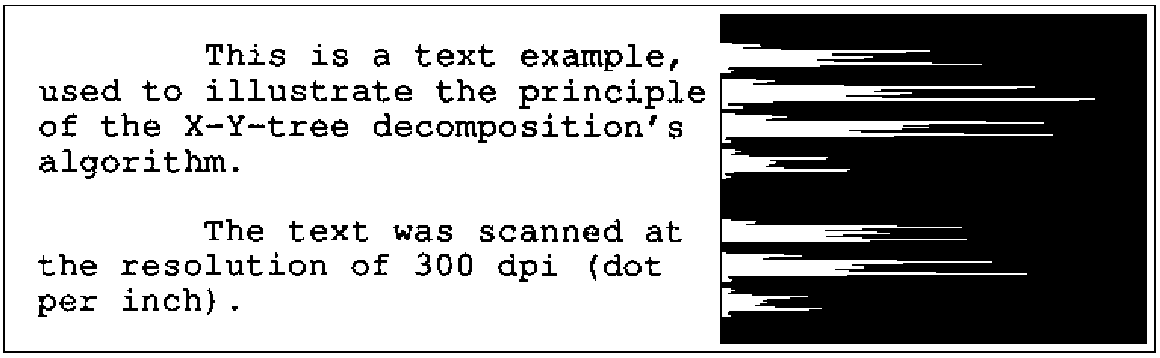


Figure 6: Gray-level histogram of another GIRO check.


This is a text example,
used to illustrate the principle
of the X-Y-tree decomposition's
algorithm.

The text was scanned at
the resolution of 300 dpi (dot
per inch).

Figure 7: Horizontal projection of black pixels.

the decomposition. [Nagy 84] embeds the picture grammar and [Viswanathan 90] introduces the publication-specific knowledge in the segmentation process. This combination of low- and high-level information results in an elegant solution but requires that the document layout has a well-defined structure. More importantly, an error in the data due to noise may irremediably lead to a segmentation failure [Srihari 86]. In the approach where low- and high-level processing are separated, the problem of noisy data is postponed to the later stage of labeling. The advantage is that the amount of data to deal with is then much smaller than in the original representation.

The basic algorithm as it is proposed in literature is presented first. Then the optimisation and the ordering properties of the algorithm are described.

3.2.1 The basic algorithm

The basic idea behind the algorithm is the exploitation of the fact that most document images have a vertical and/or horizontal structure. Indeed, a normal text page is usually composed of different horizontal text lines (see Fig. 7). This observation immediately leads to the idea of horizontally projecting the black pixels on a vertical axis (see Fig. 7). The resulting profile clearly indicates the line structure of the page. A simple *analysis of the profile*, like comparison of the projection profile with a threshold, segments the page into *blank bands* and *text bands*. A blank band corresponds to a set of contiguous rows having less than n black pixels and a text band to a set of contiguous rows having a least n pixel if the comparison's threshold is set to n. Subsequently, each text band can be vertically projected on a horizontal axis yielding the positions of the characters within this band. Fig. 8 shows the vertical projection of the first band. Thus, in the case of the text page of Fig. 7, the following procedure can be used for extracting the characters:

1. Compute the horizontal projection profile for the entire page.
2. Analyse the projection profile to extract the lines.
3. For each line, compute the vertical projection profile.
4. Analyse each projection profile obtained in step 3 to extract the characters.

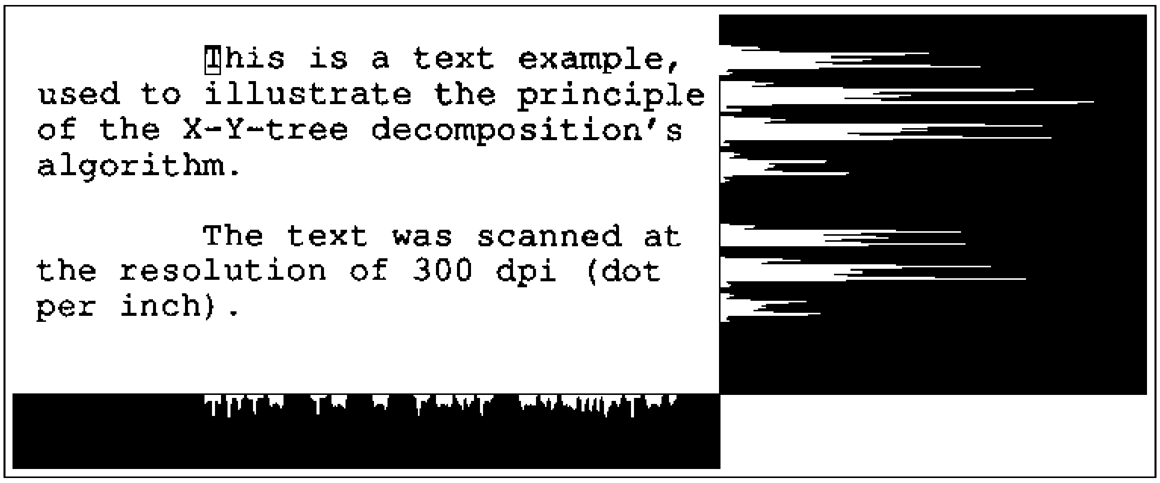

This is a text example,
used to illustrate the principle
of the X-Y-tree decomposition's
algorithm.
The text was scanned at
the resolution of 300 dpi (dot
per inch).

Figure 8: Vertical projection of the first line.

More generally, however, it may be necessary to perform more than two projections to reach the characters. Fig. 9 shows a spatial configuration (characters are symbolically represented by boxes) in which some characters are reached only after the third projection (horizontal). It can readily be seen that more complex documents may require even more projections to reach the individual characters. Therefore, in order to be sure that all indivisible zones are reached, the general algorithm should alternatively project the document vertically and horizontally until two consecutive projections yield the same result.

The natural data structure to store the result of the algorithm is a tree whose nodes represent rectangular zones. Each node may have many children each representing a sub-zone of the parent's zone. The terminal nodes (leaf-nodes) are those which are not further decomposable. They represent the indivisible zones and are called *atomic entities*. Fig. 10 shows an example of text where characters are represented by boxes, and its tree representation.

A scrutiny observation of Fig. 10 reveals that the order in which different atomic entities are reached by the algorithm is that of Fig. 11. It can be seen that this order is the same as that of the leaf-nodes in the tree representation (left-to-right). This suggests a *list representation* of the result, in the same order. The list representation has the advantage of providing easier post-processing operations like string matching. Indeed, it is quite complicated to match a string with a tree data structure, whereas it is much easier to match a string with a list. However, we must be cautious because the X-Y-tree's order is not necessarily that of the 'natural' order of the text as illustrated in Fig. 11. For instance, a human reader would rather see zone 1.2.1 following 1.1.1 and 1.2.2 following 1.1.2 in Fig. 10, and not the order depicted in Fig. 11. This problem will be addressed in Section 3.2.3 where we will establish a relation between the X-Y-tree's order and the spatial relationship between any two atomic entities.

Another problem with the X-Y-tree decomposition algorithm arises in the pres-

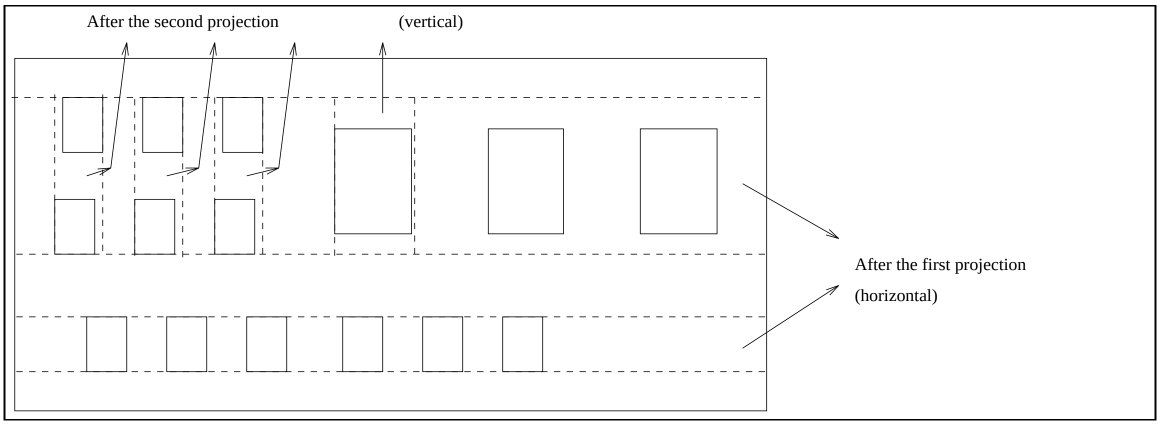


Figure 9: Example of a document requiring three projections.

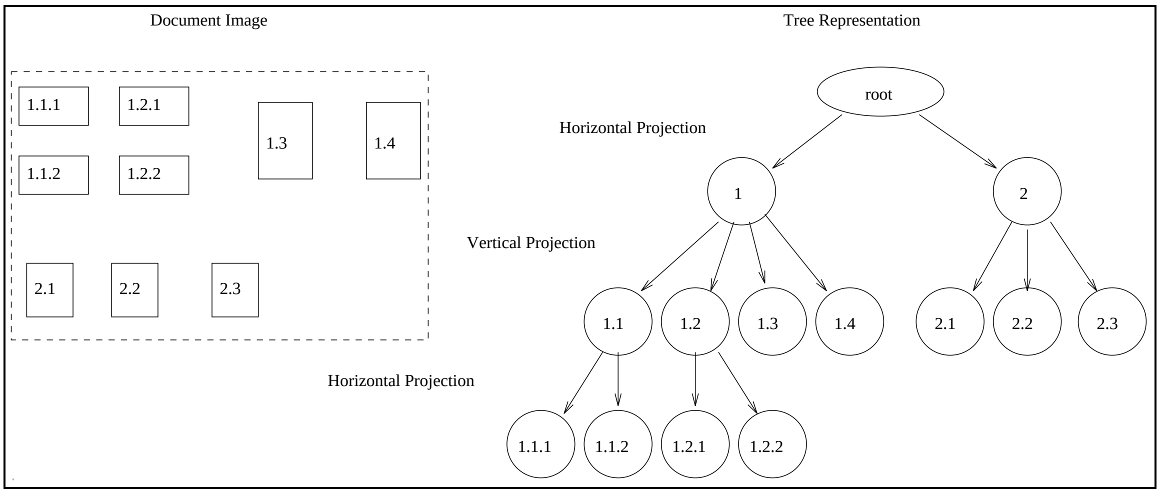


Figure 10: A document and its tree representation.

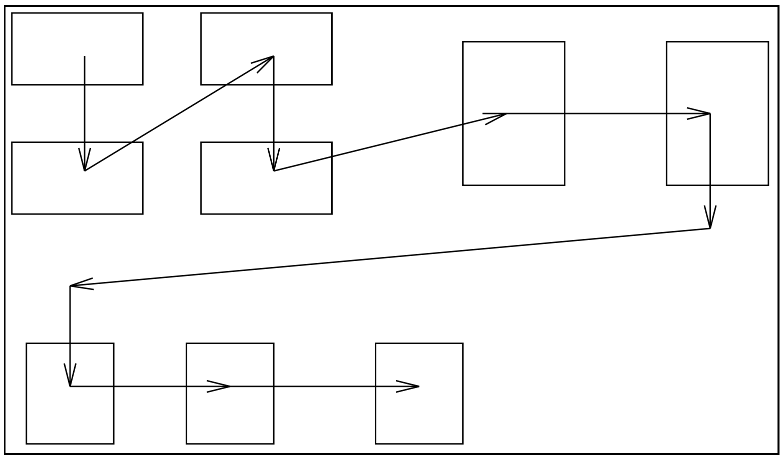


Figure 11: Illustration of the X-Y-tree's order.

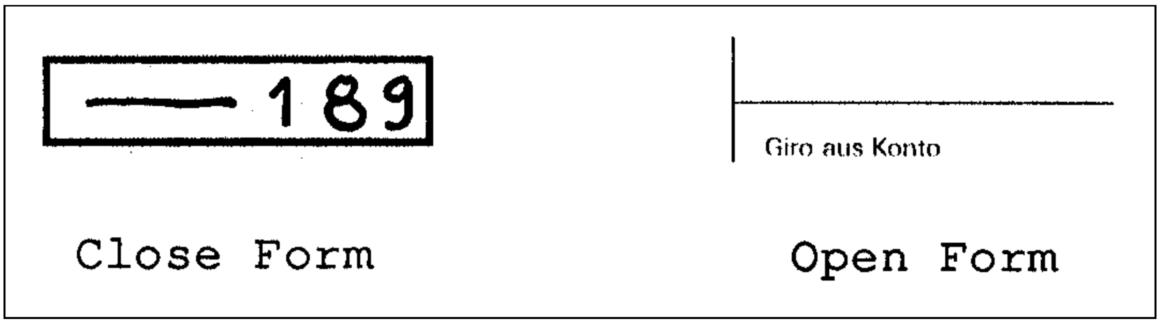


Figure 12: Typical forms that cannot be decomposed by the X-Y-tree.

ence of some particular graphics forms. Fig. 12 depicts two typical forms of graphics, close-form and open-form, that prevent the algorithm from succesful decomposition of the image. In the presence of such graphics forms in the document, it is necessary to localise them and apply the connected component analysis [Casey 90]. Another solution consists of localising these forms and applying the X-Y-tree only to the regions delimited by them.

3.2.2 Optimisation

In terms of computation time, the X-Y-tree decomposition is known to be faster than the connected component analysis [Casey 90]. Our experiments show that the computation of the basic X-Y-tree is four to eight times faster than the connected component analysis, depending on the structure and the contents of the document. The basic X-Y-tree is our implementation and the connected component labeling algorithm is from the SPIDER package [Spider 83]. In order to speed the basic X-Y-tree decomposition algorithm further up, two improvements have been added.

The first improvement is based on the observation that in the basic X-Y-tree algorithm, the projection profile is usually analysed by comparing it with a fixed threshold to separate text from blank bands. For a noiseless document, this threshold is usually very low (one or two). This suggests that it is sufficient to perform the projections only up to the limit of the predefined threshold, i.e., we stop counting the black pixels as soon as their accumulated number reaches the threshold. This is a *lossless optimisation* because the contribution of those pixels above the threshold is in any case not used by the basic algorithm.

Of course, this modified projection procedure is applicable for both horizontal and vertical directions. It turns out that this very simple idea saves about 50% of the computation time with respect to the basic algorithm. The exact reduction rate depends on the structure and the contents of the document. The denser the document is the higher is the resulting reduction rate. Indeed, for a totally white document, the reduction rate is 0% and for a totally black document, the reduction rate is nearly 100%.

The second idea to speed up the basic algorithm is a kind of sub-sampling of the projection profile where we skip n channels in the blank bands. Of course, the horizontal and vertical lines that have less than $(n+1)$ pixels width may be missed, but it is arguable (for small n) to say that a well scanned document should not have such elements or they are considered as noise. For the text bands, we keep the normal pace to avoid the situation in which two adjacent text bands are separated by a blank band having a width of only n pixels or less. This situation is often encountered in the vertical projection to separate consecutive characters. This is a *lossy optimisation* because of the risk of missing the horizontal and vertical lines that have less than $(n+1)$ pixels width.

The reduction rate of the computational complexity in comparison to the basic X-Y-tree algorithm has an upper bound of $n/(n+1)$. This maximal rate is obtained for a blank document, whereas for a non-blank document the rate necessarily decreases because of the presence of text bands. Therefore the reduction rate varies inversely with the density of the text in the document. Our experiments on 'normal' texts yield about 20% reduction rate for $n = 1$ (skip every other channel).

In summary, with respect to the basic X-Y-tree algorithm, the conditional projection yields a reduction rate proportional to the density of the text and the line skipping inversely proportional. Put together, a typical reduction rate of about 60% is obtained when $n = 1$ in the line skipping scheme.

3.2.3 Ordering problem

The spatial configuration in Fig. 10 is seldomly encountered in 'normal' texts. However, in our application, GIRO checks do have this configuration and its presence complicates the graph matching algorithm. Therefore we propose an algorithm to reorder the atomic entities provided by X-Y-tree decomposition before submitting them to graph matching. First, we consider the ordering properties from a theoretical point of view and then develop an algorithm for reordering the atomic entities.

By X-Y-tree order, we mean the order in which the atomic entities are reached by the X-Y-tree decomposition. Let us consider the sequence, or string, $s_i, i = 1..N$ of zones obtained by the X-Y-tree decomposition. Each atomic entity s_i is delimited by a rectangle $R(s_i)$ whose edges are either horizontal or vertical. It is obvious that

$$R(s_i) \wedge R(s_j) = \oslash; \forall i, j = 1..N, i \neq j \tag{1}$$

Otherwise, s_i and s_j would have been considered as a single atomic entity. We write $a < b$ iff the atomic entity a is reached before b by the X-Y-tree decomposition, or equivalently $b > a$. Let us define the lower domain $D_l(a)$ of an atomic entity a by

$$D_l(a) = \{(x, y) \in R^2 / \forall (x_a, y_a) \in R(a), (x > x_a) or (y > y_a)\} \tag{2}$$

Fig. 13 illustrates the definition of $D_l(a)$. By construction of the X-Y-tree, it follows that (see Appendix A):

$$R(a) \not\subset D_l(b) \Rightarrow a < b \tag{3}$$

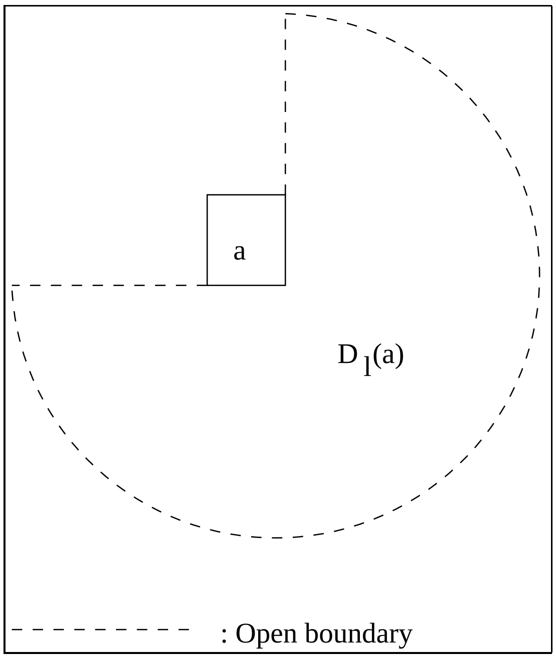


Figure 13: Definition of the Lower Domain of an atomic entity.

or equivalently

$$a > b \Rightarrow R(a) \subset D_l(b) \tag{4}$$

The last two formulas link the X-Y-tree's order to the spatial relationship between any two atomic entities. Given a list of atomic entities and any two entities a and b such that $a < b$, Formula (4) implies that $R(b) \subset D_l(a)$. Fig. 14 shows the 11 possible spatial configurations between a and b. Thus, by using a binary decision tree, the maximum number of comparison tests to classify b with respect to a is four.

The proof in Appendix A also shows that Formulas (3) and (4) remain valid if the algorithm starts with a vertical instead of a horizontal projection. This property is interesting for processing documents in Chinese and Japanese where it may be preferable to project the image in the vertical direction first.

These properties have been used to derive an efficient reordering algorithm. This algorithm consists of breaking down the links between atomic entities that are not in 'natural' order and then grouping them again in 'natural' order. The algorithm will be illustrated through a typical example where the reordering is necessary for further processings like string matching.

Let us consider the characters in Fig. 15. It is easy to see that the X-Y-Tree decomposition yields the result in the following order:

- c a d b e F G H

whereas a human reader would prefer the order 'a b c d e F G H'.

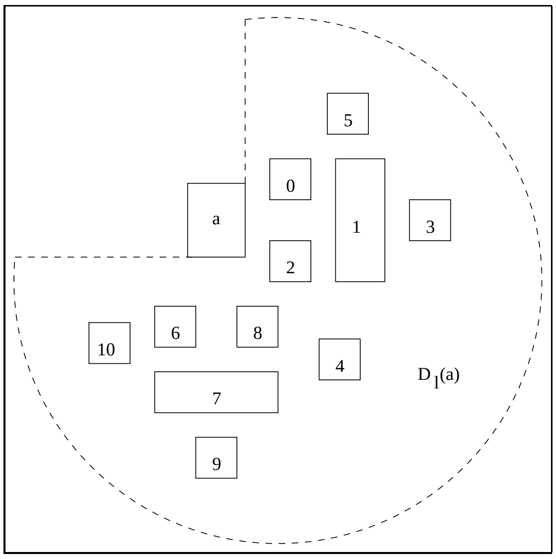


Figure 14: Possible spatial configurations for $b > a$.

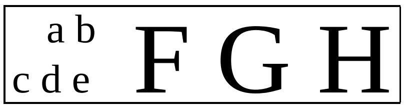


Figure 15: Text example for illustration of the reordering algorithm

The reordering algorithm consists of three steps. In the first step, characters that have already appeared in 'natural' order will be grouped together whereas those that are in 'wrong' order will be broken apart. This can be done by considering all pairs of consecutive characters. For each pair, if the second character lies to the right of the first and they are on the same line (up to some threshold) then they are grouped into the same string; otherwise the second character becomes the first character of a new string. The result of this step is a list of strings:

- c
- a
- d
- b
- e F G H

In the second step, strings that are in 'natural' order will be concatenated. Two strings are in 'natural' order iff the first character of the second string lies to the right of the last character of the first string and these two characters on the same line (up to some threshold). The result of this step is:

- c d e F G H

- a b

In the last step, strings are reordered. Two strings have to be exchanged if any character of the second string lies above some character of the first string. The final result is:

- a b
- c d e F G H

The properties implied by Formulas (3) and (4) have been implicitly used in the first step of the algorithm. In this step, the grouping of characters already in 'natural' order is performed in $\mathcal{O}(n)$ operations whereas it would have required $\mathcal{O}(n^2)$ or at least $\mathcal{O}(nlog(n))$ steps if pairs of characters that are in 'natural' order had not been produced in the correct order by the segmentation algorithm (for example, by the connected components analysis algorithm). The reduction is due to the fact that we consider only pairs of consecutive characters provided by the X-Y-Tree decomposition (there are $n-1$ pairs in a list of n characters). Now, we will show that the first step truly groups together characters already in 'natural' order. Let us consider a pair (a, b) where b immediately follows a. It is clear that $b > a$ which implies (Formula (4)) that b belongs to $D_l(a)$ and takes one of the eleven possible relative positions with respect to a as shown in Fig. 14. Moreover, if a and b are more or less on the same line (position 0,1,2 or 3) then (a, b) is in 'natural' order (there can not be anything else in between because b would not have immediately followed a).

In the second and third steps, such reasoning is not valid anymore because we are dealing with strings instead of characters. However, the time complexity is limited because the number of strings is usually much smaller than that of characters.

4 HIGH LEVEL PROCESSING

High-level processing comprises two modules: graph matching and OCR engine. We will first describe the graph matching algorithm (Section 4.1) and then the OCR engine together with some post-processings (Section 4.2).

4.1 Graph Matching

In general, graph matching consists of establishing a correspondence between the model graph and the data graph. In our application, the model graph is represented in Fig. 3 whereas the data graph is represented by the reordered list of atomic entities. (Since each element of the list contains information about its position, the relative positions - binary relations of the data graph - between elements can be computed whenever the needs arise.) More specifically, the correspondence establishment refers to the assignment to each node of the model graph a subset of atomic entities. All subsets are disjoint, i.e., each atomic entity corresponds to at most one node of the model graph. Those atomic entities that do not have a node correspond to data lying outside of the region of interest (Fig. 1). For each node of the model graph, the assignment process will be called *identification*. As mentioned in Section 2, the graph matching algorithm in our application is restricted to the identification of GRAPHICS and TITLE objects, leaving out the INFORMATION objects. The algorithm performs as follows:

1. Identify GRAPHICS objects (vertical and horizontal lines as well as boxes).
2. Eliminate those atomic entities that lie outside of the region of interest (see Fig. 1), i.e., to the right of the vertical line or above the horizontal line found in step 1.
3. Identify the following TITLE objects:

- Einzahlung für/Versement pour/Versamento per
- Zugunsten von/En faveur de/A favore di
- Konto/Compte/Conto
- Fr. c.

4. Identify INFORMATION objects according to the following rules:

- Financial Institution: all atomic entities lying between the first and second TITLE objects.
- Receiver: all atomic entities lying between the second and the third TITLE objects.
- Account Number: all atomic entities lying to the right of the third TITLE object and above the fourth TITLE object.

The identification of GRAPHICS objects is quite straightforward. After segmentation, horizontal and vertical lines as well as boxes are identified by looking for atomic entities that satisfy some size constraints. The elimination of atomic entities that are out of the region of interest is based on tests of their position with respect to the lines identified as GRAPHICS objects. TITLE objects in this application are predefined strings of characters. Their identification (or localisation) will make use of all knowledge explicitly and implicitly provided by the PTT. We will only consider the first three of them since the fourth one has fixed spatial relations to the boxes (see Fig. 2) and can be easily identified by looking for atomic enties that lie just above the boxes. The next two sub-sections will describe TITLE objects knowledge and how to use it.

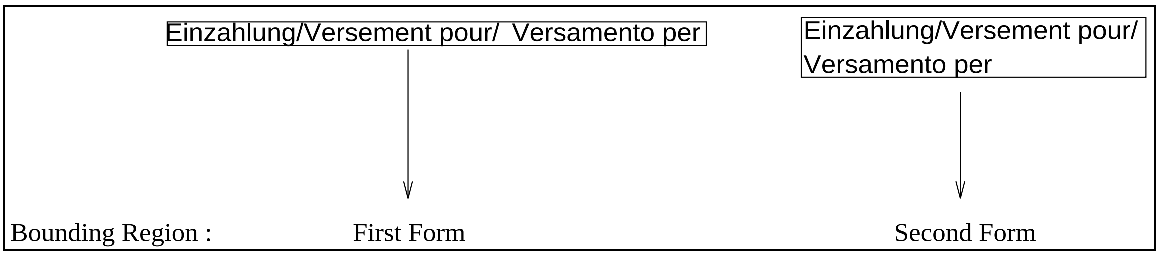

Einzahlung/Versement pour/ Versamento per
Einzahlung/Versement pour/ Versamento per
Bounding Region :
First Form
Second Form

Figure 16: Two possible forms of the bounding region of the first TITLE object.

4.1.1 TITLE Objects Knowledge

In our application, the first three TITLE objects have the same basic properties and differ from each other only by their specific values. For instance, all three have fixed contents but each has its own string of characters. Therefore, we will take only the first one as an example and explain it in detail.

The TITLE object knowledge is divided into two parts: structural and statistical. Structural knowledge comprises the size of characters, the total length of the string and various possible forms of the rectangular bounding region (see Fig. 16) Statistical knowledge characterises information contained in the string of characters. More specifically, we will use the histogram values of the vertical projection of black pixels (also called vertical profile) as statistical features (Fig. 17). In order to make the vertical profile invariant with respect to the variation of inter-character spaces, we eliminate these spaces before histogram computation, see Fig. 17. Since the font of the characters does not change from one check to another, the form of the vertical profile remains the same. Their size is also fixed and thus no normalisation is needed. Note that instead of the vertical profile, we could have used the string of characters as feature, the OCR engine for character recognition and performed a string comparison. However, after some preliminary tests, we found that this approach is very time consuming and decided to use it only if the vertical profile is not sufficient. Fortunately, it turned out that (see Section 5) the vertical profile is sufficiently discriminant for our application.

In order to deal with small variations and noise, each structural feature is considered with a tolerance value, usually equal to 10 or 20% of the absolute value. As for the statistical feature, i.e., vertical profile, the tolerance is expressed by the variance of the distance (considered as a random variable) between an actual profile and the reference profile. We have used the Euclidean distance and found that it was sufficiently discriminant for our purpose.

Finally, let us note that the structural features may be directly obtained from the specifications of the PTT whereas the statistical features (reference profile) is extracted through a training phase. The reference profile is defined as the arithmetic average of a set of training profiles.

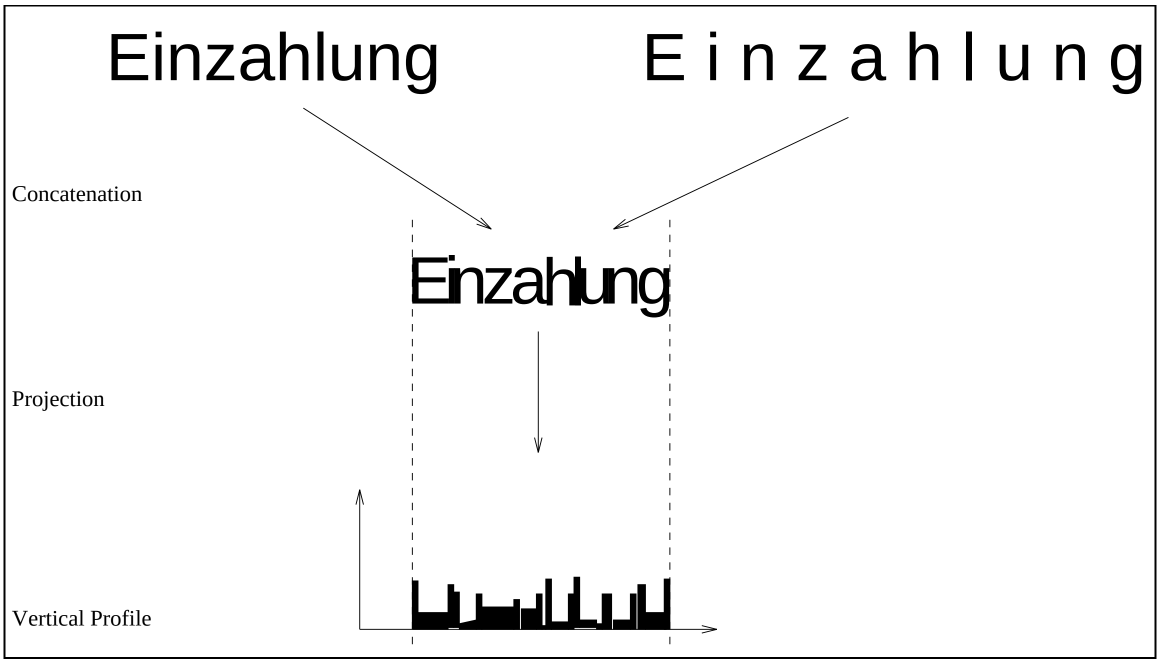


Figure 17: Computation of the vertical profile of a string of characters.

4.1.2 TITLE Objects Identification

In Section 4.1.1, we presented the TITLE object knowledge which consists of a structural and a statistical part. Next, we will propose an approach which combines the two parts in a hybrid algorithm. This algorithm consists of identifying a TITLE object in the reordered list of atomic entities provided by the segmentation module. In order to appropriately use structural and statistical knowledge, we will first study their relationships and then derive an efficient algorithm.

Let us notice that the two parts of the knowledge are not independent from each other. Indeed, a pattern that does not satisfy the structural constraints is unlikely to satisfy the statistical ones. Conversely, a pattern that satisfies the former is more likely to also satify the latter although there exist practical cases where it does not. In our application, we require that a pattern (set of atomic entities) is accepted as a TITLE object if and only if it satifies both structural and statistical constraints. By definition, the statistical features can be extracted and tested only if the location of each atomic entity of the set is known. As location is part of the structural knowledge, we first perform the structural test. The hybrid identification algorithm is straighforward (see Fig. 18). It contains two tests which will now be described in more detail.

The *test_structure_is_correct* comprises two parts, namely, search and testing of the bounding region (Fig. 19). The first part in turn contains two procedures which work as follows. The procedure *update total_length* adds the length of the new entity to *total_length* and *update bounding_box* extends the sizes of the *bounding_box* to include the new entity. In the first part, the algorithm receives an entity which has

```
detected = FALSE;
FOR i=1 TO number_of_atomic_entities DO BEGIN
        /* Hypothesise that the current entity is the
        starting character of the searched object */
        IF test_structure_is_correct THEN
            IF test_statistics_is_correct THEN BEGIN
                detected = TRUE;
                BREAK;
            END;
END;
IF detected THEN output_the_location;
```

Figure 18: Hybrid algorithm.

```
/* Search a string of characters up to the object length */
total_length = 0;
bounding_box = bounding_box_of_the_starting_entity;
WHILE (total_length < minimal_length) AND NOT(end_of_list) DO
            IF (the_next_entity satisfies size_constraints) THEN BEGIN
                update total_length;
                update bounding_box;
            END;
/* Test the form of the bounding_box */
structure_is_correct = FALSE;
IF (bounding_box satisfies form_1) OR
    (bounding_box satisfies form_2) THEN BEGIN
                structure_is_correct = TRUE;
                output_the_location;
END;
```

Figure 19: Test_structure_is_correct algorithm.

```
statistics_is_correct = FALSE;
compute vertical_profile;
compute distance(current_profile, reference_profile);
IF (distance < threshold) THEN
    statistics_is_correct = TRUE;
```

Figure 20: Test_statistics_is_correct algorithm.

been hypothesised as the starting character of the searched object. The algorithm successively tests the entities that follow, up to the minimal length of the searched object. It is clear that if the hypothesis is correct, the starting atomic entity and those following it correspond to the characters in the searched object, and the first part of the algorithm stops before the end of the list at an approximately correct place. The resulting *bounding_box* is then a good approximation of the correct one. Note that this part is independent of which of the two forms in Fig. 16 the actual object takes because they are equivalent after reordering. In the second part, the form of *bounding_box* is checked. Its conformity with either one of the two possible forms is sufficient to validate the hypothesis as structurally correct.

The *test_statistics_is_correct* on the other hand receives a list of atomic entities that already satisfy the structural test (Fig. 20). It computes, by concatenation and projection, the vertical profile and the distance to the reference profile. The test is considered as correct if the distance is smaller than a predefined threshold.

Note that as it is presented, the hybrid algorithm will accept the first occurence of a pattern that satisfies both kinds of constraints and stop. This limitation has proved sufficient for our application. A more general algorithm could be easily derived by testing all hypotheses that satisfy the constraints and put them in a list of potential candidates together with a confidence level.

The proposed algorithm is very efficient because of the *nested structure* of the two tests. The *test_structure_is_correct* procedure is much faster than the *test_statistics_is_correct* one since the former involves only the testing of the sizes of atomic entities whereas the latter requires the computation of the vertical profile prior to the statistical test. Therefore, the majority of candidates are ruled out by the faster test and only few of them have to go through the slower test.

In order to derive the computational complexity of the matching algorithm, let us notice that the bulk of computation burden is concentrated in the identification of TITLE objects (Step 3 of the matching algorithm) and define the following quantities:

- N_t = number of TITLE objects in the model graph.
- P = number of atomic entities produced by X-Y-tree decomposition.
- M = length of the longest string among TITLE objects.

- S = maximum area of an atomic entity (in pixels).

For simplicity purpose, we will consider only the worst case. The counting unit is *operation* which is bounded by a small constant integer (less than ten). For instance, the test of whether the height of an atomic entity is within some interval requires one substraction to compute the height and two comparisons with the minimal and maximal values, but is counted as only one operation. The identification of N_t TITLE objects requires N_t applications of the hybrid algorithm. Each application forms at most P hypotheses (Fig. 18). For each hypothesis, one *test_structure_is_correct* and one *test_statistics_is_correct* are performed. Thus the computational complexity is $N_tP(\#operations\_in\_test\_structure\_is\_correct) + N_tP(\#operations\_in\_test\_statistics\_is\_correct)$. In the *test_structure_is_correct* procedure, the search part requires at most M operations whereas the test of bounding region is negligible. In the *test_statistics_is_correct* procedure, there are two parts, namely, the computation of vertical projection and of the distance. For vertical projection, the complexity is proportional to the total area of all atomic entities and is performed only once for each. It requires PS operations for the whole matching algorithm and thus is not multiplied by N_tP. The computation of the distance requires $M\sqrt{S}$ operations where $\sqrt{S}$ represents the average number of channels in the vertical profile of an atomic entity. The total computational complexity is $(N_tPM) + (PS + N_tPM\sqrt{S}) = P(N_tM(\sqrt{S}+1) + S)$. Typical values in our application are $P = 200, N_t = 3, M = 50$ and $S = 400$ at the resolution of 200 dots per inch (dpi) which yield 710000 operations. For large values, the time complexity of the matching algorithm is $\mathcal{O}(N_tPM\sqrt{S})$.

4.2 OCR and Post-Processing

OCR is one of the most important aspects in document image analysis and understanding. It has been studied since many decades and nowadays commercial products are widely available at reasonable prices. In the context of our application (Fig. 4), the OCR engine is a module which reveives an input binary bitmap of a text region and outputs a stream of characters. The post-processing refers to the correction (mostly substitution) brought by the specific knowledge of the application and does not interfere with the OCR engine at all.

4.2.1 OCR Engine

The OCR engine used in this application is a commercial product provided by Xerox Imaging Systems. To our knowledge, this product is the best *open* software package on the market at the time of purchase. Here, we will present the main features of the package.

- Omnifont: the system is able to recognise characters independently of any specific machine-printed font (fancy fonts are excluded). This property is perfectly suited to our application since we are not interested in font recognition.

- Separated multilingual capabilities: there are ten language packs, all European. The weak point is that the system does not accept any mixture of languages for the time being. However, we expect that Xerox will remedy this problem soon.

- Training capability: apart from the language packs, the system can be trained with new characters or symbols. However, the recognition is specific to the trained resolution, i.e., the same symbol at another resolution will not be recognised.

- Possibility to emphasize on 'alpha', 'numeric' or 'alphanumeric' modes: this feature may help to resolve certain ambiguities like between an 'l' (el) and a '1' (one). In our application, the regions corresponding to the Institution and Receiver INFORMATION objects are fed into the OCR engine in the alphanumeric mode and the region corresponding to the Account Number in numeric mode. It is important to understand that the mode setting helps resolving ambiguities. It does not exclude the other symbols.

4.2.2 Post-Processing

At the present time, the post-processing in our application aims at correcting substitution errors of the OCR engine. The correction is based solely on the expected results. In the Account Number, we know that there can only be numerals and therefore the following corrections have been included:

- $'?' \rightarrow' 7'$.

- $'g' \rightarrow' 9'$.

which constitute the most often encountered errors. As for the other INFORMATION objects, there is no evident correction in view.

Of course, post-processings could include more sophisticated operations. For instance, we can think of a verification phase in which the name of the receiver is searched through a database. If any inconsistencies arise, the system could go back to the segmentation phase and restart the analysis of the regions of interest by using some modified scheme.

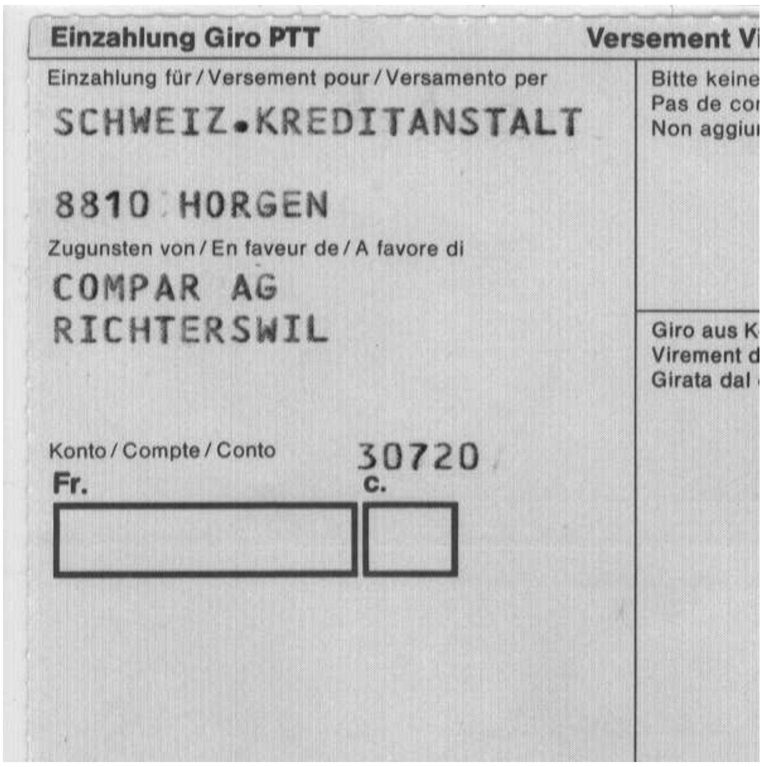
5 RESULTS

In this section, all the processing steps will be presented for one typical GIRO form. Starting with the input gray-level image (Fig. 21), the global binarisation is performed (Fig. 22). The binary image is then segmented using the X-Y-Tree decomposition algorithm yielding a list of atomic entities (Fig. 23). The graph matching algorithm identifies all the present objects, in particular the INFORMATION ones (Fig. 24) whose bounding regions are fed back to the binarisation procedure. These regions are separately binarised (Fig. 25) again and the binary bitmaps are input to the OCR engine. The final results are three streams of ASCII characters (Fig. 26).

We plan to test the system at various resolutions, namely, 200, 300 and 400 dots per inch (dpi). According to the size of characters in our documents, we think that 200 dpi is the lowest possible resolution. On the other hand, a resolution higher than 400 dpi does not provide any additional information. So far, we have experimented with the lowest resolution and the results are as follows.

The size of left part of GIRO check forms is 3 x 3 inches. The computation time on a Sun SparcStation 2 (30 MIPS) is about 0.2 second for the first phase and 0.7 to 2.0 seconds for the second phase depending on the contents of the GIRO form. The most time consuming part is the OCR engine which operates at an average rate of 100 characters per second. If a faster version is to be implemented, there are two possible solutions. The first is to use another OCR engine with dedicated hardware and the second, simpler solution consists of using a faster workstation.

As for the overall recognition rate, tests on 40 checks show that segmentation and graph matching algorithms perform with no error. Note that the reference profiles of TITLE objects are trained and tested using the same set of samples. In the future, we should use a test set that contains also samples not belonging to the training set. There are approximately 2300 characters on these 40 checks and the correct recognition rate is 96%. At higher resolutions, it is reasonable to expect a higher correct recognition rate but also more computation time. The question of how the results vary versus resolution will be investigated in the near future.



Einzahlung Giro PTT **Versement Vi**

Einzahlung für / Versement pour / Versamento per

SCHWEIZ.KREDITANSTALT

8810 HORGEN

Zugunsten von / En faveur de / A favore di

COMPAR AG
RICHTERSWIL

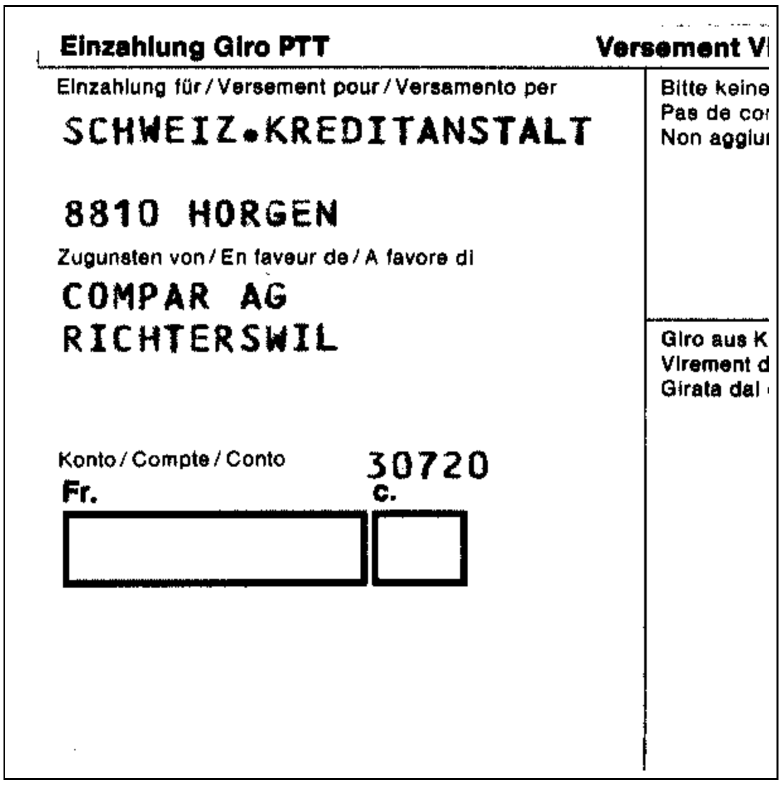
Konto / Compte / Conto 30720

Fr. **c.**

Bitte keine
Pas de co
Non aggiu

Giro aus K
Virement d
Girata dal

Figure 21: Input image.


Einzahlung Giro PTT

Versement Vi

Einzahlung für / Versement pour / Versamento per

SCHWEIZ•KREDITANSTALT

8810 HORGEN

Zugunsten von / En faveur de / A favore di

COMPAR AG
RICHTERSWIL

Konto / Compte / Conto 30720

Fr. c.

Bitte keine
Pas de coi
Non agglui

Giro aus K
Virement d
Girata dal

Figure 22: Globally binarised image.

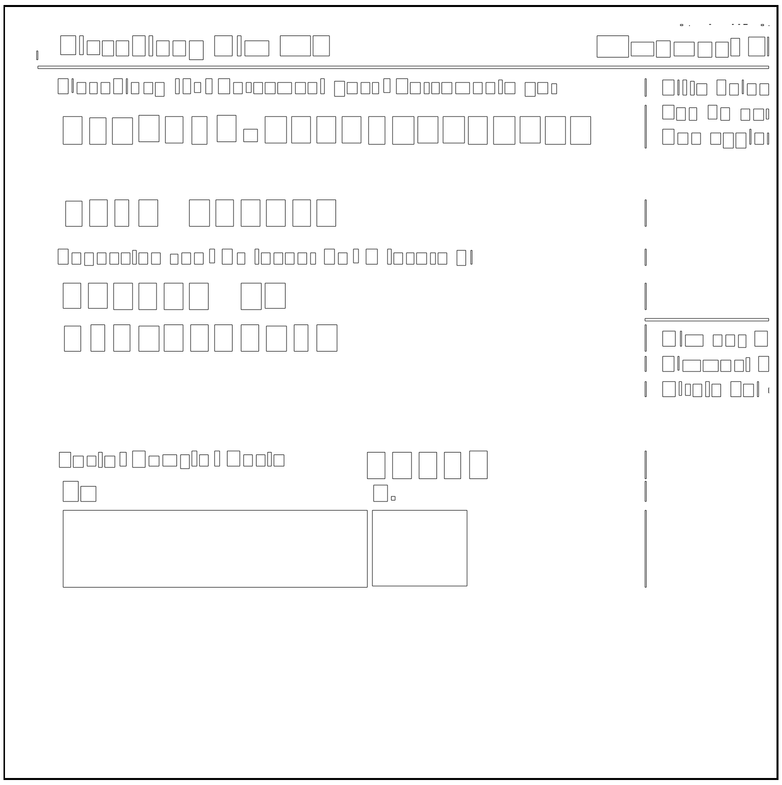


Figure 23: Result of the segmentation.

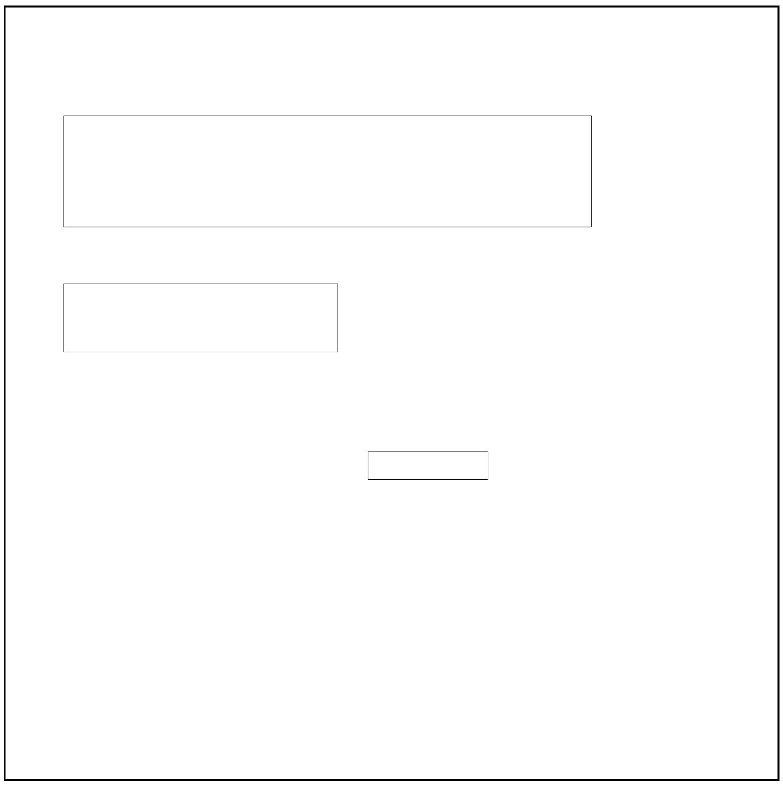
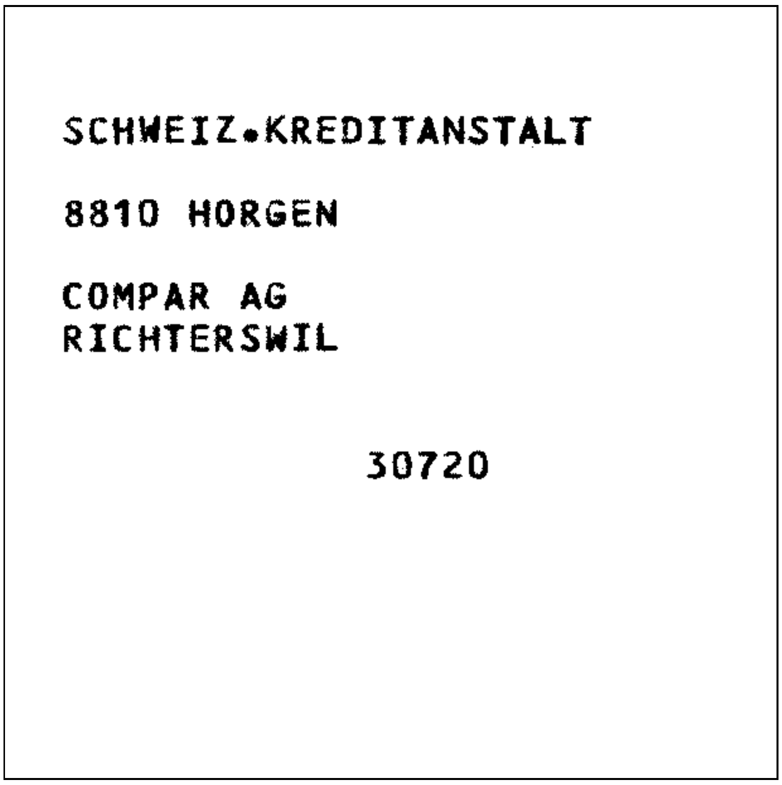


Figure 24: Regions corresponding to INFORMATION objects.



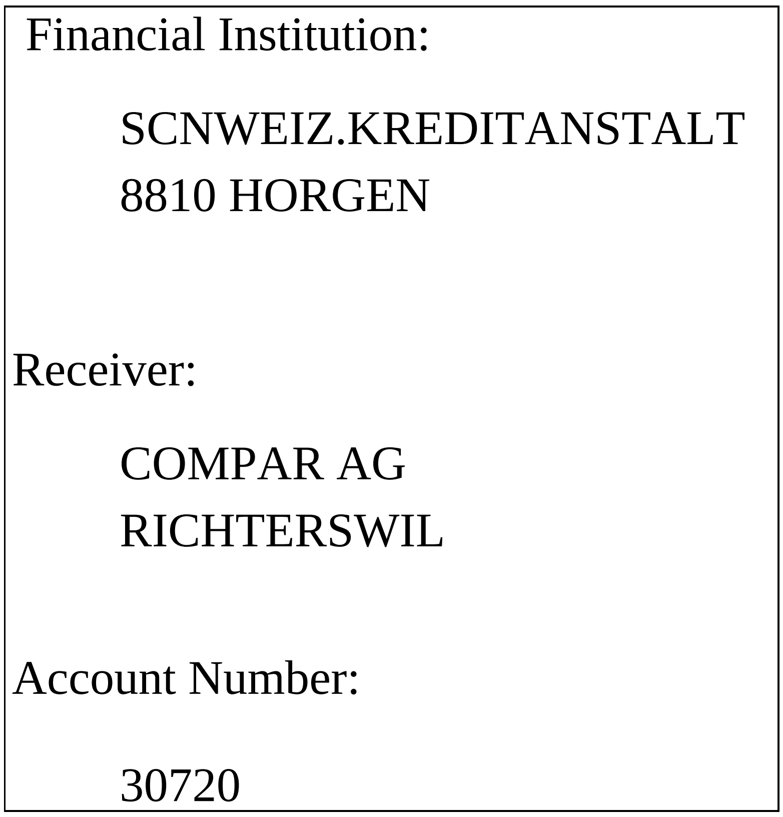
SCHWEIZ.KREDITANSTALT

8810 HORGEN

COMPAR AG
RICHTERSWIL

30720

Figure 25: Locally binarised image.



Financial Institution:

SCNWEIZ.KREDITANSTALT
8810 HORGEN

Receiver:

COMPAR AG
RICHTERSWIL

Account Number:

30720

Figure 26: Output streams of characters.

6 FUTURE RESEARCH AND CONCLUSIONS

In this report, we have presented a system that can recognise various parts of a GIRO check form. The system comprises two modules: low- and high-level. The low-level module performs automatic binarisation and segmentation based on X-Y-Tree decomposition. The high-level module contains a graph matching procedure and the OCR engine. The particular connection between different procedures yields a very fast and accurate system.

To complete the investigation on GIRO checks recognition, two additional problems should be addressed in the future. First, hand-written numerals recognition should be considered. Indeed, these entities appear quite often in real documents and have been so far voluntarily ignored (see Fig. 2). Scientific research on this problem has been intensive since at least two decades and a recognition rate as high as 95% has been reported lately [Nishida 92]. Second, more sophisticated post-processing techniques should be added to the present version of the system, including the search of the name and address in a database to confirm the recognition result.

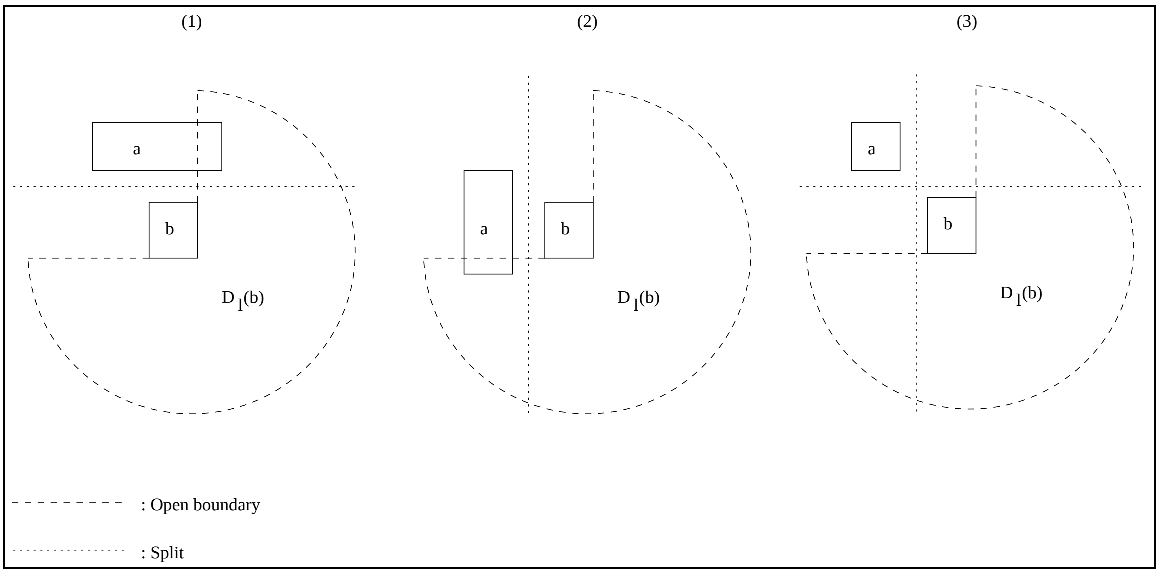


Figure 27: Relative positions between two atomic entities.

A Proof of The Ordering Formula

Let us consider any two atomic entities a and b resulting from the X-Y-tree decomposition such that $R(a) \not\subset D_l(b)$, obviously we can distinguish 3 possible spatial configurations as shown in Fig. 27.

Since a and b are two distinct atomic entities, there must be exactly one projection that gives rise to their separation during the decomposition. It is obvious that in

- Case 1: it must be a horizontal projection and since the analysis order is downward, a is reached before b.
- Case 2: it must be a vertical projection and since the analysis order is left-to-right, a is reached before b.
- Case 3: it can either be a horizontal or a vertical projection, but in both cases, due to the corresponding analysis order, a is reached before b.

Thus, in any case, a is reached before b, i.e., $a < b$.

Moreover, it can be seen that the proof does not assume which projection (horizontal or vertical) the algorithm starts with. Therefore, Formula (3) remains valid in the case the algorithm starts with a vertical projection.

References

[Baird 87] Baird H.S., ”The Skew Angle of Printed Documents”, Proc. Conf. of the Society of Photographic Scientists and Engineers, Rochester, New York, May 20-21, 1987.

[Baird 90] Baird H.S., "Anatomy of A Page Reader", Proc. of the Workshop on Machine Vision Applications, Tokyo, Nov. 28-30, pp. 483-486, 1990.

[Casey 90] Casey R.G. and Wong K.Y., "Document-Analysis, Systems and Techniques", in *Image Analysis and Applications*, R. Kasturi and M. Trivedi (Ed.), Marcel Dekker, New York, pp. 1-35, 1990.

[Kasturi 90a] Kasturi R., Bow S.T., El-Masri W., Shah J., Gattiger J.R. and Mokate U.B., "A System for Interpretation of Line Drawings", IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 12, No. 10, Oct. 1990, pp. 978-991, 1990

[Kasturi 90b] Kasturi R., Raman R., Chennubhotla C. and O'Gorman L., "Document Image Analysis: An Overview of Techniques for Graphics Recognition", Pre-Proceedings of the Workshop on Syntactic and Structural Pattern Recognition, Murray Hill, New Jersey, U.S.A., June 13-15, pp. 192-230, 1990.

[Nagy 84] Nagy G. and Seth S., "Hierarchical Representation of Optically Scanned Documents", Proceedings of the 7th Int. Conf. on Pattern Recognition, Montreal, pp. 347-349, July 1984.

[Nagy 90a] Nagy G., "Document Analysis and Optical Character Recognition", in *Progress in Image Analysis and Processing*, edited by Cantoni V., Cordella L.P. et al., World Scientific, pp. 511-529, 1990.

[Nagy 90b] Nagy G., "Towards A Structured-Document-Image Utility", Pre-Proceedings of the Workshop on Syntactic and Structural Pattern Recognition, Murray Hill, New Jersey, U.S.A., June 13-15, pp. 293-309, 1990.

[Nagy 91] Nagy G. and Viswanathan, "Dual Representation of Segmented Technical Documents", First Int. Conf. on Document Analysis and Recognition, St.-Malo, France, Sept. 30 - Oct. 2, pp. 141-151, 1991

[Nishida 92] Nishida H. and Mori S., "A Model-Based Split-and-Merge Method for Recognition and Segmentation of Character Strings", Proceedings of the Workshop on Syntactic and Structural Pattern Recognition, Berne, Switzerland, Aug. 26-28, 1992.

[Okamoto 90] Okamoto M. and Miyazawa A., "An Experimental Implementation of Document Recognition System for Papers Containing Mathematical Expressions", Pre-Proceedings of the Workshop on

Syntactic and Structural Pattern Recognition, Murray Hill, New Jersey, U.S.A., June 13-15, pp. 335-350, 1990.

[Otsu 79] Otsu N., "A Threshold Selection Method from Gray-Level Histogram", IEEE Trans. on SMC-9, pp. 62-66, Jan. 1979.

[Pavlidis 91] Pavlidis T. and Zhou J., "Page Segmentation by White Streams", First Int. Conf. on Document Analysis and Recognition, St.-Malo, France, Sept. 30 - Oct. 2, pp. 945-953, 1991.

[Rao 90] Rao N.S.V., "Computational Aspects of Similarity Measures Based on Linearization of Lattice Points", Pre-Proceedings of the Workshop on Syntactic and Structural Pattern Recognition, Murray Hill, New Jersey, U.S.A., June 13-15, pp. 351-365, 1990.

[Spider 83] *Spider Users' Manual*, AIST MITI Japan, 1983.

[Srihari 86] Srihari S.N. and Zack G.W., "Document Image Analysis", Proceedings of the 8th International Conference on Pattern Recognition, Paris, France, Oct. 27-31, pp. 434-436, 1986.

[Viswanathan 90] Viswanathan M., "Analysis of Scanned Documents - A Syntactic Approach", Pre-Proceedings of the Workshop on Syntactic and Structural Pattern Recognition, Murray Hill, New Jersey, U.S.A., June 13-15, pp. 450-459, 1990.