*Pattern Recognition, Vol 31, No. 9, pp. 1191--1204, September 1998*

# Pink Panther: A Complete Environment for Ground-truthing and Benchmarking Document Page Segmentation [1]

Berrin A. Yanikoglu [a] and Luc Vincent [b]

[a] *IBM Almaden Research Center, 650 Harry Road, San Jose, CA 95120*
*berrin@almaden.ibm.com*

[b] *Xerox Desktop Document Systems, 3400 Hillview Avenue, Palo Alto, CA 94304*
*lucv@adoc.xerox.com*

We describe a new approach for the automatic evaluation of document page segmentation algorithms. Unlike techniques that rely on OCR output, our method is region-based: segmentation quality is assessed by comparing the segmentation output, described as a set of regions, to the corresponding ground-truth. Error maps are used to keep track of all the errors associated with each pixel, regardless of the document complexity. Misclassifications, splitting, and merging of regions are among the errors detected by the system. Each error can be weighted individually and the system can be customized to benchmark virtually any type of segmentation task.

## 1 Introduction

Page segmentation is the process by which a document page image is decomposed into its structural and logical units (*regions* or *zones*), such as images, paragraphs, headlines, tables, etc. This process, also referred to as *zoning*, *layout analysis*, or *page decomposition*, is critical for a variety of document image analysis applications. Once the regions are found (*zoning*) and their types identified (*labeling*), they may also need to be ordered (*ordering*) according to the natural reading order(s) of the page. Region ordering is sometimes considered part of page segmentation itself.

Document recognition systems rely heavily on page segmentation in order to “understand” the structure of a document and ultimately be able to *recompose* it in a word processing environment. Segmentation provides the necessary coarse-level understanding of the document: discriminating between text and graphics, separating a column of text from an adjacent one etc. Furthermore, reading machines for the blind need accurate page segmentation to

[1] This work was done when the authors were at Xerox Desktop Document Systems, Peabody, MA.

be able to read the text in multi-column documents in a correct order. Even digital copiers and related software/hardware systems rely on document segmentation techniques to correctly identify the various types of regions present on a page (e.g., text, continuous tone images, halftones, line-art), and render them appropriately[1].

Until recently, OCR accuracy was the only aspect of document recognition systems that was commonly benchmarked. However, nowadays sophisticated document recognition systems are handling very complex documents such as newspapers, magazines, and junk mail, and thus it is becoming increasingly important to be able to benchmark page segmentation as well. A segmentation benchmarking tool would be helpful in deciding between commercial systems for a given application, as well as enabling developers of page segmentation systems to easily test their algorithms.

Several different page segmentation algorithms have been developed in the past couple of decades, some described in literature [2] and some proprietary [3]. Among the many different approaches to the problem are rule-based systems[4], use of connected component bounding-boxes[5], and analysis of background and *white streams* information[6–8]. Some of these algorithms are specifically designed to work on particular types of document, whereas others are meant to be completely general. Furthermore, some algorithms are designed for very specific subtasks of what is generally referred to as page segmentation: for example, some may simply detect the graphics in a document page image, while others may only care about the text. Some algorithms generate coarse segmentations (e.g., at the galley level) while others decompose pages down to the paragraph level, or even down to the line level. For some applications, a page segmentation algorithm may need to distinguish between halftones and continuous tones, for others, the distinction is irrelevant.

Faced with such a diversity of methods and such a wide variety of goals, a question arises: how should the quality of a given segmentation algorithm for a particular segmentation task be assessed? The accuracy of a page segmentation system is typically evaluated by running the system on a large number of document images and "eyeballing" the results, a tedious and subjective process. Benchmarking of document page segmentation is further complicated by the various ways segmentation mistakes can be handled. For instance, if the segmentation mistakes will be corrected manually (i.e., re-zoning parts of the document), one may want to evaluate the segmentation performance in terms of how many regions need to be re-zoned to correct the segmentation. On the other hand, if the segmentation output is to be directly fed to the OCR system, one might be concerned with the amount of images that are classified as text or the amount of merged text columns etc.

In this paper we describe a complete

environment, called *Pink Panther*, for creating segmentation ground-truth files and benchmarking page segmentation algorithms. The performance of a given page segmentation system is evaluated by running the system on a large number of document images and comparing the output for each document to the corresponding, previously created ground-truth. We have developed a ground-truthing tool, called *GroundsKeeper*, for easily creating the ground-truth data, as explained in Section 3. Both *GroundsKeeper* and the benchmarking tool (*Cluzo*) can be tailored to a variety of ground-truthing and benchmarking needs, respectively.

The paper is organized as follows. In the next section, we describe the previous work in this area. Section 3 describes our ground-truthing methodology, the tool we developed to create segmentation ground-truth files, and the file format used to represent the ground-truth information. In Section 4, we describe the benchmarking process. Finally, we show how the system can be customized to a wide variety of applications.

## 2 Background and Previous Work

Two approaches have been proposed in literature for automatic benchmarking of page segmentation. The first one, developed at the University of Nevada in Las Vegas (UNLV), is purely text-based[9–11], whereas the approach we have chosen is region-based[12,13]. These two approaches are briefly described below, and we explain why we believe our region-based approach offers significant advantages over text-based approaches.

### *2.1 Text-Based Approaches*

The Information Science Research Institute (ISRI) of the University of Nevada in Las Vegas has designed an interesting technique to evaluate the quality of page segmentation algorithms working within OCR software packages. Briefly, this method works as follows:

First, page segmentation (including region ordering) *and* character recognition subsystems of an OCR system are applied to the document page, and the result is output as an ASCII string.

Then, using string matching algorithms[14], the number of insertions, deletions, and block moves necessary to convert this output into the ideal ground-truth string while minimizing the overall cost, are determined. The overall cost is computed as $C = n_i \times c_i + n_m \times c_m$, where $c_i$ is the cost of an elementary insertion operation, $c_m$ is the cost of a block move, and $n_i$ and $n_m$ are the number of insertions and moves, respectively. The cost of a deletion is set to zero (for reasons explained below).

Finally, in order to find the portion of the overall cost that is due to segmentation mistakes alone, the cost of errors that are purely due to character recognition mistakes are subtracted form the overall cost, $C$. This cost is determined by running the OCR system on the same page, this time using *manual* segmentation.

One of the main points in favor of this technique is that it is purely text-based, and therefore does not require the page segmentation subsystem to use any particular output format. In addition, although its underlying string-matching algorithms are rather elaborate, the overall approach is fairly straight-forward and the ground-truth files for use with this approach (text ground-truth) are very easy to create. Therefore, the UNLV zoning evaluation system has been well accepted by the document recognition community.

Nonetheless, this system has some limitations listed below:

– The system can only deal with text regions: the measured accuracy only reflects the accuracy on segmenting text zones, and the segmentation of document images containing no text cannot be evaluated.
– The output is merely a set of numbers (number of insertions, block moves, and perhaps deletions). It therefore provides very little information on the types of mistakes that were actually made (whether regions were split or merged, images mistaken for text, or headline regions segmented less correctly than regular text regions, etc.). Such information would be very valuable in improving segmentation algorithms.
– Only one zone order (or at most a small set of zone orders) is considered correct. If a zoning algorithm produces a perfect segmentation, but does not output regions in one of the few chosen orders, it is penalized. This is too rigid, since in many cases, region order is not uniquely defined (see Section 3.1). For example, how should such text regions as captions, headers, footers, or insets, be ordered? In an ideal benchmarking system, all possible reading orders should be considered equally correct.
– The current metric only takes *insertions* and *block moves* into account, while *deletions* are assumed to have zero cost. The reason for this is to avoid penalizing systems that recognize "too much", such as recognizing text inside images that do not appear in the ground-truth. This means that an algorithm will not be penalized for classifying noise regions as text, or worse, for detecting images as text.
– The method requires a zoning algorithm to be part of an OCR system, in order to be able to benchmark it; hence, stand-alone segmentation algorithms cannot be benchmarked. Furthermore, the zoning subsytem of an OCR sytem can not be evaluated on segmenting documents in languages that are not supported by that OCR system.
– In order to be able to "subtract" the character errors from the total cost, the character recognition performance needs to be independent of the segmentation performance. This is often not true: a "messy" segmentation generally results in

poor character recognition performance (e.g., if the segmentation splits or joins two galleys, the hyphens are incorrectly paired and the OCR engine can lexically verify fewer words). Alternatively, a poor OCR engine might "hide" some of the segmentation errors when the errors overlap.

### *2.2 Proposed Region-Based Approach*

For the reasons listed above, we have taken a region-based approach to benchmarking page segmentation: the segmentation quality is evaluated by comparing the segmentation output and the corresponding ground-truth which are both ASCII files describing the regions on the page. A region is represented as an arbitrary polygon (the outline of a zone) together with various attributes, such as its *type*, *subtype* and *parent* zone, as explained in Section 3.1.

To find the correspondence between the segmentation and the pre-stored ground-truth, first the overlap between the segmentation regions and the ground-truth regions are determined. This is done using the regions' ON-pixel contents instead of their polygonal representations: each region is modeled as the set of ON-pixels its associated polygon contains. This way, the detailed shape of region polygons is not taken into account, and insignificant differences between ground-truth and segmentation polygons are automatically ignored.

By analyzing the region correspondences, one can detect such segmentation problems as missed, merged, split or misclassified ground-truth regions, and extraneous segmentation regions. This overall scheme is efficiently implemented through the use of *region maps* and *error maps,* as described in Section 4. As shown later, this system is more complex and cumbersome to use than a text-based system, but offers greater flexibility and avoids most of the limitations of the previous approach.

## 3 Creation of Ground-Truth Files

A reasonably large number of document images and their segmentation ground-truth is needed to evaluate a segmentation algorithm. An X-windows tool, called *GroundsKeeper,* was developed to speed up the creation of ground-truth files. This tool, described in Section 3.2, allows the user to view a document image, zone it with simple mouse clicks, and specify information about each zone, as well as the ordering relationships between zones. The zoning information can then be saved in a simple ASCII format, called *RDIFF (Region Description Information File Format),* as the ground-truth file.

### *3.1 Representation of Ground-Truth Information*

The RDIFF format used to represent page segmentation information (ground-truth or segmentation) is a simple ASCII format containing some header information such as the name of the associated document image, the resolution, and the page size, followed by a list of region descriptions. Each region has an associated *type* (e.g., text, image), and *subtype* (e.g., headline, caption), may refer to a *parent* region, and may have an arbitrary number of attributes (e.g., reverse-video). Region shape itself is an arbitrary polygon, specified by the coordinates of its vertices. The parent zone information is used to specify hierarchical information. For instance, cells inside a table are zoned individually and point to the table (zoned as a whole) as their parent region. Similarly, any text inside a halftone is zoned separately and points to the halftone as its parent region.

Even though the RDIFF format defines a type and a subtype for a region, the type of a given subtype can be redefined for a given benchmarking task. For example, a logo is defined as an image subtype by default, however, if a particular application wanted to treat logos as text regions and evaluate the segmentation accordingly, it could do so by overriding the type of a logo for the benchmarking process (see sections 3.2 and 4.1).

Some of the features of the RDIFF format are as follows:

- It is a tag-value based format: each entry in the format starts with a tag, followed by the value of this tag. The advantage of this approach is its flexibility: an RDIFF reader typically only knows about the tags that are relevant to its purpose and ignores the rest. In addition, even as new tags are added with each new release of the format, the new RDIFF files remain backwards compatible with previously written RDIFF readers.
- It can easily be expanded. For instance, region shape is currently an arbitrary polygon. This is sufficient in most applications, however if it became important to support regions with holes, or disconnected regions, new region representations (e.g., lists of polygons) could be added and a new tag could be used to specify the representation scheme chosen for each region (the default scheme being the single polygon).
- It imposes no restrictions on region locations: regions can overlap, be included in one another, etc.

An example of RDIFF file is given in Figure 3.1.

*Partial Region Ordering*

Note the R_ORDER_RULE at the end of the RDIFF file shown in Figure 3.1. This rule represents a set of ordering constraints between regions, where $X < Y$ indicates that region X precedes region Y. This provides us with a *partial ordering* of the zones describing the *necessary* ordering relationships among the regions. As mentioned in Section 2.1, one complete ordering of all regions is too strict, and instead, a num-

```
BEGIN_IMAGE
FILENAME /data/tif/h/ht21.tif
IMAGE_WIDTH 2560
IMAGE_HEIGHT 3300
IMAGE_XRES 300
IMAGE_YRES 300
END_IMAGE

BEGIN_SUMMARY
INIT_FILE_VERSION 1.8 1995/04/18
TOTAL_REGIONS 25
TEXT_REGIONS 22
IMAGE_REGIONS 3
END_SUMMARY

BEGIN_REGIONS
R_NUMBER 1
R_SUBTYPE HALFTONE
R_ATTACHMENT 0
R_PARENT 0
R_DRAW_TYPE Box
R_NAME Zone1
R_ATTRIBUTES SHADED_BACKGROUND
REGION_POLYGON 301 692 71 681
...
R_NUMBER 9
R_SUBTYPE CAPTION
R_ATTACHMENT 1
R_PARENT 0
R_DRAW_TYPE Box
R_NAME Zone9
REGION_POLYGON 722 838 78 266
...
R_NUMBER 19
R_SUBTYPE REG_TEXT
R_ATTACHMENT 0
R_PARENT 0
R_DRAW_TYPE Constrained-Polygon
R_NAME Zone19
REGION_POLYGON 428 1086 376 982
COUNT 6
774 428
982 428
982 1086
...
R_ORDER_RULE 18<19<20<21<22<23
```

Fig. 1. Sample RDIFF file.

ber of ordering constraints (or rules) should be used to represent all possible region orderings of a document. This way we can, for example, order regular text regions and figures in two separate orders, using multiple R_ORDER_RULE tags. Fig. 2 shows an example where several ordering constraints are used to order figures belonging to each column among themselves, and to enable either of the two possible reading orders of the last four regions. Partial ordering thus makes it possible to deal with ambiguous cases by specifying only the necessary constraints.

Even a partial ordering is not always sufficient to fully and unambiguously represent the relationships between zones of a complicated document. In some cases, two regions are not ordered with respect to each other but are logically associated (e.g. figures and their corresponding captions). In this case, the caption is "attached" to the image. Attachements are specified with the R_ATTACHMENT tag in the RDIFF language. The difference between attachments and orderings is that, for instance, even though a caption does not have to come before or after the figure it belongs to, if the figure is moved the caption should be moved with it. Currently, region ordering accuracy is not benchmarked.

### *3.2 GroundsKeeper*

To create test suites of segmentation ground-truth files in the RDIFF format, we developed an X-Windows based tool, called *GroundsKeeper*

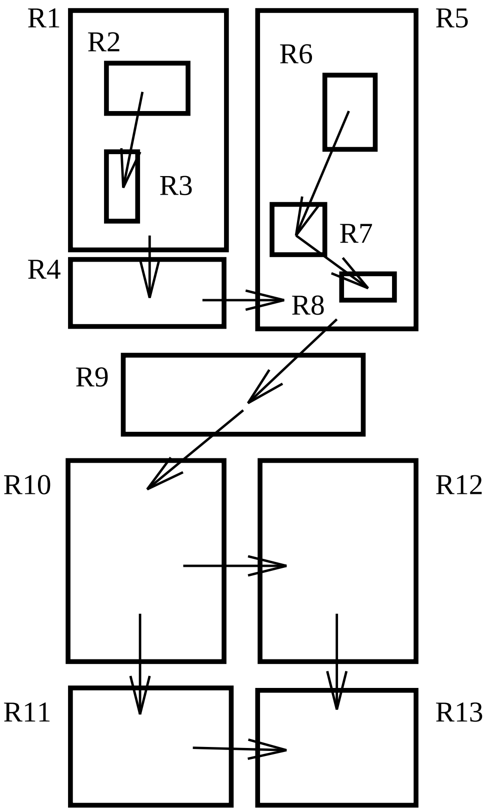


Fig. 2. The order between regions in this sample page can be specified using five partial orders: $R_2 < R_3$, $R_6 < R_7 < R_8$, $R_1 < R_4 < R_5 < R_9 < R_{10}$, $R_{10} < R_{11} < R_{13}$ and $R_{10} < R_{12} < R_{13}$. Note that the last two constraints indicate that the regions $R_{10} \ldots R_{13}$ can be read in one of two ways.

(see Fig. 3). *GroundsKeeper* allows a user to view a document image and draw zones of various types around the different page regions, using simple mouse clicks. The zones can be drawn using one of the following drawing methods: rectangles (sufficient for most cases), constrained polygons (polygons whose edges are vertical or horizontal), and arbitrary polygons (useful for complex layouts, or skewed pages).

After drawing a zone, one can label it with its type, subtype, parent zone, attached zones, and any number of *attributes*. Types, subtypes, and attributes shown in menu selections, are defined in a start-up file: they are therefore completely customizable for any particular application. For instance, we use the start-up file shown in Fig. 4 for our segmentation ground-truthing. Note that we chose to create fairly detailed ground-truth files, since one can always use less information than these files contain.

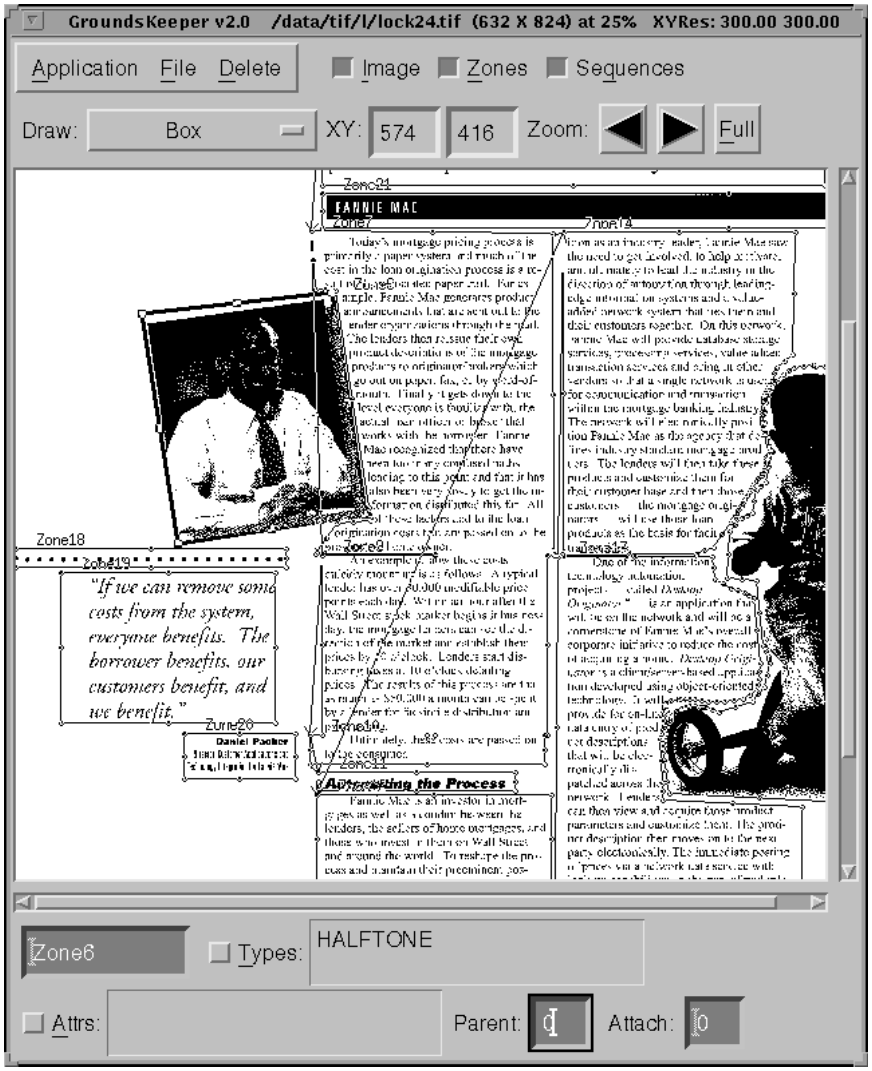
Each new zone is given a unique identifying number, displayed on the screen, and can be given a name (derived from its number by default). Once created, zones can be deleted, moved, or modified. Furthermore, previously created ground-truth files can be read and updated.

Region ordering rules (*sequences*) are specified by clicking on regions, one after the other; a sequence is then displayed as a succession of arrows on the screen. Multiple region ordering rules can be created in any order by starting a new sequence each time.

By default, the image is shown with the regions and the sequences drawn over it, but one can also display only the regions, the regions and sequences, or the image alone. The ability to display the image and the regions in different zoom levels is also available. In fact, a good way to use *Groundskeeper* is to first draw the regions coarsely at low-resolution, and then zoom-in and adjust them.

*Groundskeeper* has a number of additional features designed to make the creation of RDIFF files easier, such as the *inheritance* of region information. For instance, when zoning multiple cells inside a table, the user needs to input the region subtype (cell) and the parent (the table) information only once. The cells zoned subsequently would inherit this information by default, significantly speeding up the zoning process. In fact, for most pages, ground-truthing time is around five minutes.

Using *Groundskeeper* we have created half a dozen test suites of ground-truthed documents (magazines, flyers, newspapers, business letters, fax, tables), for a total of about 800 RDIFF files.



GroundsKeeper v2.0 /data/tif/l/lock24.tif (632 X 824) at 25% XYRes: 300.00 300.00

Draw: Box

XY: 574 416

FANNIE MAE

"If we can remove some costs from the system, everyone benefits. The borrower benefits, our customers benefit, and we benefit."

Automating the Process

Zone6

□ Types: HALFTONE

□ Attrs:

Parent: 0

Attach: 0

Fig. 3. *GroundsKeeper* window showing a document image together with some ground-truth regions created. Ordering constraints between regions are shown as arrows.

Creating segmentation ground-truth is not always straight-forward. A ground-truthing handbook was therefore written in which potentially ambiguous cases were described, in order to take as much subjectivity out of the ground-truthing process as possible. One of the decisions we have faced was the necessary level of detail; we have decided that, within limits, zones should be logically and structurally uniform, and that paragraphs were the smallest units of interest of regular text (i.e., we do not zone lines or characters separately). Figure 4 shows all the possible subtypes and attributes currently used in ground-truthing.

Furthermore, in order to remove the main source of ground-truthing ambiguity, which is deciding on the label (subtype) of a region, we have expanded *GroundsKeeper* to allow multiple region labels for a given region. However, the benchmarking algorithm currently uses only the first label of a region.

## 4 Our Segmentation Benchmarking Algorithm

The performance of a given page segmentation system is evaluated by running the system on a large number of document images, organized in different test suites (e.g. magazine articles, flyers, business letters). Then, the output of the segmentation algorithm for each document is compared to the corresponding, previously created ground-truth, and various types of segmentation errors such as misclassifications of region types, missed pixels, and splitting and merging of regions, are detected.

As mentioned before, our approach is region-based: the segmentation output describing the regions on the page, is compared to the corresponding ground-truth of the same form. A region-based approach was previously thought as infeasible due to the lack of standardization of region representation schemes (e.g., boxes, polygons), which makes it difficult to find the correspondence between the ground-truth and the segmentation regions. This problem is handled by comparing the regions *not* by their shapes but their ON-pixel contents.

Hence, two regions that enclose the same image area (e.g. a headline) are correctly considered identical regardless of how tightly they enclose the area or what region representations schemes are used.

In the rest of the paper, $S$ and $G$ refer to the set of segmentation and ground-truth regions, respectively, and we use the word "overlap" to denote an overlap in ON-pixel contents.

*Region Maps*

In order to efficiently deal with regions as sets of ON-pixels, the first step in our system is to create *region maps*, one for the ground-truth and another for the segmentation regions. Each map is a reduced-resolution representation of the original document image, in which each pixel is tagged to indicate all the region(s) it belongs to.

In building the region maps, one has the choice of allowing for region overlaps or not: regions can be rendered while keeping track of their overlaps as described below, or they can be rendered one after the other without worrying about overwriting previously rendered pixels. Apart from the global specification about how overlapping regions should be rendered, one can also define some region subtypes to be *hollow*, so that they are rendered to contain only those pixels that are not included in any other region (see Section 4.1).

To keep track of the overlaps, the system requires that region numbers (i.d.'s) are all positive, and uses negative numbers to indicate the overlaps. Specifically, each ON-pixel is assigned the number of the region it belongs to; if this pixel belongs to more than one region, it is given a unique *negative* number that represents the particular set of regions that are overlapping at that pixel location. A hypothetical example is shown in Figure 5, where, for instance, label -4 corresponds to the overlap between regions 1 and 3 only, whereas label -6 corresponds to the overlap between regions 1, 3 and 4.

Region maps provide a compact and efficient way of handling arbitrarily complex documents where several regions might overlap.

*Finding Region Correspondences*

Using region maps, the amount of overlap between each segmentation region and each ground truth region is first determined. This is done by scanning the two region maps, for each region pair in $S \times G$ whose bounding boxes overlap.

After finding the overlaps, the actual correspondences between the segmentation and ground-truth regions are determined. An overlap does not necessarily indicate a correspondence: a segmentation region overlapping with two ground-truth regions (one enclosed in the other) corresponds to only one of them. For instance, in Fig. 6.a, the dashed segmentation region corresponds to the enclosing image region (the text region is missed in the segmentation), whereas in Fig. 6.b, the segmentation region corresponds to the enclosed text inside the image (the image region is missed).

Since two ground-truth regions overlap only if one is enclosed in the other, deciding on which ground-truth region to assign the overlapping segmentation region is done by computing a simple match score between the overlapping ground-truth and the segmentation regions. For a ground-truth region $g$ and the segmentation region $s$, the score is defined as the percentage of the ON-pixels of the ground-truth region covered by $s$ minus the percentage of the ON-pixels of $s$ outside of $g$, as:

$$match(g, s) = overlap(g, s)/area(g)$$

$- outside(g, s)/area(s)$

For instance, in Fig. 6.a, the segmentation region $S1$ fully covers both $G1$ and $G2$, but because it has a significant number of ON-pixels outside $G2$, its match score to $G2$ is low. On the other hand in Fig. 6.b, $S1$ covers only a small portion of $G1$ while fully covering $G2$ and it does not contain any ON-pixels outside $G1$ or $G2$, its match score to $G2$ is high.

### *Region Alignments*

When a ground-truth region corresponds to more than one segmentation region, the *alignment* of the segmentation regions is analyzed. Two regions are called *vertically aligned* if they do not overlap along the $x$-axis, otherwise the alignment is called *horizontal alignment* (see Figure 7). Similarly, when a segmentation region overlaps with more than one ground-truth region, the alignment of the ground-truth regions is analyzed.

From region overlaps and the alignment types, the occurrence of various segmentation errors (e.g., missed or horizontally merged ground-truth regions) can be detected. The image pixels affected by each error and the associated costs are then determined by further analysis, as explained below.

### *Detecting Segmentation Errors*

After finding the region correspondences and analyzing their alignments, error analysis is performed to detect the occurence of various kinds of segmentation errors, their locations, and the associated penalties. The segmentation errors that are currently detected are the following:

- noise region
- missed ground-truth region
- vertically split ground-truth region
- horizontal split ground-truth region
- vertically merged ground-truth region
- horizontally merged ground-truth region
- mislabeled pixels

A *noise* region is a segmentation region that does not overlap with any ground-truth regions: it is incorrectly detected as a valid region. Similarly, a *missed* region is a ground-truth region that does not overlap with any segmentation regions: it is incorrectly thought to be a noise (invalid) region. On the other hand, if no single segmentation region covers a given ground-truth region, that ground-truth region is called *split*. The alignment of the segmentation regions indicate whether the split is a horizontal or a vertical one. On the contrary, if a segmentation region matches more than one ground-truth regions, it is *merging* them. As above, the alignment of the ground-truth regions indicate whether the merge is a horizontal or a vertical one.

The severity of these segmentation errors varies from application to application. For instance, if segmentation is to be used in the context of an OCR system, vertical merging of text regions is usually not a big mistake, especially if the merged regions appear consecutively in the reading order. On the other hand, horizontal merging of text regions should be strongly penalized. For halftone regions, vertical and horizontal merges are usually equally bad.

### *Estimating Error Costs*

When a segmentation error is detected, the associated image area is analyzed (using the region maps) to determine the cost of the mistake. For an objective and fair evaluation of the segmentation, one should not flag an entire region as "bad" if it has only few erroneous pixels. For instance, if a ground-truth region matches only one segmentation region but few of its pixels are not covered, it would be too severe to mark the entire region as being split. In order to differentiate between the severity of a split involving few pixels (very common on regions where the boundaries are not well defined) and one that involves most of the region, the severity of the mistake needs to be assessed in terms of the number of ON-pixels involved. For instance, when a ground-truth region is merged, our system normally considers only the area that is actually merged (not the entire region) as erroneous, as shown in Fig. 8. The exact area (shaded in Fig. 8) where the segmentation error occurs is found by scanning the two region maps. For instance, when a ground-truth region, $g$, is split, each line (row) is scanned to see if there is a corresponding segmentation region covering that entire line. If not, it means that the line is split and all the pixels on it are marked as such. Similarly, each line of a merging segmentation region is scanned to see whether there is a corresponding ground-truth region covering all the pixels on that line. If not, all the pixels on that line are marked as merged.

Alternatively, the user could penalize the entire region, regardless of the size of the actual error area. The method for computing error costs are indicated in a start-up file, as one of the following alternatives:

- *unit* cost: the cost of an erroneous region is 1 regardless of its size. This makes sense when segmentation is to be manually corrected, in which case the main thing that matters is the number of regions to be re-zoned, not their size. When the unit-cost-method is used, the total cost is normalized by the number of ground-truth regions on the page.
- cost proportional to the *height* or *size* (ON-pixel count) of the erroneous area. If the segmentation output is to be directly fed to the OCR system, it is more appropriate to compute the cost in proportion to the height or size of the erroneous area. One could even penalize images by size and

text by height. For height and size based methods, the total cost is normalized by the number of lines containing ON-pixels, and the number of ON-pixels on the page, respectively. The default error cost is the size of the erroneous area.

The regions are currently analyzed along the horizontal scan lines because most English text is written in that direction (hence, horizontal splits or merges are severe). In order to handle documents that might contain vertically written text, it would be necessary to also include the direction of writing inside a region in the ground-truth file and scan for errors along this direction. This will be addressed in future versions.

### *Weighting and Combination of Errors*

When a segmentation error is detected, how should it be weighted with respect to other errors? How should errors be combined if they affect the same ground-truth region? For example, if (part of) a ground-truth region is found to have been split twice by segmentation, should this be considered twice as bad? Similarly, if this region is found to have been both merged and split, should one of these errors have precedence over the other one, or should both errors be counted?... The answers to all these questions depend on the type of segmentation being benchmarked, and can be specified in the start-up file used by our benchmarking tool (see next section).

To deal with all the possible ways one may want to weight and combine errors, we use a scheme based on *error maps* (same concept as the region maps). Using an error map, we can keep track of and penalize all the errors associated with a given pixel in the image (e.g., an erroneous split and an erroneous merge), or only the most serious one. The weight of an error, indicating the severity of the error, is specified in the start-up file. One can specify, for instance, *the weight of mislabeling a text region with shaded background as halftone.* Hence, in addition to indicating what the cost should be proportional to (unit, or by height or by size), the start-up file can indicate which errors to take into account (all or the most serious), and what the weight for a given error should be. The weights of the errors are used to determine whether a particular error should be penalized (positive weight) or not, and how severe it is with respect to the other errors.

After all errors have been detected, the segmentation quality is characterized using the costs of each of the error types (e.g., noise, splits, merges) and the overall page cost. The system also reports the cost associated with each specific error (e.g., reverse-video text recognized as halftone) that has a non-zero weight in the start-up file. Hence, one can even vary the amount of information output by the system about the segmentation performance.

As an example, consider the set of ground-truth regions shown in Fig. 9: eight text regions and two image regions are present. The double-spaced text was zoned into five different paragraphs according to our ground-truthing guidelines, but for most applications, zoning it into two galleys would have been sufficient. Hence, in the start-up file, the cost of vertically merging text regions were set to zero. Fig. 10 shows the output of the automatic segmentation algorithm which mistakenly combined two galleys into one. Furthermore, the text part of the logo was split and joined to the merged text galleys. The corresponding error map created by our benchmarking algorithm is shown in Fig. 11. The black pixels in the map show the horizontally merged text of the two galleys and the horizontally split pixels of the logo.

Note that we consider all the pixels (ON or OFF) between the first and last ON-pixels on a line as defining that line, and mark them as erroneous, if the line is found as erroneous. This not only increases the weight of the text areas with respect to the dense image areas, but also lessens the difference between the weights of boldface characters and regular ones.

The output of the benchmarking for the example shown in Figures 9 and 10 is given in Fig. 12. Zoning errors and labeling errors are listed separately, as they belong to different aspects of the segmentation process. The start-up file used to produce this output, specified six different types of mislabeling (e.g., mislabeling text as image, confusion of image subtypes), but only one type of vertical merge, horizontal merge, etc. Also recall that the cost of vertical merges and splits was set to zero.

In this example the overall percentage of bad pixels is found to be about 25% with the particular settings of the start-up file, when about two thirds of the document is merged. However, because the analysis was based in this case on region sizes (ON-pixel counts), the correct zoning of the halftone skewed the results favorably. One way to achieve an overall cost that would reflect our "eyeballing" judgement would be to to base the analysis on region heights.

### *4.1 Customizing the Benchmarking*

Just like *GroundsKeeper,* our benchmarking tool *Cluzo* can be customized to a particular application. The start-up file required by the program is decomposed into several sections:

- Listing of all the region types, subtypes, attributes etc. that can be encountered in RDIFF files.
- The type (text or image) of each region subtype. For instance, the type of the subtype LOGO can be set to IMAGE or TEXT, depending on the application.
- Description of region equivalencies: in this section, a user can specify, for instance, that subtype A is in fact equivalent to subtype B, and that subtype D is equivalent to subtype B as well. For example, if the user does not care about distinguishing between dif-

ferent subtypes of image regions (e.g., graphics, halftones, line-art) and text regions (e.g., captions, headlines, footers), the total number of region subtypes could be reduced to two. This is also useful in mapping the subtype names used in the segmentation output to those used in the ground-truth (e.g., TIME-STAMP to TIMESTAMP).

- Listing of regions to ignore. With this feature, it is possible to completely exclude certain region subtypes (e.g., frames) for the benchmarking process.
- Listing of hollow regions. The regions defined as hollow will be assigned only those pixels inside them that do not belong to other regions (e.g., the subtype FRAMEBOX which is the subtype of frames around an image or text area, is declared as hollow, and therefore only contains those pixels that actually form the frame). This feature not only makes ground-truthing of hollow regions easy, but also extends the possible region shape to an arbitrary polygon *with* holes!
- Listing of all the different error types, together with their weights. This can be done for generic regions as well as for particular region types. For example, the start-up file we currently use contains the following:

```
Weight_of Vertically_split
  REGION 0
Weight_of Vertically_split
  GRAPHICS 3
Weight_of Vertically_split
  HALFTONE 3
```

This specifies that the weight of a vertical split is 0, except when it involves a GRAPHICS region or a HALFTONE region, in which case the weight is 3.

A more complicated example is:

```
Weight_of Mislabeled
  REGION 2
Weight_of Mislabeled
  REG_TEXT with-attribute
  SHADED_BACKGROUND
  as HALFTONE 1
```

This indicates that the weight of misclassifying a (any) region is 2, and the weight of misclassifying a regular text region with shaded background as halftone is 1 (less severe). As mentioned before, the system will output the specific cost of misclassifying regular text regions with shaded background as halftone, in addition to the more general errors.

Weights specified later in the start-up file overwrites the previous ones; this makes it easy to specify the weights, from more general to more specific. For instance in the above example, the mislabeling cost for the generic region (keyword "REGION") is specified first, and then the exceptions are indicated.

- What error costs should be proportional to (unit, height, size). By default, the cost of an error is the number of pixels involved in that error.
- Definition of the error combination method: when an area of the page is involved in multiple segmentation mistakes, should all these mistakes be taken into account, or should only the most serious

mistake (the one with highest associated weight) prevail?

## 5 Conclusions, Future Work

We described the *Pink Panther* automatic page segmentation benchmarking environment. *Pink Panther* consists of two separate parts: *GroundsKeeper*, an X-windows tool for creating segmentation ground-truth files, and *Cluzo*, the segmentation benchmarking system.

Our approach to benchmarking segmentation is region-based: segmentation quality is assessed by comparing the segmentation output, described as a set of regions, to the corresponding ground-truth. This approach enables us to benchmark segmentation performance directly, unlike text-based techniques that rely on the OCR output.

The benchmarking system can be used to benchmark various different segmentation tasks and is able to keep track of a large number of error types, providing very detailed information on the segmentation quality. In addition to reporting the location, type and severity of all the segmentation mistakes on the page, the system also computes an overall cost from the individual costs of the mistakes, according to the specifications in the start-up file. The relative weights of various error types and how they should be combined, are indicated in the start-up file by the user. Another powerful aspect of the system is its ability to deal with multiple region orderings, which is important to properly handle complex document structures. The system has been developed with flexibility in mind, considering all the different ways this system could be used in practice. This makes it a compelling alternative to text-based segmentation benchmarking approaches.

Note that by penalizing errors only by the number of pixels associated with that error (and not the whole region), small zoning mistakes in the ground-truth, that might be invisible in low resolution, become insignificant. Using partial region ordering also removes some of the potential ground-truthing ambiguity.

The system is now available on the DIMUND (Document Understanding and Character Recognition) list web site (`http://documents.cfar.umd.edu`).

In future versions of our benchmarking environment, we are planning on extending the benchmarking environment to handle documents where the direction of the writing can be vertical as well as horizontal, such as in Chinese and Japanese documents and in some captions and sidebars in English documents (often without upright letters).

Another extension to our benchmarking system that is very significant is the capability of handling grayscale and color documents: we are currently adding this to *GroundsKeeper* and considering a few options for extending *Cluzo* to benchmark grayscale and color images as well. This will potentially open a whole new range of applications beyond document analysis, such as in medical imaging, for our benchmarking.

## References

[1] Ying-Wei Lin. Digital image processing in the Xerox docutech document processing system. In Luc Vincent and Theo Pavlidis, editors, *SPIE/SPSE Vol. 2181, Document Recognition*, San Jose CA, February 1994.

[2] M. Nadler. A survey of document segmentation and coding techniques. *Comp. Vis., Graphics and Image Processing*, 28:240–262, 1984.

[3] Luc Vincent. Page segmentation in Textbridge. Unpublished Document, Xerox Desktop Document Systems, 1993.

[4] J.L. Fisher, S.C. Hinds, and D.P. D'Amato. A rule-based system for document image segmentation. In *Int. Conf. on Pattern Recognition*, pages 567–572, Atlantic City, NJ, June 1990.

[5] Jaekyu Ha, Robert Haralick, and Ihsin Phillips. Document page decomposition by the bounding-box projection technique. In *Int. Conf. on Document Analysis and Recognition.*, pages 1119–1122, Montreal, 1995.

[6] Theo Pavlidis and J. Zhou. Page segmentation by white streams. In *Int. Conf. on Document Analysis and Recognition.*, pages 945–953, Saint-Malo, France, 1991.

[7] Henry S. Baird. Background structure in document images. In *Advances in Structural and Syntactic Pattern Recognition*, volume 5, pages 253–269. World Scientific, 1992.

[8] A. Antonacopoulos and R.T. Ritchings. Flexible page segmentation using the background. In *12th Int. Conf. on Pattern Recognition*, pages 339–344, Jerusalem, October 1994.

[9] Stephen V. Rice, Junichi Kanai, and Thomas A. Nartker. A preliminary evaluation of automatic zoning. Technical report, ISRI, 1993.

[10] Junichi Kanai, Tom A. Nartker, Stephen V. Rice, and George Nagy. Performance metrics for document understanding systems. In *Int. Conf. on Document Analysis and Recognition.*, pages 424–427, Tsukuba, Japan, October 1993. IEEE Computer Society Press.

[11] Junichi Kanai, Stephen V. Rice, Thomas Nartker, and George Nagy. Automatic evaluation of OCR zoning. *IEEE Trans. Pattern Anal. Machine Intell.*, 17(1):86–90, January 1995.

[12] Sabine Randriamasy, Luc Vincent, and Ben Wittner. An automatic benchmarking scheme for page segmentation. In Luc Vincent and Theo Pavlidis, editors, *SPIE/SPSE Vol. 2181, Document Recognition*, San Jose CA, February 1994.

[13] Berrin A. Yanikoglu and Luc Vincent. Ground-truthing and benchmarking document page segmentation. In *Int. Conf. on Document Analysis and Recognition.*, Montreal, August 1995.

[14] Esko Ukkonen. Algorithms for approximate string matching. *Information and Control*, 64:100–118, 1985.

[15] Stephen V. Rice, Junichi Kanai, and Thomas A. Nartker. The third annual test of OCR accuracy. Technical report, ISRI, 1994.

[16] Stephen V. Rice, Frank Jenkins, and Thomas A. Nartker. The fourth annual test of OCR accuracy. Technical report, ISRI, 1995.

[17] Sabine Randriamasy and Luc Vincent. Benchmarking page segmentation algorithms. In *IEEE Int. Conf. on Computer Vision and Pattern Recog.*, Seattle, WA, June 1994.

[18] Luc Vincent and Berrin Yanikoglu. A complete environment for ground-truthing and benchmarking page segmentation algorithms. In *Symposium on Document Image Understanding Technology*, Bowie, MD, October 1995.

[19] Olivier Desforges and Dominique Barba. Segmentation of complex documents multilevel images: a robust and fast text bodies-headers detection and extraction scheme. In *Int. Conf. on Document Analysis and Recognition.*, pages 770–773, Montreal, 1995.

[20] George Nagy and S. Seth. A prototype document analysis system for technical journals. *IEEE Computer*, 25(7):10–22, July 1992.

[21] Lawrence O'Gorman. The document spectrum for bottom-up layout analysis. *IEEE Trans. Pattern Anal. Machine Intell.*, 15(10):1162–1173, October 1993.

[22] Theo Pavlidis and J. Zhou. Page segmentation and classification. *Comp. Vis., Graph. Im. Proc.: Graphical Models and Image Processing*, 54(6):484–496, November 1992.

```
ATTRIBUTE_NAME ANGLED
ATTRIBUTE_NAME BAR-CHART
ATTRIBUTE_NAME BULLETS
ATTRIBUTE_NAME CELL-TABLE
ATTRIBUTE_NAME CURVED-TEXT
ATTRIBUTE_NAME HAND-DRAWN
ATTRIBUTE_NAME ILLEGIBLE
ATTRIBUTE_NAME INCOMPLETE
ATTRIBUTE_NAME OUTLINED
ATTRIBUTE_NAME PIE-CHART
ATTRIBUTE_NAME REVERSE_VIDEO
ATTRIBUTE_NAME SHADED_BACKGROUND
ATTRIBUTE_NAME TAB-TABLE
ATTRIBUTE_NAME UNDERLINED
ATTRIBUTE_NAME UPSIDE-DOWN
ATTRIBUTE_NAME VERTICAL

REGION_TYPE_NAME CAPTION
REGION_TYPE_NAME CELL
REGION_TYPE_NAME DROPCAP
REGION_TYPE_NAME FOOTER
REGION_TYPE_NAME FOOTNOTE
REGION_TYPE_NAME FRAMEBOX
REGION_TYPE_NAME GRAPHICS
REGION_TYPE_NAME GRID
REGION_TYPE_NAME HALFTONE
REGION_TYPE_NAME HEADER
REGION_TYPE_NAME HEADLINE
REGION_TYPE_NAME HRULE
REGION_TYPE_NAME INSET
REGION_TYPE_NAME LINEART
REGION_TYPE_NAME LOGO
REGION_TYPE_NAME RAISEDCAP
REGION_TYPE_NAME REG_TEXT
REGION_TYPE_NAME SIDEBAR
REGION_TYPE_NAME SIGNATURE
REGION_TYPE_NAME SUBTITLE
REGION_TYPE_NAME TABLE
REGION_TYPE_NAME TABLE_COLUMN
REGION_TYPE_NAME TABLE_ROW
REGION_TYPE_NAME TIMESTAMP
REGION_TYPE_NAME UNKNOWN_TYPE
REGION_TYPE_NAME VRULE
```

Fig. 4. Example of *GroundsKeeper* start-up file specifying all possible region types and attributes.

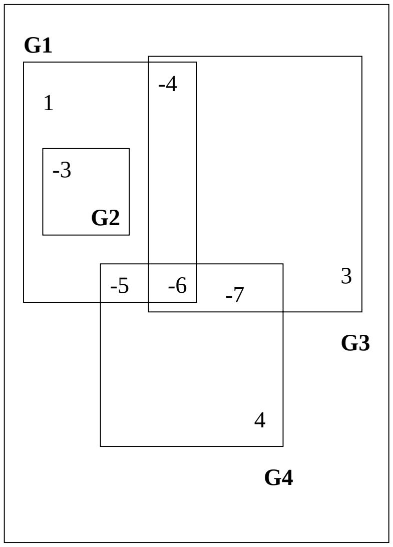


Fig. 5. Sample region map. Negative values correspond to unique overlaps.

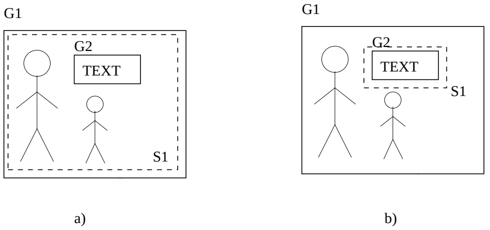


Fig. 6. Region correspondences. In a) the dashed segmentation region $S1$ corresponds to the enclosing image region $G1$, while in b) the segmentation region $S1$ corresponds to the enclosed text region, $G2$.

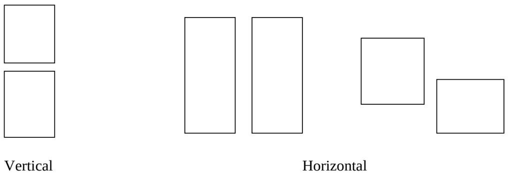


Fig. 7. Alignment types of regions (horizontal or vertical).

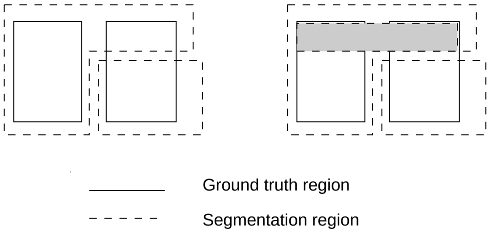
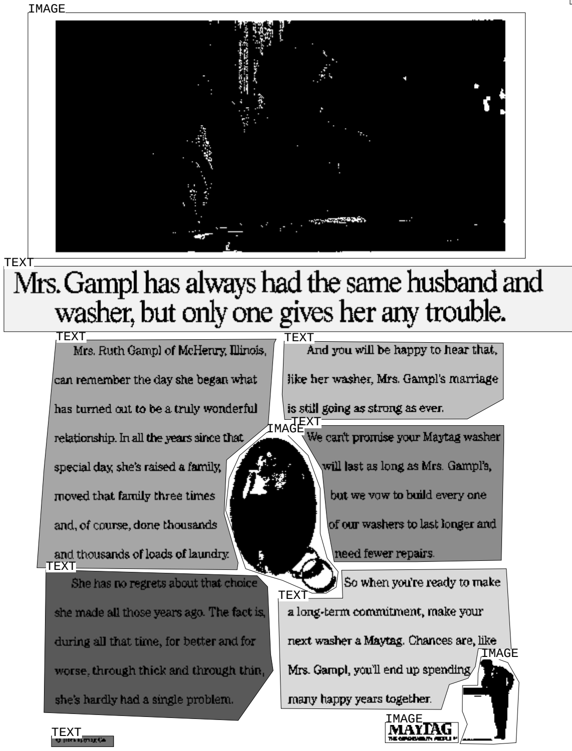


Fig. 8. When benchmarking the segmentation of an OCR system, only the pixels in the shaded area are found to be part of an erroneous horizontal merge of text regions.



Mrs. Gampl has always had the same husband and washer, but only one gives her any trouble.

Mrs. Ruth Gampl of McHenry, Illinois, can remember the day she began what has turned out to be a truly wonderful relationship. In all the years since that special day, she's raised a family, moved that family three times and, of course, done thousands and thousands of loads of laundry.

She has no regrets about that choice she made all those years ago. The fact is, during all that time, for better and for worse, through thick and through thin, she's hardly had a single problem.

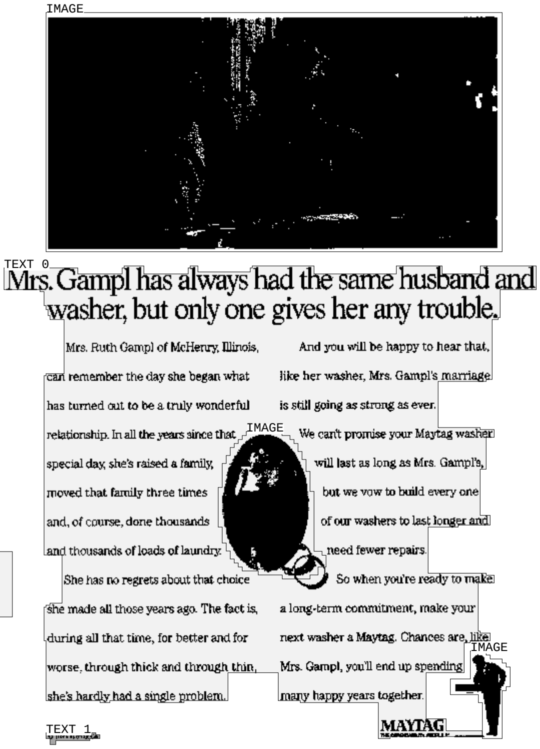
And you will be happy to hear that, like her washer, Mrs. Gampl's marriage is still going as strong as ever.

We can't promise your Maytag washer will last as long as Mrs. Gampl's, but we vow to build every one of our washers to last longer and need fewer repairs.

So when you're ready to make a long-term commitment, make your next washer a Maytag. Chances are, like Mrs. Gampl, you'll end up spending many happy years together.

MAYTAG

Fig. 9. Display of the ground-truth regions for a document image.



Mrs. Gampl has always had the same husband and washer, but only one gives her any trouble.

Mrs. Ruth Gampl of McHenry, Illinois, can remember the day she began what has turned out to be a truly wonderful relationship. In all the years since that special day, she's raised a family, moved that family three times and, of course, done thousands and thousands of loads of laundry.

She has no regrets about that choice she made all those years ago. The fact is, during all that time, for better and for worse, through thick and through thin, she's hardly had a single problem.

And you will be happy to hear that, like her washer, Mrs. Gampl's marriage is still going as strong as ever.

We can't promise your Maytag washer will last as long as Mrs. Gampl's, but we vow to build every one of our washers to last longer and need fewer repairs.

So when you're ready to make a long-term commitment, make your next washer a Maytag. Chances are, like Mrs. Gampl, you'll end up spending many happy years together.

MAYTAG

Fig. 10. A segmentation result for the same image as in Fig. 9. Note the horizontal merging of text galleys.

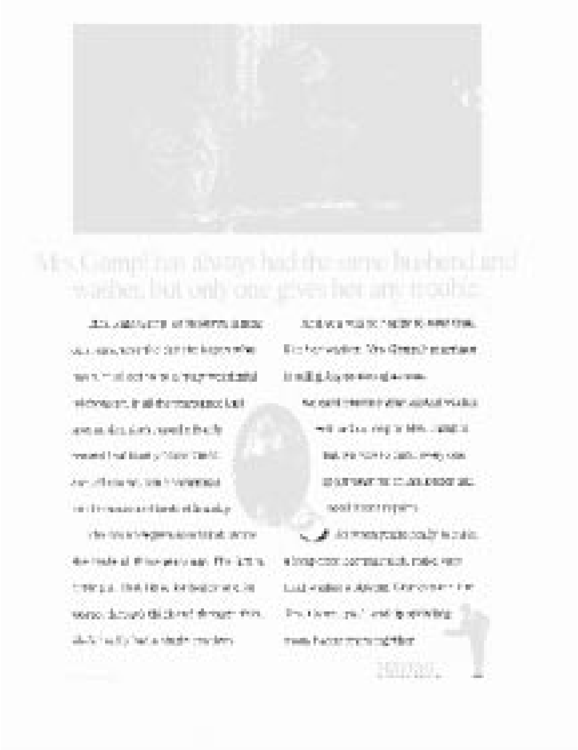

Mrs. Gompf has always had the same husband and
washer, but only one gives her any trouble.

Fig. 11. Error map corresponding to the segmentation result displayed in Fig. 10. Pixels involved in a merging error are shown in black.

```
Ground-truth file: bhng11.sgt
Segmentation file: bhng11.rdiff
Image file:        bhng11.tif
ZONING:
Missed regions     = 0.00
Noise regions      = 0.00
Horizontal merges  = 22.65
Horizontal splits  = 0.94
Vertical merges    = 0.00
Vertical splits    = 0.00
```

Fig. 12. Sample benchmarking output.