BIRZEIT UNIVERSITY

Document Layout Analysis

Using Minimum Homogeneity Structure

Majd Taweel - 1161422
Ibrahim Muala - 1160346

Computer Vision
Instructor: Aziz Qaroush

***Abstract*—This report demonstrates our implementation of the Minimum Homogeneity Structure (MHS) Algorithm. It firstly states the problem encountered, the motivation around solving such problem and a brief description of the system proposed. Furthermore, it shows an overview of the proposed system. It further provides results achieved after implementing the proposed system. Finally, it explains the conclusions achieved from implementing this algorithm.**

***Index Terms*—Document layout analysis, Minimum homogeneity algorithm, Minimum homogeneity structure, OCR**

I. Introduction

Document layout analysis is a rising topic in modern days. It has received more attention due to the need of exporting old printed documents into a digital version. This is important because organizations of all sizes and types need to have better and faster access to their previously recorded data, archives and reports that were printed on physical paper. These documents are firstly analysed using an accurate document analysis algorithm and the piped to an Optical Character Recognition (OCR) model to deduce the text content these documents contain.

The MHS algorithm [1], analyzes documents on several different layers sequentially before classifying the objects contained in the document. Fig. 1 provides a block diagram of the MHS algorithm. Our implementation first starts with pre-processing, outputting a binary (black and white) image. Then the binary image continues to text and non-text classification layer, which separates text and non-text content. This classification layer is actually what is known as Minimum Homogeneity Algorithm (MHA) [2], the previous version of the MHS algorithm. Fig. 2 shows a flowchart of the classification layer (MHA). This results into two different document for text and non-text content. Those two documents then passes different processes. Text lines are extracted from the text document and then paragraphs are separated using these extracted lines. Text regions are then deduced from the resulted paragraphs. The non-text document continues to yet another classification layer that then determines each non-text element's type (line, table, separator, image and negative-text). After that, the deduced regions from the text and non-text documents are refined and then labelled, resulting in the final page layout.

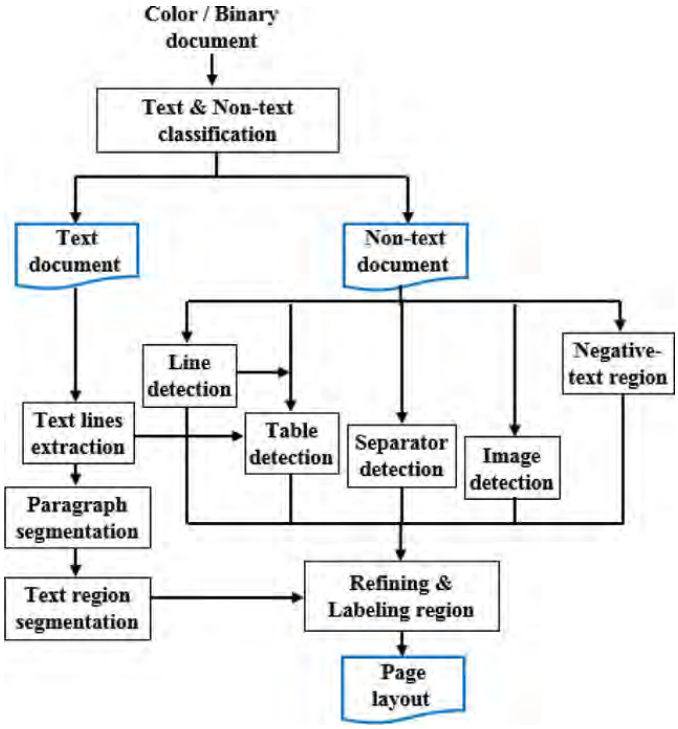


Fig. 1: Block diagram of the MHS algorithm [1].

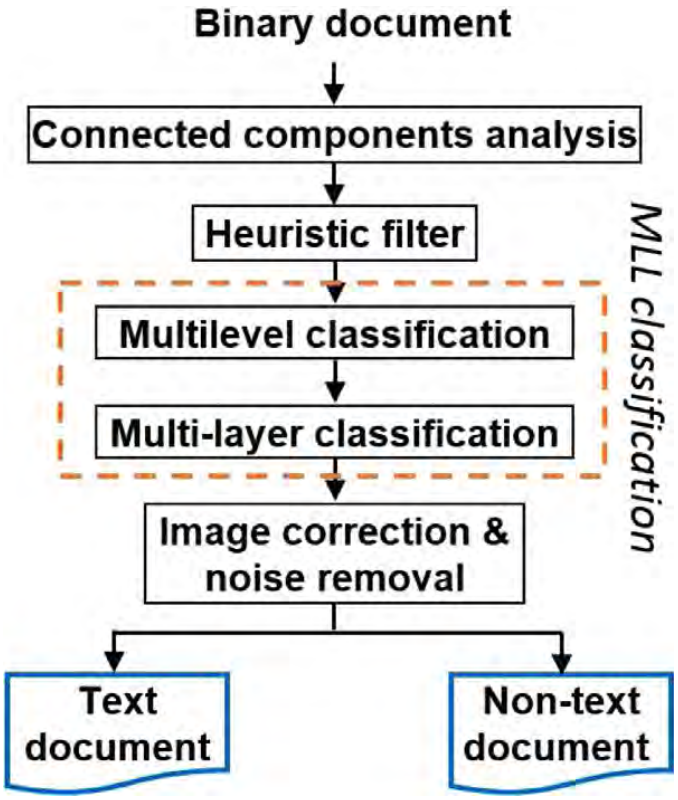


Fig. 2: Flowchart for the classification process (MHA) of the MHS system [1].

The rest of the report is organized in the following manner, Sect. II gives an overview of the system. Sect. III demonstrates results of our implementation. Sect. IV shows the limitations of our algorithm. Finally, the conclusion is found in Sect. V.

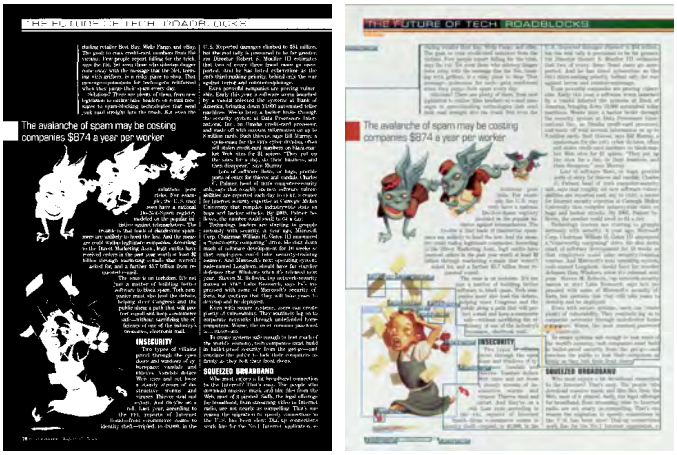
II. System Overview

The MHS algorithm consists of several processes or layers as stated before. These layers are explained in the following sections.

A. Pre-processing

This layer prepares the image for classification, and the quality of the operation it does on the image are reflected on the remainder of the layers, the binary image's quality greatly effects the rest of the operations. Take for instance, documents that contain light colored text with a white background, the text will match the background in the binary image due to the closeness of its color and the background's. Another example is text separators or lines, sometimes lines are partially merged with the background, because of inconsistencies in the line's thickness along its length. This will cause small parts of the line that were considered from the foreground to be miss-classified as text. Since these lines are probably very close to text regions, there is a high chance that they will be merged together, causing huge issues in deducing text regions, see Fig. 3.

The operations performed in this layer are as follows:



(a) Binary image with errors. (b) Page layout.

Fig. 3: Bad layout analysis output due to binarization errors.

1) If the image is colored it is converted to a grayscale image.
2) The image is slightly smoothed to remove some of the noise it may contain.
3) The smoothed image is converted to a binary image.
4) The binary image is resized to a certain resolution so that it does not take unreasonable computation time if its original resolution is very big.

Fig. 4 shows a sample image and its respective binary image.

(a) Original Image. (b) Binary Image.

Fig. 4: Image Pre-Processing and Binarization.

B. Heuristic Filter

The heuristic filter filters the image from obvious non-text elements depending on some heuristics, hence, the name heuristic filter.

Specifically, there are four properties that a Connected Component (CC) is considered non-text if it suffice any of them:

1) The area of the CC is less or equals to 6 pixels (considered as noise).
2) The bounding box of CC includes 4 or more other CCs.
3) The density of the CC is less than 0.06.
4) The height to width rate or width to height rate is less than 0.06.

The non-text components determined from the previous step is then removed from the binary image, achieving a new image that almost only contains text. Actually, in [1], the authors stated that the fourth condition is removed due to the miss-classification of the Korean character (–). However, this condition is preserved in our implementation for two reasons, first, many characters and text elements will be classified as non-text throughout the layers, and we will mostly be dealing with English documents. This condition works well in identifying lines, removing it for the sake of one character isn't justified since many other text components will be miss-classified at first. Nonetheless, this miss-classification will be dealt with later.

Fig. 5 shows the contours of text CCs (green) and non-text CCs (red) after going through the heuristic filter. It can be noticed as mentioned before, many text components we're classified as non-text. This inaccuracy must be dealt with either ways.

Fig. 5: CCs Contours After Applying The Heuristic Filter.

C. Multilevel/Multi-layer Classification

This layer consists of two sub-layers, multilevel classification and multi-layer classification. It also contains a recursive filter that is applied on the regions extracted from both sub-layers.

In multilevel classification, the image is split into homogeneous regions iteratively. This splitting is done both horizontally and vertically until all regions are homogeneous or can't be split anymore.

The splitting is executed by leveraging horizontal and vertical porijections of images. In our implementation, a region is considered as an object, as soon as a region object is initialized, its projections are extracted. Fig. 6 show such projections.

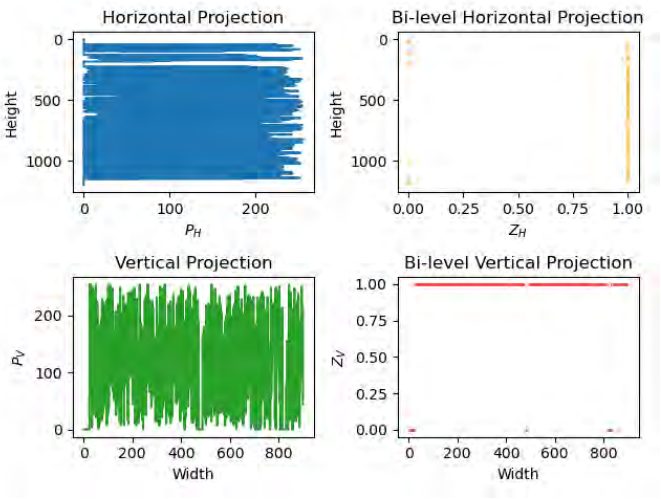


Fig. 6: Horizontal and Vertical Projections and Bi-Level Projections of The Whole Image.

As we can notice from Fig. 6, the original projections is very rough and need to presented in a better way so we can benefit from them. That is why the bi-level projections is extracted by transforming each value that is more than 0 to 1. These projections represent horizontal and vertical lines in the image they were extracted from. Regions are split in regard with these lines. The splitting happens on what may call foreign lines. Those can be defined as white or black lines that have very different heights (widths in vertical projections) than its neighbors. Most likely, these lines have larger heights (widths in vertical projections). Such a line can be interpreted as headings (black line), or large whitespace (white line) separating chapters or paragraphs.

After achieving the homogeneous regions, the recursive filter is applied on each of these regions. The filtered regions is further split into higher order homogeneous regions (if possible) and the recursive filter is applied again. This is repeated until the regions can't be modified anymore. Detailed specification of the recursive filter are provided in [2].

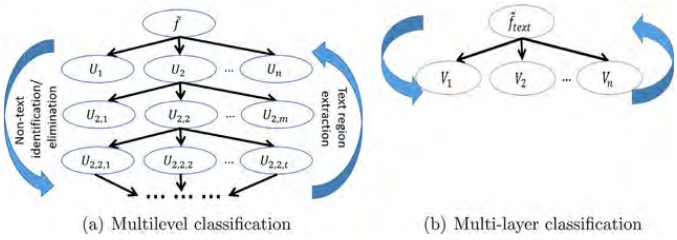
In multi-layer classification, the image is split into the first-level homogeneous regions (only once), then the recursive filter is applied on those regions. This operation is repeated until the regions can't be modified anymore. Note that, when repeating this operation, the first-level homogeneous regions are not further split, instead we re-split the whole image and retrieve the new first-level homogeneous regions. Fig. 7 illustrates the flow in the multilevel and multi-layer classification.

The details of multilevel and multi-layer classification layer can be found in [1] and [2].

Fig. 8 shows text and non-text components after the multi-level and multi-layer classification.

D. Text Segmentation

After the multilevel and multi-layer classification, the text image is almost ready to be segmented into paragraphs. Now,



(a) Multilevel classification (b) Multi-layer classification

Fig. 7: Illustration of multilevel classification and multi-layer classification [1].

Fig. 8: CCs Contours After Applying Multilevel and Multi-Layer Classification.

this layer will double as a text segmenter and a text and non-text classifier. As we seen from the steps before and the images produced from them, some elements were miss-classified, but, these elements almost always reside inside a region that only contains elements from their correct class.

Firstly, we perform morphological closing that only closes horizontally. Such closing can be achieved by using for example, a kernel of height 1 and width 5. This will connect horizontally close characters with each other. This can be seen in Fig. 9.

Then, we filter the remaining whitespace depending on some heuristics as described in [3]. Fig. 10 is a color coded image that shows horizontally chains (blue, green and red) and removed whitespace between them (dark blue, dark green, dark red and dark purple). Dark blue represents removed whitespace due to its small size, dark green for being isolated, dark red for being within-column and dark purple for being labeled as a candidate within-column whitespace and resides at the top or bottom of a vertical whitespace chain.

After going through this process, almost all components belonging to the same line are connected. However, there is still one step before segmenting the paragraph. As noted many times before, some components were miss classified, to fix this we need to move text components classified as non-text to the text image and move the non-text components found in the text image to the non-text image. The way this is implemented is

Fig. 9: Applying morphological closing on the text image.

Fig. 10: Horizontal Chains and Removed Whitespace Between Them.

by checking for intersection between components.

Firstly, the bounding box image is extracted from the text image. The bounding box image is simply an image that contains the bounding box of each CC instead of the CCs themselves. This is demonstrated in Fig. 11.

Then, morphological closing is applied to the bounding box image, this time vertically, using a kernel of height 3 and width 1 for example. In our implementation this was done for 4 iterations, to ensure that close components merge appropriately. Then we check for every element in the non-text document, if this element's area is less than some text element and they are almost completely intersected, then thus non-text element is a text element and belongs to the text document. The same thing is repeated for text elements. After these steps, the image in Fig. 12 is achieved.

It can be observed that there are now some extra components added into the text image that aren't connected to the text blocks they are contained in. These are the newly added elements previously found in the non-text image. It can also be noticed that some elements were removed from the text image, which are the elements that were moved to the non-text image.

Fig. 11: Bounding box text image.

Fig. 12: Adding miss-classified text components to the text image.

Finally, morphological closing is applied yet again to deduce the boundaries of the text blocks, since the newly added text elements need to be connected to their belonging text blocks. The morphological closing is done both horizontally and vertically this time, in our implementation, a kernel of height 3 and width 3 were used and the closing was done for 4 iterations. It can be observed from Fig. 13 that the new components are now connected with their text-blocks.

The text image is now finally ready to be segmented into paragraphs. After the text regions have been deduced, each region is separated taking in consideration excluding components intersected with multiple regions. The bounding boxes of each regions can be seen in Fig. 14.

Text regions is segmented by comparing 3 lines at the same time, and when an indented starting or ending line is found the regions is segmented. This can be illustrated in Fig. 15.

When the middle index is pointing to an indented line and the next index is also pointing to one, the region is segmented between the middle and the next index. If only the middle index is pointing to an indented line, the region is split between the previous and the middle index. The purpose of the previous

Fig. 13: Applying morphological closing on the bounding box text image with new components.

Fig. 14: Bounding boxes of text regions.

index here is being a reference for comparison, it is used to determine if a line is indented or not by comparing between their widths. The splits segmenting paragraphs can be seen in Fig. 16.

After this, these regions are smoothed using morphological closing, a small 3x3 kernel with 4 iterations will suffice. Fig. 17 shows the finalized text regions.

Fig. 15: Example of paragraph segmentation [1].

Fig. 16: Bounding boxes of text regions after paragraph segmentation.

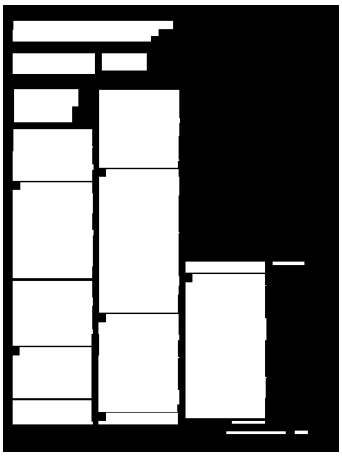


Fig. 17: Smoothed text regions after paragraph segmentation.

Fig. 18: Text and non-text components after non-text classification.

Fig. 19: Contours of refined regions.

E. Region Refinement

We now have separate text and non-text documents with the text document finalized. Text regions has already been refined in the text segmentation step. We can see from Fig. 18 that text regions are finalized, whereas the image in the top right isn't contoured in the correct matter.

To refine non-text regions, for each non-text component we check if the bounding box of this components intersects with other non-text components, and if the intersection between this component and the other smaller components it intersects with is greater than or equals to 0.9, then remove the other smaller components and replace the components with its bounding box. Fig. 19 shows the regions contours after refinement.

F. Non-Text Classification

This step classifies each non-text component into one of the following classes: {image, line (horizontal or vertical), separator, negative text}. The MHS originally classifies tables too, however, in our implementation table classification is not implemented due to its inaccuracy and tables variations that can't yet be effectively classified without using learning algorithms.

Firstly, each component with an area less than 50 pixels is considered as noise and then removed. The rest of the components continues on. If the component's density is almost 1 (bigger than or equals to 0.9), then it is a negative text candidate. This candidate is then negated and then passed to the multi-layer classifier. If the sum of the areas of the text components is bigger than the sum of the areas of the non-text ones achieved from the multi-layer classifier, then it is indeed a negative text component. If the component's density is less or equals to 0.1 then this component is a line. The orientation of the line is determined by checking the larger dimension, if its width is larger than its height, then it's a horizontal line, otherwise, it's a vertical line. If the component's density is less than or equals to 0.02 and the component contains text components, then it is classified as a separator. Otherwise, the component is an image.

After the classification step, text and non-text regions are labeled as shown in Fig. 20, providing the final page layout.

Fig. 20: Labeled final page layout image.

III. Results and Discussions

In this section, we will show the layout analysis results of some images from Pattern Recognition & Image Analysis Research Lab (PRIMA) dataset and their corresponding ground truths. This comparison is illustrated in Fig. 21, Fig. 22 and Fig. 23.

IV. Limitations

The performance of our implementation is reasonable. However it lacks in accuracy. Even though the results looks promising, it still needs improvements in several areas. For instance, the binary image quality needs to be improved substantially in order to classify components correctly. The segmentation process fails hardly for some images, see Fig. 21l. In addition, text segmentation still needs improvements, due to the existence of under-segmented areas where text blocks are very close to each other and over-segmented areas where text text lines and/or words are very far from each other. Future work might start by improving these areas. Moreover, the classification of non-text elements is still lacking in identifying tables, headings, headers, footers, graphics, graphs, captions, credits, floating text, drop-capitals and others, which requires great effort to implement. Furthermore, some miss-classification exists in non-text components, some dark images are classified as negative text, where they really are just normal images. Further, negative text is very general, headers and headings for example, can be negative text, hence, identifying the type of the negative text block is important. These are less crucial than the previous faults concerning segmentation, but still worth improving.

V. Conclusion

To sum up, Document layout analysis is a difficult process that can only produce sufficient outcomes through several levels of processing and computations, as can be seen from the implementation of this system. The layers in the MHS system are based on carefully extracted properties from many documents with different layouts. However, this system is heavily dependant on the quality of the binary image. In addition, the processes in this system are heavily dependant on statistics and observations found in most documents, which determines the accuracy of the system. Nonetheless, it still has room for some improvements.

References

[1] T. Aly, I. Na, and S. Kim, "Hybrid page segmentation using multilevel homogeneity structure," *ACM IMCOM 2015 - Proceedings*, 01 2015.

[2] ——, "Page segmentation using minimum homogeneity algorithm and adaptive mathematical morphology," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 19, 09 2016.

[3] K. Chen, F. Yin, and C. Liu, "Hybrid page segmentation with efficient whitespace rectangles extraction and grouping," in *2013 12th International Conference on Document Analysis and Recognition*, 2013, pp. 958–962.

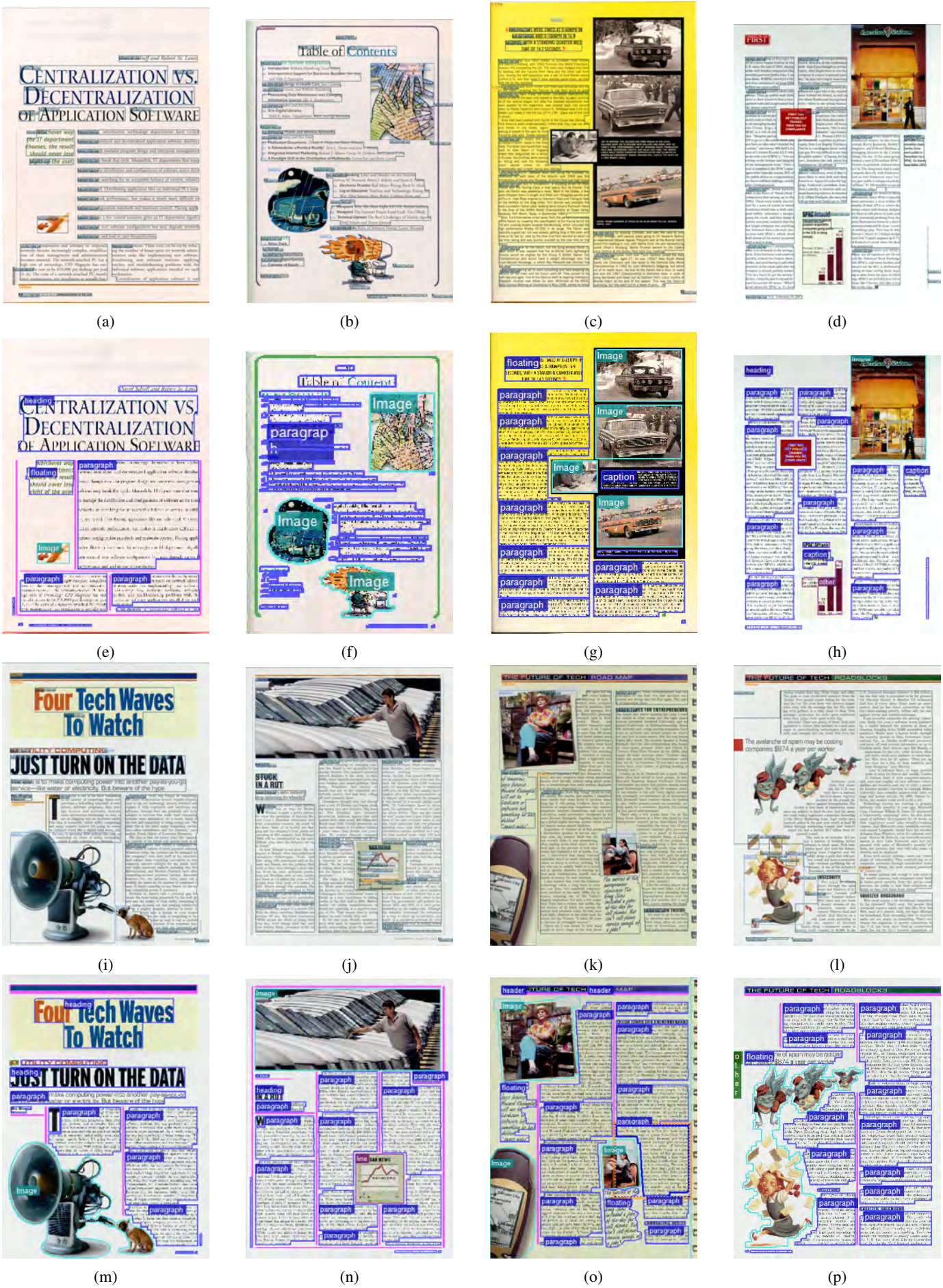
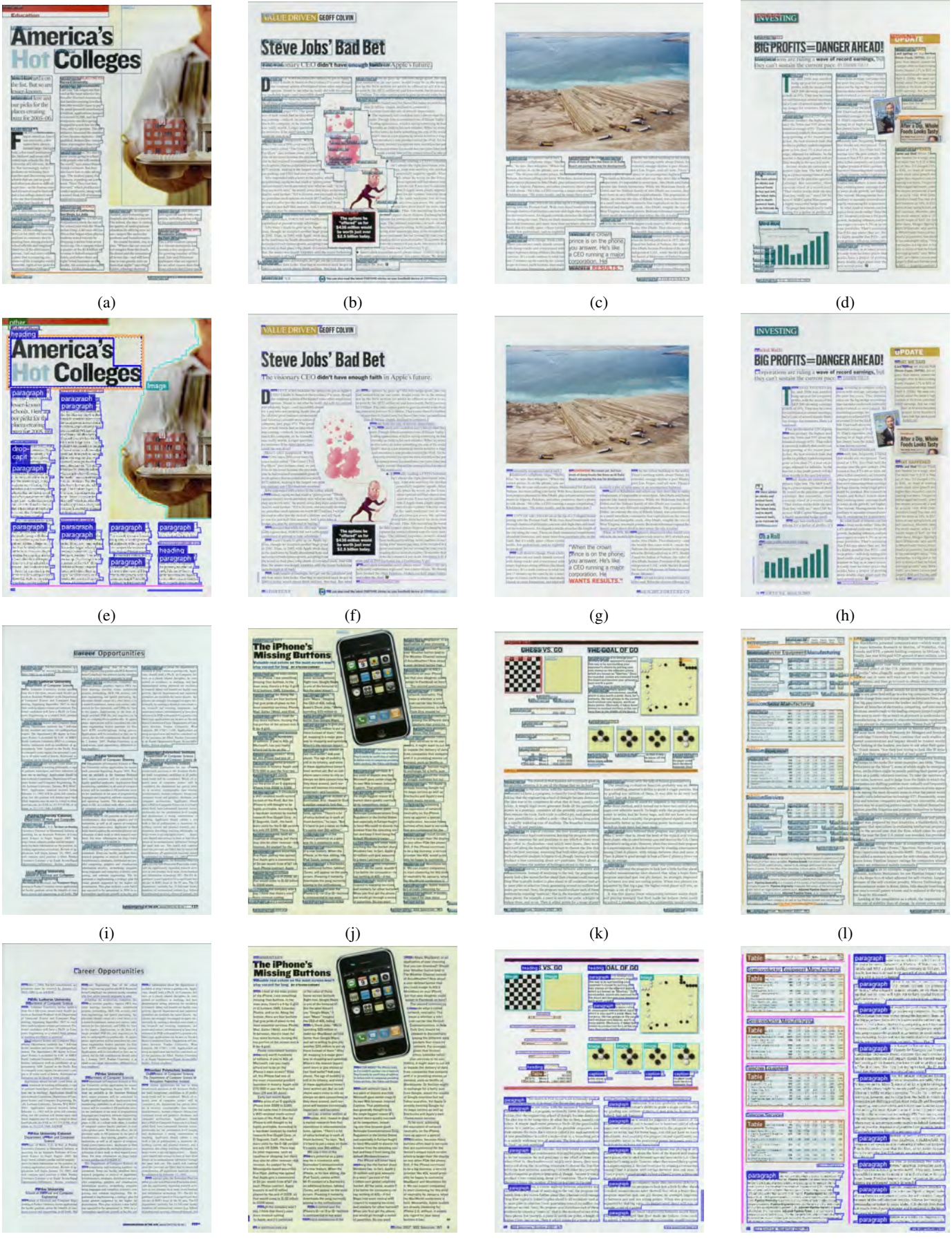


Fig. 21: Example 1 of results (a, b, c, d, i, j, k, l) and ground truths (e, f, g, h, m, n, o, p) comparison.



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

(e) (f) (g) (h)

(i) (j) (k) (l)

(m) (n) (o) (p)

Fig. 22: Example 2 of results (a, b, c, d, i, j, k, l) and ground truths (e, f, g, h, m, n, o, p) comparison.

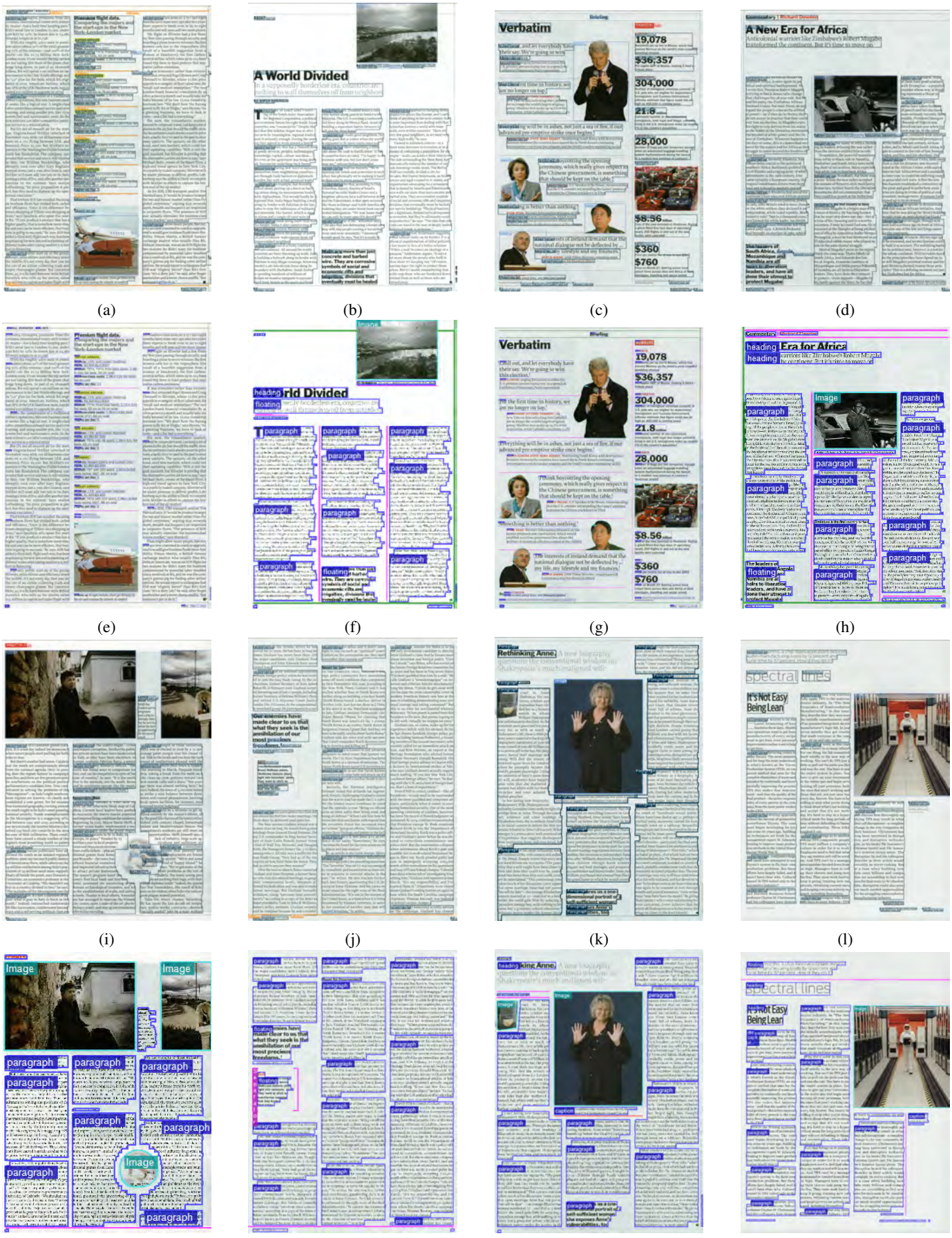


Fig. 23: Example 3 of results (a, b, c, d, i, j, k, l) and ground truths (e, f, g, h, m, n, o, p) comparison.