CVC-UAB's participation in the Flowchart Recognition Task of CLEF-IP 2012

Marçal Rusiñol, Lluís-Pere de las Heras, Joan Mas, Oriol Ramos Terrades, Dimosthenis Karatzas, Anjan Dutta, Gemma Sánchez, and Josep Lladós

Computer Vision Center, Dept. Ciències de la Computació,
Edifici O, Universitat Autònoma de Barcelona,
08193 Bellaterra (Barcelona), Spain.
{marcal, lpheras, jmas, oriolrt, dimos, adutta, gemma, josep}@cvc.uab.cat

Abstract. The aim of this document is to describe the methods we used in the flowchart recognition task of the CLEF-IP 2012 track.
The flowchart recognition task consisted in interpreting flowchart line-drawing images. The participants are asked to extract as much as structural information in these images as possible and return it in a predefined textual format for further processing for the purpose of patent search. The Document Analysis Group from the Computer Vision Center (CVC-UAB) has been actively working on Graphics Recognition for over a decade. Our main aim in participating in the CLEF-IP flowchart recognition task is to test our graphics recognition architectures on this type of graphics understanding problem.
Our recognition system comprises a modular architecture where modules tackle different steps of the flowchart understanding problem. A text/graphic separation technique is applied to separate the textual elements from the graphical ones. An OCR engine is applied on the text layer while on the graphical layer identify with nodes and edges as well as their relationships. We have proposed two different families of node and edge segmentation modules. One dealing with the raw pixel data and another working in the vectorial domain. The locations of nodes identified are fed to the recognizer module which is in charge of categorizing the node's type. We have proposed two different node descriptors for the recognizer module. The module analyzing the edges is analysing the connections between nodes and categorizes the edge style. Finally, a post-processing module is applied in order to correct some syntactic errors.
We have submitted four different runs by combining the two variants of the segmentation module together with the two variants of the recognition module.

Keywords: Flowchart Recognition, Text/Graphics Separation, Raster-to-Vector Conversion, Symbol Recognition.

1 Introduction

Graphics Recognition can be defined as the branch of document analysis that focuses on the recovery of graphical information from documents. Graphical

documents such as engineering drawings, diagrams, architectural floor plans, maps, etc. strongly depend on diagrammatic notations defined in terms of a visual language. A Graphics recognition system can be structured in three levels. First, a lexical level aims to extract basic infomation components such as graphical primitives (straight lines, arcs, solid areas) and text zones. The second level, the syntactic level, is related to the structure, i.e. how graphical entities are constructed. It can be modeled by a grammar describing the valid instances of diagrams according to the notation. The recognition involved in the syntactic level locates the graphical entities and classifies them into various classes by their shape and context. Finally, a functional or semantic level defines what the syntactically recognized entities mean in the context where they appear.

In this paper, we describe the flowchart recognition system that we proposed in the context of the Intellectual Property (IP) track at CLEF 2012. The IP track was first organized in 2009 as a prior art retrieval task. In 2010 and 2011, the track continued as a benchmarking activity of the CLEF conference and in 2011 two image-based tasks were added. One devoted to find patent documents relevant to a given patent document containing images and another aimed at categorizing given patent images into predefined categories of images (such as graph, flowchart, drawing, etc.). In CLEF-IP 2012 a new image-based task was proposed. The flowchart recognition task deals with the interpretation of flowchart line-drawing images. The participants are asked to extract as much structural information as possible in these images and return it in a predefined textual format for further processing for the purpose of patent search.

The Document Analysis Group from the Computer Vision Center (CVC-UAB) has been actively working on the Graphics Recognition research topic and in this notebook paper, we describe our submitted flowchart recognition system. The rest of this paper has been organized as follows. We detail in Section 2 the architecture overview and summarize the submitted runs. Section 3 details each of the modules that comprises the system's pipeline. In section 4 we present and analyze the obtained results and finally in section 5 we draw our concluding remarks.

2 Architecture Overview

The architecture of our recognition system has been designed as follows. As we can see in Fig. 1, the system pipeline has been structured in separate modules dealing with the different steps of the flowchart identification problem.

First of all, a text/graphic separation module has been applied to separate the textual elements from the graphical ones. An OCR engine has been applied on the text layer while on the graphical layer we analyze the nodes and edges. For nodes/edge segmentation we have applied two different strategies resulting in two alternative segmentation modules: a pixel-based and a vector-based approach. The vector-based approach requires a conversion module that transforms the raw pixel image into a vectorial representation.

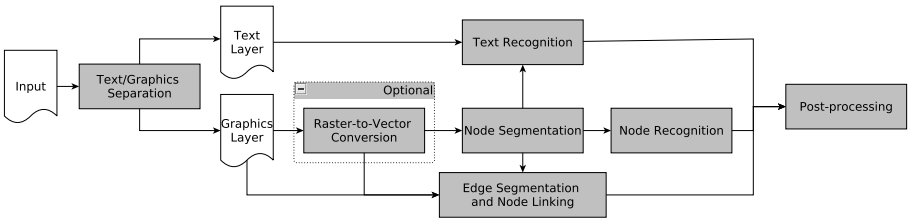


Fig. 1. System's architecture overview.

The output of the node segmentation module is a list of bounding-boxes of the detected nodes. These locations were subsequently fed to the recognizer module which is in charge of establishing the node's type. Here we have used two different node descriptors, namely a descriptor based on the geometric moments and the Blurred Shape Model (BSM) descriptor [1], resulting in two alternatives for the node recognition module. The module analyzing the edges is devoted to assess which nodes are connected and classify the edges in terms of their style..Finally, a post-processing module has been applied in order to correct certain *syntactic* errors.

The combination of the two alternative segmentation modules with the two alternative recognition modules results to four system variants producing the four submitted runs summarized in Table 1.

Table 1. Submitted runs produced by the four different system variants.

Id	Run	Segmentation	Descriptor
*R*1	IP-FR-CLEF2012.CVC-UAB.GMOMENTS	Pixel-based	Geometric moments
*R*2	IP-FR-CLEF2012.CVC-UAB.BSM	Pixel-based	BSM
*R*3	IP-FR-CLEF2012.CVC-UAB.VECTORIALGMOMENTS	Vectorial-based	Geometric moments
*R*4	IP-FR-CLEF2012.CVC-UAB.VECTORIALBSM	Vectorial-based	BSM

3 Module Description

Let us detail in this Section each of the modules composing our architecture.

3.1 Text/Graphics Separation

The first step in the proposed flowchart recognition architecture is to separate text elements from graphical elements. Within the Document Image Analysis research field, various text/graphics separation algorithms have been developed over many years. One of the most used methodologies that yields acceptable results in a variety of mixed-type documents is the algorithm proposed by Tombre et al. in [6] based on the well-known approach of Fletcher and Kasturi in [2].

需要去拜读

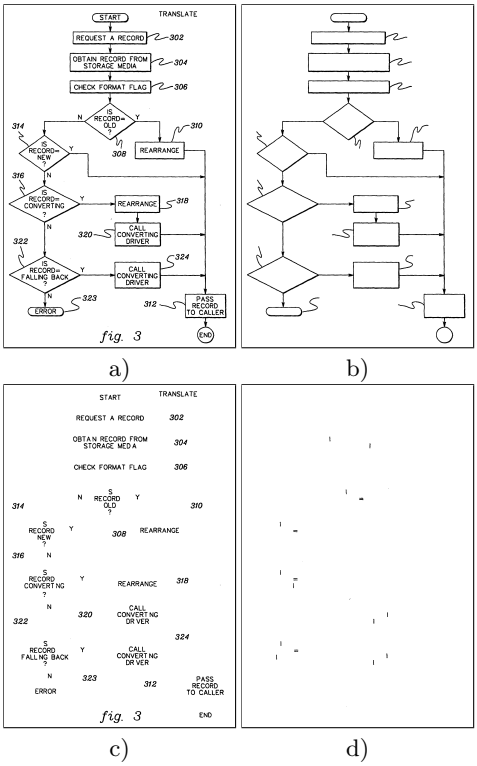


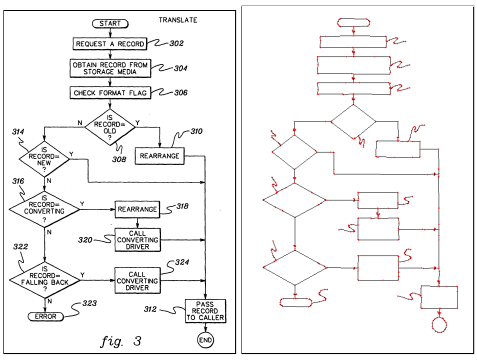
Fig. 2. Example of the text/graphics separation module. a) Original image, b) graphical layer, c) textual layer, d) undetermined layer.

This module proceeds as follows, given an input image, features are extracted from connected components (height and width ratios, orientation, etc.). Then, several adaptive thresholds determine whether a connected component corresponds to a graphical element or a textual element. The connected components that the system is not certain about are assigned to an undetermind layer via a rejection criterion. The output of the module is an image consisting of three separate layers, namely the graphical, the textual and the undetermined one. We can see an example of the results obtained by this text/graphics separation module in Fig. 2.

3.2 Raster-to-vector Conversion

Some of the submitted runs are based on the analysis of vectorial data instead of working on the raw image domain. Thus we need a module that converts the raster image into a vectorial representation. We have used the raster-to-vector conversion described in [5] by Tombre et al. This algorithm first computes the skeleton of the image by means of a distance transform. Chains of pixels are then polygonally approximated by using the algorithm proposed by Rosin and West in [4] which is based on a recursive split-and-merge technique. We can see an example of the obtained results after the raster-to-vector conversion in Fig. 3.

需要阅读



a) b)

Fig. 3. Example of the raster-to-vector conversion module. a) Original image, b) vectorial representation applied to the graphical layer.

3.3 Node Segmentation

We have submited two different alternatives for the node segmentation module depending on the primitives used to segment nodes and edges from the flowchart images. On one hand we have two system variants based on a node segmentation module that works with the raw pixels from the graphical layer and on the

other hand we propose another segmentation based on the analysis of vectors arisen from the raster-to-vector conversion of the graphical layer. In both cases, the segmentation strategy first attempted to segment *symbol* nodes (i.e. rectangles, ovals, diamonds, circles, etc.) and then tried to segment the *no-box* nodes, corresponding to text which is not enclosed by any graphical entity.

Pixel-based Node Segmentation

The node segmentation module takes as input the graphical layer obtained through the text/graphics separation module and outputs a list of bounding-boxes where the nodes might be found. The pixel-based approach analyses connected components (CCs) to determine whether they might be a node or not. A CC is a maximal set of spatially connected pixels that share a common property (here, the same colour). In the current system connected components of interest are the ones corresponding to the white area inside the nodes and the main problem is to discriminate those from connected components corresponding to the background. After a preliminary step where very small and very large CCs are filtered, the remaining CCs are labeled as node candidates. Here, not all the remaining components correspond to nodes since the white areas produced by loops formed by edges connecting nodes are also detected as candidates. In order to discriminate the real node components from the rest we set device heuristics over a couple of features:

- **Solidity**: computed as the ratio between the number of pixels in the CCs and the area of the convex hull of the CCs. Since nodes tend to be convex, objects below a solidity threshold are rejected.
- **Vertical Symmetry**: computed as the ratio between the amount of pixels in the right and the left part of the CCs. Since nodes tend to be vertically symmetric, objects below a symmetry threshold are rejected.

The remaining CCs after this filtering are considered as nodes and the bounding-box of each element is provided. Going back to the textual layer, once the symbol nodes have been detected, we can easily extract the no-box nodes as the text clusters that do not fall within any detected symbol node. We can see an example of the node segmentation output in Fig. 4 b). This segmentation strategy has been used in the runs $R1$ and $R2$.

Vectorial-based Node Segmentation

Distinctively from pixel-based segmentation, the Vectorial-based Node Segmentation takes as input the vectorial representation of the images obtained after the Raster-to-vector Conversion, see Fig. 3 b). It is based on the exploration of loops in the planar graphs obtained from the vectorial images. In these graphs, nodes are the vectorial lines and edges are the connection points between these lines. The seek for loops is driven by the implementation of the optimal algorithm for finding regions in a planar graph from Jiang et al. in [3]. As in the

pixel-based segmentation, the heuristic processes based on Solidity and Symmetry presented above are followed to rule out inconsistent instances. Finally, the system outputs the bounding-boxes for the remaining nodes. The no-box nodes are determined by looking at textual clusters connected to shape nodes via an edge. This approach is implemented in runs $R3$ and $R4$.

3.4 Node Recognition

The module devoted to recognize the nodes' types takes as input any of the two proposed node segmentations and tries to classify the node to one of the possible classes based on its shape. We have proposed two different shape recognizers.

Pattern recognition systems usually require shape descriptors invariant to rigid (scale, translation, rotation) and even affine transforms. However, in the context of flowchart recognition, the shape descriptors needed to recognize the nodes' type just need to be invariant to translation and scale, while a lack of invariance to other transforms is actually beneficial as it results to increased discrimination.

Translation invariance is not an issue since both node segmentations localised node areas. Both shape descriptors proposed are invariant to scale. In order to assign a node label a nearest neighbor classifier is used over a training set of labeled nodes. Let us detail the two different descriptors we used.

The Blurred Shape Model descriptor (BSM) was originally created to perform handwritten musical score recognition but it has also been applied to other related document analysis with different degrees of success [1]. The BSM descriptor is a zoning-based descriptor. Shapes are divided into a 15×15 regular grid. Then, the area and the center of gravity are computed for each cell. The final BSM descriptor is constructed by weighting the areas computed by the inverse of distances between two gravity centers of adjacent cells. This weighted average is performed in order to achieve robustness to local variations of the shape under analysis. A normalization by the object area is used in order to achieve invariance to scale. This shape recognizer is used in the runs $R2$ and $R4$.

Geometric Moments have been widely used as shape descriptors since lower order moments represent certain well known fundamental geometric properties of the underlying image functions. The central $(p + q)$-th order moment for a digital image $I(x, y)$ is expressed by

$$\mu_{pq} = \sum_{x,y} (x - \bar{x})^p (y - \bar{y})^q I(x, y) \tag{1}$$

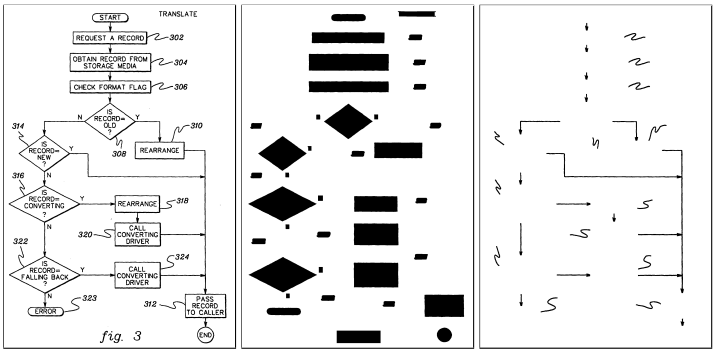
The use of the centroid $(\bar{x}, \bar{y})$ allow the descriptor to be invariant to translation. A normalization by the object area is used to achieve invariance to scale.

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\gamma}} \quad \text{where} \quad \gamma = \frac{p+q}{2} + 1 \tag{2}$$

We have used the geometric moments up to third order as a feature vector. This shape recognizer is implemented in the runs $R1$ and $R3$.

3.5 Edge Segmentation and Node Linking

Edge segmentation has been done in a similar fashion in both the pixel-based and the vectorial-based approaches. Taking as input both the graphical layer and the bounding-boxes from the node segmentation we obtain an edge layer image. We can see an example in Fig. 4.



a) b) c)

Fig. 4. Example of the node and edge segmentation modules. a) Original image, b) node layer, b) edge layer.

In order to assess which nodes are connected by an edge we pair-wisely select connected components in the node layer and check whether any element of the edge layer provokes that those two disjoint components merge into a single one. An analysis of the edge stroke's width is performed to differentiate between directed and undirected edges. Since our node linking modules strongly depend on the extraction of connected components, they fail at detecting the *dotted* edges. The pixel-based approach fails if the edge is broken by some text but the vectorial-based one performs a post-processing step that merges colinear broken edges making it robust to such situations.

3.6 Text Recognition

The module dealing with text recognition receives as input the textual layer obtained by the text/graphics separation module and the bounding-boxes arisen from the node segmentation module. The contents of each bounding-box are processed by the commercial OCR from ABBYY[1], and no further post-processing steps are applied. Taking a look at the raw output from the OCR, some simple

[1] ABBYY Finereader Engine 10 `"http://www.abbyy.com/ocr_sdk/"`

steps for dealing with out-of-vocabulary elements would drastically improve the performance of this module.

3.7 Post-processing

Testing our architecture against the training data we realized that when two edges collide, in the ground-truth we expect to see a new node of type *point*. Obviously our node segmentation and recognition do not tackle such nodes since they can not be found via a connected component analysis. We decided to include the detection of those nodes in a post-processing step.

The final graph is syntactically analyzed and when we find three or more nodes that are connected through exactly the same edge element, we add a new intermediate node of type *point*. This procedure is iteratively done since there are no node junctions with cardinality higher than two.

4 Evaluation

Let us first evaluate the proposed system's performance qualitatively by looking at specific cases where the we face some problems. Let us then present the obtained results.

4.1 Qualitative Evaluation

Although visually the system seems to perform quite well, we have identified several cases where the proposed modules fail, examples of which are presented in Fig. 5.

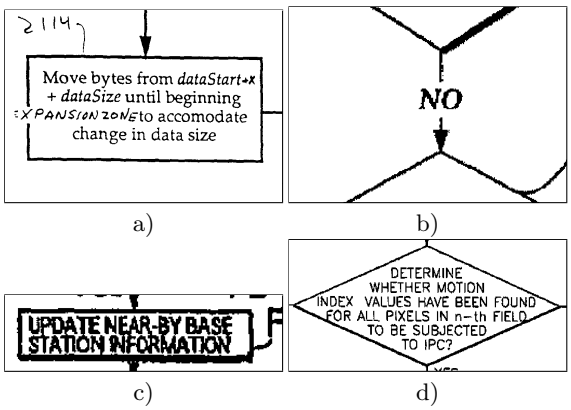


Fig. 5. Problematic cases. a) Broken node, b) broken edge, c) Low-quality text, d) Text/graphics overlapping.

First of all, the node and edge segmentations are based on connected components. When either a node (Fig. 5a)) or an edge (Fig. 5b)) is broken it is usually not well segmented by any of the proposed methods. In addition, low-quality documents are hard to "read" by the OCR engine (Fig. 5c)). Finally, the text/graphics separation module is also based on the analysis of connected components, so when text characters overlap with graphical elements, they are not properly segmented. This is the case shown in Fig. 5d), where the character *F* of *FROM* and the character *D* from *FIELD* touch the diamond shape and thus are classified as graphics and assigned to the graphical layer.

5 Conclusions

In this paper, we presented the flowchart recognition system that the Document Analysis Group from the Computer Vision Center (CVC-UAB) has submitted in the context of the Intellectual Property (IP) track at CLEF 2012. The architecture of our recognition system is structured in different modules dealing with different steps of the flowchart identification problem.

First of all, a text/graphic separation technique has been applied to separate the textual elements from the graphical ones. An OCR engine has been applied on the text layer while on the graphical layer we analyze the nodes and edges. For nodes/edge segmentation we have applied two different strategies: a pixel-based and a vector-based approach. The node locations have been fed to the recognizer module which is in charge of categorizing the node's type. Here we have used two different node descriptors, namely a descriptor based on the geometric moments and the Blurred Shape Model (BSM) descriptor. The module analyzing the edges determines which nodes are connected and categorizes the edge style. The combination of the two segmentation modules together with the two recognition modules results to the four system variants and an equal number of submitted runs.

Acknowledgment

This work has been partially supported by the Spanish Ministry of Education and Science under projects RYC-2009-05031, TIN2011-24631, TIN2009-14633-C03-03, Consolider Ingenio 2010: MIPRCV (CSD200700018) and the grant 2009-SGR-1434 of the Generalitat de Catalunya.

References

1. S. Escalera, A. Fornés, O. Pujol, P. Radeva, G. Sánchez, and J. Lladós. Blurred shape model for binary and grey-level symbol recognition. *Pattern Recognition Letters*, 30(15):1424–1433, November 2009.
2. L.A. Fletcher and R. Kasturi. A robust algorithm for text string separation from mixed text/graphics images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(6):910–918, November 1988.

3. X.Y. Jiang and H. Bunke. An optimal algorithm for extracting the regions of a plane graph. *Pattern Recognition Letters*, 14(7):553 – 558, 1993.
4. P.L. Rosin and G.A. West. Segmentation of edges into lines and arcs. *Image and Vision Computing*, 7(2):109–114, May 1989.
5. K. Tombre, C. Ah-Soon, P. Dosch, G. Massini, and S. Tabbone. Stable and robust vectorization: How to make the right choices. In *Graphics Recognition Recent Advances*, volume 1941 of *Lecture Notes in Computer Science*, pages 3–18. Springer-Verlag, 2000.
6. K. Tombre, S. Tabbone, L. Pélissier, B. Lamiroy, and P. Dosch. Text/graphics separation revisited. In *Document Analysis Systems*, volume 2423 of *Lecture Notes in Computer Science*, pages 615–620. Springer-Verlag, 2002.