Stable, Robust and Off-the-Shelf Methods for Graphics Recognition

Karl Tombre, Christian Ah-Soon, Philippe Dosch, Adlane Habed, Gérald Masini
Loria–CNRS–INRIA–UHP
B.P. 239, 54506 Vandœuvre-lès-Nancy CEDEX, France
Karl.Tombre@loria.fr

Abstract

We claim that time has come in graphics recognition for choosing stable and robust methods, even—or especially—when this means implementing methods proposed by others, instead of inventing a new algorithm which ends up being a minor variation on an old idea. In this spirit, we present some of the choices our own team has made.

1. Introduction

Our research group has been investigating various aspects of graphics recognition techniques for more than ten years, designing a number of methods for low-level processing, symbol recognition and high-level interpretation. Of course, we have also implemented various algorithms proposed by other researchers, when appropriate. However, it has been our unpleasant experience that it is very difficult to *reuse* software components over a larger period. In addition, the recent emphasis in the whole research community on performance assessment and evaluation leads to the necessity of being able to precisely describe the qualities of a given module, and, if necessary, to replace it with a better version, without endangering the whole system.

All this has led us to start a "consolidation" activity in our group, by taking up again the best methods we have developed, or which others have developed, and designing a set of stable and robust methods, well-implemented as C++ classes, so as to allow for later reuse and stabilized behavior.

This paper is about our choices, when it comes to robust methods in graphics recognition applications. We do not claim that there are no other possible methods, nor that there is no room for improvement through further research in these areas. But we definitely believe that a lot of effort, in many research groups, is spent "reinventing the wheel", maybe with minor variations, whereas surprisingly little work is done on several issues which can be considered as major challenges.

In this paper, we do not cover the whole range of methods and steps used in graphics recognition, but rather concentrate on those which we deem the most crucial and time-consuming—in terms of development time—in the design of a graphics recognition system. Instead of developing our own special brand, we have aimed, whenever possible, at choosing "off the shelf" the methods best suited to our purpose, with few, well-defined parameters to meet the robustness criterion, and with a stable implementation.

In graphics recognition, as in many other image analysis applications, we often end up with lots of *ad hoc* thresholds. In most cases, these thresholds are fixed in a very empirical way. As a first step towards robustness, our aim has therefore been to have as few thresholds as possible. In addition, whenever a threshold becomes necessary, we aim at having straightforward relations between this threshold and the physical properties of the document. Our ultimate aim is to have a complete toolbox of well-specified, robust methods, so that we can concentrate on new problems.

2. Binarization

Built-in binarization of most scanners is sufficient for clean documents. But, when dealing with blueprints or large drawings which have been folded and stored away for a long period, we may have to use some adaptive binarization method. Such a method is computationally costlier than the built-in tools, but is necessary to avoid false objects due to folds, or to the merging of lines close to each other on a poor-quality blueprint.

Several methods have been proposed for that. Trier and Jain's goal-directed evaluation [7] is especially interesting, although it is mainly aimed at text-rich documents, as the evaluation is based on the performances of character recognition on the resulting binary image.

Basically, adaptive thresholding methods can be divided into two classes: Methods based on the computation of a local threshold from measures such as gliding averages, and methods based on finding some contours and filling the contours of the "black" objects.

Basing ourselves on Trier and Jain's evaluation, we implemented one method from each of these two categories: Niblack's method [4] with Yanowitz and Bruckstein's post-processing step [11] for the local average approach, and Trier and Taxt's improvement on a method originally proposed by White and Rohrer for the contour approach [8]. Probably because of the special nature of graphical documents, the latter yields much better results.

We therefore chose to implement a variation of Trier and Taxt's method. Our main changes concern the use of Gaussian filtering, which has become standard in edge detection, instead of *ad-hoc* filters such as the Sobel gradient.

There are three thresholds in this method: An activity threshold T_A, a post-processing threshold T_P, for which there are unfortunately no clear default values, and the width of the Gaussian, i.e. σ. Our experiments show that the most important parameter is the latter, which must be chosen such that the convolution masks have approximately the same width as the thickest lines in the image.

As we want to deal with robust methods, let us stress again that whenever the document is clean, using the built-in binarization coming with the scanner software is the best choice! The method described above is only useful when the degradations make this binarization useless.

3. Text/graphics segmentation

Many papers have been published on text/graphics separation, but we don't feel that there are that many different ideas around. When it comes to the basic principles, most methods end up being small variations on well-known themes, and we know how to separate graphics from text when they are not touching each other. The basic idea is to analyze the connected components.

One of the best explained methods in literature is that of Fletcher and Kasturi [2]. We therefore strongly suggest that, instead of spending a lot of time on reinventing some new algorithm—which, most of the time, does not give any real improvements on the known methods—teams do use this method, changing if necessary some of the parameters to reflect the specificities of the documents to be processed. Then, they will get more time left to concentrate on the really difficult problem of separating touching text and graphics, where only partial solutions have been proposed.

As noted, we chose to implement Fletcher and Kasturi's method. As they designed their method for mixed text/graphics documents, some of the thresholds must be adapted to the new situation. We also added an absolute threshold on the size of a text component. Thus, we end up having three thresholds, but their interpretation is straightforward, and they have proven to be very stable for a family of graphics documents. We therefore suggest that people using this algorithm find out which values are the best for their application, and keep these values for all images.

As proposed by Fletcher and Kasturi, this is followed by string grouping using the Hough transform.

4. Thin/thick separation

We sometimes need to make further segmentation of the graphics part, by finding different thickness classes. For this, we have used morphological filters. Let I be the image of the graphics part. According to the limit we want to set between thin and thick lines, we set a size n and perform an erosion $J = I \ominus B_n$, B_n being in our case a $(2n+1) \times (2n+1)$ square. The thick lines can then be retrieved lines through partial geodesic reconstruction ($n+1$ iterations):

$$K_0 = J; K_i = (K_{i-1} \oplus B_1) \cap I \quad \text{for} \quad i = 1 \ldots n+1$$

This yields two images $I_{thick} = K_{n+1}$ and $I_{thin} = I - I_{thick}$.

5. Vectorization and arc recognition

Vectorization, i.e. raster-to-graphics conversion, has been given a lot of attention, and many algorithms have been proposed. There are also a number of commercial packages which perform some kind of vectorization. Most methods are based on the combination of a skeletonization algorithm, followed by some kind of polygonal approximation. Other methods are also available, including various sparse-pixel approaches, run-based algorithms and approaches directly working on the image or on the distance transform. Although these methods yield good results, they all have their specific weaknesses, so that we cannot say that perfect raster-to-vector conversion is available. However, the quality is good enough to use the result as input data for higher-level recognition and analysis methods. Various interesting post-processing steps have also been proposed, to enhance the quality of the vector description.

We have experimented several algorithms, having interesting properties and yielding good results. However, because of stableness and robustness criteria, we finally came back to what most teams use: Some kind of thinning, followed by polygonal approximation. This stems from the fact that this approach requires the lowest number of parameters to be set; with other approaches, we often had to fine-tune our parameters for each new family of documents.

This does not mean that there is no room for improvements! On the contrary: Surprisingly, skeletonization has mostly been done using iterative thinning (the paradigm of "peeling an onion"), despite the well-known problems of small barbs and junction distortion. But there has been other algorithms around for a long time, especially those

based on the distance transform. We have therefore implemented a skeletonization method, first proposed by Sanniti di Baja [1], and based on the 3–4 distance transform. This is followed by polygonal approximation; after having used Wall and Danielsson's iterative algorithm [10] for many years, we have now switched to an algorithm which essentially works *without any threshold*, that of Rosin and West [6], based on a previous algorithm proposed by Lowe, and which also allows us to recognize arcs subsequently.

We are currently developping a post-processing method to better position the junction points, using a method proposed by Janssen [3], and we aim at adding geometric constraints to the vectorization and arc recognition, using a method proposed by Röösli and Monagan [5].

6. Tiling

It is often impossible to process the whole grey level image produced by a high-resolution scanner at the same time, especially when dealing with large drawings in mechanical engineering or architecture. When appropriate, we therefore divide the image into smaller rectangular tiles, which can be separately processed with lower memory requirements. The features provided by vectorization performed on the different tiles are then merged together.

The image is first divided into tiles, all having the same width and length. Once the vectorization has been performed on the set of tiles, the corresponding vectors are divided into several pieces belonging to different neighbouring tiles. In order to be able to efficiently re-assemble these pieces, all the tiles partially overlap. The width W of the overlapping zone is chosen from the estimated maximum thickness T_{max} of the line drawings, which is supposed to be initially known, as a line drawing cannot be correctly skeletonized if it is not entirely included into a tile [9].

The segments belonging to overlapping zones are finally matched. As the initial amount of data is significantly reduced thanks to the vectorization, this can be handled globally. The matching process is based on computing the Hausdorff distance between each segment in a tile and its candidate segments for matching in neighbouring tiles. The process is simple and robust, and yields good results, provided text/graphics segmentation has been correctly performed.

7. Conclusion

We have tried to prove that there are enough methods around for the basic processing tools of a graphics recognition system. *Please don't invent new ones!* Or more precisely: You should have very good reasons for designing a new algorithm in one of these areas. However, there is a real need for finding the right tool, doing the right thing. When picking up tools from the shelf, we should make sure that we choose methods as *robust* and *stable* as possible.

We do not claim that we have found the ultimate toolbox. But if this paper can give incentives to other teams to work in this spirit, we hope that our community can progressively come up with a set of mature methods, for which some reference implementation could even be made available to the whole community.

Of course, we are aware of several lacks in what we have done until now. We need methods to characterize the behaviour of the algorithms, to determine as automatically as possible the thresholds we use, and to analyze the influence one threshold has on the others. Since many methods are still developed and tested on a limited number of drawings, the validation scope should be significantly extended, so as to be sure that the methods are robust enough. The problem of precision is also crucial, especially regarding vectorization algorithms.

References

[1] G. S. di Baja. Well-Shaped, Stable, and Reversible Skeletons from the (3,4)-Distance Transform. *Journal of Visual Communication and Image Representation*, 5(1):107–115, 1994.

[2] L. A. Fletcher and R. Kasturi. A Robust Algorithm for Text String Separation from Mixed Text/Graphics Images. *IEEE Transactions on PAMI*, 10(6):910–918, 1988.

[3] R. D. T. Janssen and A. M. Vossepoel. Adaptive Vectorization of Line Drawing Images. *Computer Vision and Image Understanding*, 65(1):38–56, Jan. 1997.

[4] W. Niblack. *An Introduction to Digital Image Processing*, pages 115–116. Prentice Hall, 1986.

[5] M. Röösli and G. Monagan. Adding Geometric Constraints to the Vectorization of Line Drawings. In R. Kasturi and K. Tombre, editors, *Graphics Recognition—Methods and Applications*, volume 1072 of *Lecture Notes in Computer Science*, pages 49–56. Springer-Verlag, May 1996.

[6] P. L. Rosin and G. A. West. Segmentation of Edges into Lines and Arcs. *Image and Vision Computing*, 7(2):109–114, May 1989.

[7] Ø. D. Trier and A. K. Jain. Goal-Directed Evaluation of Binarization Methods. *IEEE Transactions on PAMI*, 17(12):1191–1201, Dec. 1995.

[8] Ø. D. Trier and T. Taxt. Improvement of "Integrated Function Algorithm" for Binarization of Document Images. *Pattern Recognition Letters*, 16:277–283, Mar. 1995.

[9] A. M. Vossepoel, K. Schutte, and C. F. P. Delanghe. Memory Efficient Skeletonization of Utility Maps. In *Proceedings of 4th International Conference on Document Analysis and Recognition, Ulm (Germany)*, pages 797–800, Aug. 1997.

[10] K. Wall and P. Danielsson. A Fast Sequential Method for Polygonal Approximation of Digitized Curves. *Computer Vision, Graphics and Image Processing*, 28:220–227, 1984.

[11] S. D. Yanowitz and A. M. Bruckstein. A New Method for Image Segmentation. *Computer Vision, Graphics and Image Processing*, 46(1):82–95, Apr. 1989.