Object Building and Positioning to Generate Synthetic Graphical Documents

Mathieu Delalandre
CVC, Barcelona, Spain
mathieu@cvc.uab.es

Tony Pridmore
SCSIT, Nottingham, England
tony.pridmore@nottingham.ac.uk

Eric Trupin
LITIS, Rouen, France
eric.trupin@univ-rouen.fr

Hervé Locteau
LITIS, Rouen, France
herve.locteau@univ-rouen.fr

Ernest Valveny
CVC, Barcelona, Spain
ernest.valveny@uab.es

Abstract

In this paper we present a system to build synthetic graphic documents for performance evaluation using two main components. The first one performs a vectorial distortion on the objects without using any prior knowledge. The other position the objects in the documents using four filling algorithms allowing to preserve a partitioning between the objects and the document background. Experiments done on more of 300 000 symbols show that our system distort and fill in a logarithmic and gaussian way.

1. Introduction

In recent years there has been a noticeable shift of attention within the graphics recognition community to the topic of performance evaluation. Performance evaluation is divided into two main topics: ground-truthing and performance characterization. The first is concerned with the production of test document databases and their corresponding ground-truth [1], while the second deals with the matching of system results to that ground-truth [5]. In this paper we are more interested in ground-truthing. Two main approaches exist in the literature: based on real-life or on synthetic documents.

The approach based on real-life documents is the most common [1]. Representative documents are obtained from paper archives and digital libraries and ground-truth is created and edited manually using suitable graphic user interfaces. This kind of ground-truthing results in realistic and unbiased data but raises different problems: how to define the ground-truth, how to deal with errors in the ground-truth introduced by users, the delay and cost of ground-truth acquisition, the intellectual property of documents and effort required to constitute large databases. In many cases these problems render the approach impractical.

A complementary approach, which avoids these difficulties, is to create and use synthetic documents. Here, the test documents are built by an automatic system which combines pre-defined models of document components in a pseudo-random way. Test documents and ground truth can therefore be produced simultaneously. In addition, large numbers of documents can be generated easily and with limited user involvement. This topic is emerging and only [2] [5] [6] [4] exist in the literature. Figure 1 gives examples of the documents produced by these systems.

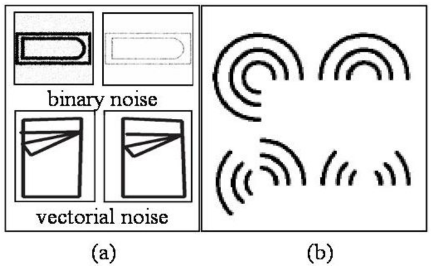


Figure 1. examples of synthetic document
(a) segmented symbol (b) arc set

The systems proposed by [6] and [4] support the generation of degraded images of segmented symbols (a). In these systems, the symbol models are described in a vector graphics format. The vector graphics files are then converted into images. Two kinds of noise are added: binary [6] [4] and vectorial [4]. The systems of [2] [5] are similar to previous ones but use dynamic models. These dynamic models are described in a mathematical formalism which allows the composition of various graphic shapes (b). In [2] two models are proposed to generate the land parcels and houses that appear in cadastral maps. In [5] another model is defined and used to generate images composed of several arcs of different lengths and radius. The arc images are then degraded by using a binary degradation algorithm.

All these systems are interesting, but in order to evaluate complete graphics recognition systems we need whole documents (eg. engineering drawings, architectural maps). Multiple objects must be created and connected to produce more complete and realistic documents. This process requires three system components:

Object Building uses topological and distortion algorithms to generate separate objects from models.

Object Positioning places the generated objects in the documents according to pre-defined spatial constraints.

Rule Management exploits domain rules in order to control the topological links between objects and connect them to generate realistic test documents.

The design of such a system is a challenging task. In this paper we focus on the first two components. Our system is summarised in Figure 2. It employs object building and positioning to create synthetic documents composed of multiple unconnected components. In particular, we have focused on vectorial distortion within the building phase and the filling strategies used during positioning. In what follows we present each of these two components in sections (2) and (3). In section (4) we describe initial experiments and the results they produced. Finally, in section (5) we conclude and give our perspectives.

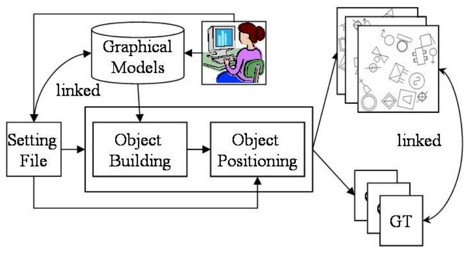


Figure 2. our system

2. Object Building

2.1. Modelling and Selection

Following the systems proposed by [6] and [4] we use geometrical primitives (straight line, arc and circle) and their associated thickness attributes to describe the component models. Each model is then stored in an individual file kept inside a database. The user accesses the contents of the database by defining in the setting file the models he/she wants to include in the test document and assigning a probability of inclusion to each. Models are then selected at random, taking these probabilities into account, to build the objects. Once selected, three kinds of transformations are applied to the models: classical rotation and scaling (set with two limits and a gap), and then a vectorial distortion (presented in next subsection).

2.2. Vectorial Distortion

Following selection, scaling and rotation operations our system applies vectorial distortion to the selected objects. In the literature only the work of [4] deals with this topic. This work proposes a system that supports learning, from sample documents, of probabilistic models of objects which take into account their variability. These models can then be used to build and to distort objects.

Basing distortion on real images provides a more realistic test set, but the work involved may be a significant proportion of that required to generate test data directly from real documents. In our system we have then adopted an opposed method which operates without any prior knowledge. While the proposed method does not produce “realistic” noise, however any learning step and any sample documents are needed. Our method is based on the generation of gaussian random numbers. As shown in Figure 3 (a), this generation consists of finding the v value from a uniformly distributed number s. The problem here is the that the gaussian function cannot be inverted analytically. So in order to solve it the common way described in the literature is to use the *Box-Muller* transformation to pass from the uniform distribution to the gaussian one.

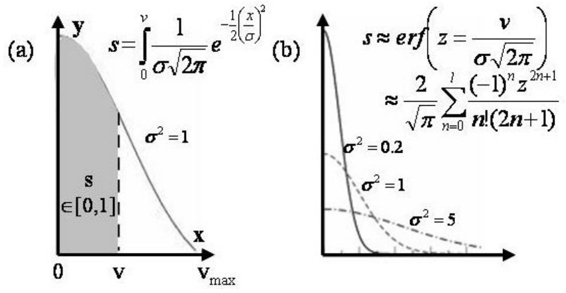


Figure 3. gaussian random numbers
(a) uniform distribution (b) no uniform distribution

This transformation allows efficient computation of gaussian random numbers but does not allow the user to set different variance values σ^2 (b). To solve this problem we compute an approximation of the gaussian sum by using the $erf(z)$ function[1]. This function can be computed in various ways but usually as a Maclaurin polynomial. We need next to solve $erf(z) \approx s$ in order to find the v value. This is difficult because to obtain a good approximation of

[1]http://mathworld.wolfram.com/Erf.html

s a high value of l (the order of the Maclaurin polynomial) is needed which involves a complex factoring step. In order to solve it we apply a dichotomic search algorithm on the x axis. Our experiments have shown that with a Maclaurin polynomial built with $l = 20$ the v values are obtained with 10^{-3} precision in less than 20 iterations.

Next, we use the gaussian random numbers in our distortion method. We believe that this distortion should be as low-level and domain independent as possible. We therefore apply it to the graphic primitives that compose the objects without taking into account any model-specific features. In our method we have considered the classic geometrical transformations (scaling, rotation, moving) set by distortion parameters. Figure 4 gives the functions $\{f_s(v), f_r(v), f_m(v)\}$ that we have defined to compute these parameters. The user can then select and combine in the setting file the types of distortion as well as the value of the variance for each type. Figure 4 gives some distortion examples obtained with different settings.

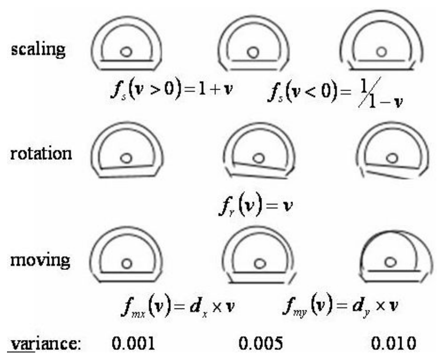


Figure 4. vectorial distortion

3. Object Positioning

3.1. Introduction

Following the building of objects we position them inside documents. This process starts with empty documents and fills them, in a pseudo-random way, with generated objects. The user defines, in the setting file, the size of the document and the number of objects that he/she wants to include in it. The process will stop when this number of objects is reached. However, it may be that this number cannot be reached without unrealistic levels of overlap between objects. So the system must be able to detect this and stop automatically. Indeed, as shown in Figure 5 a partitioning between objects and background must be respected in order to produce more "realistic" documents.

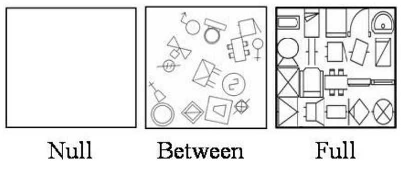


Figure 5. document filling

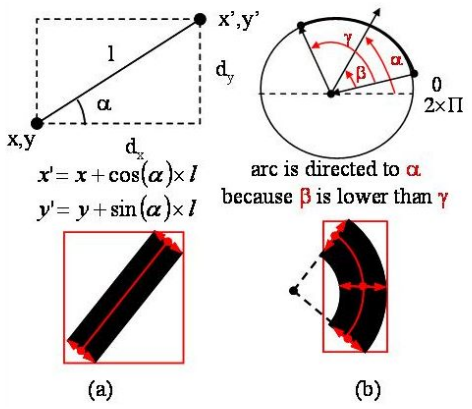
In order to realize such a process our key idea is to consider it as an array management problem. For that purpose we have defined four positioning algorithms inspired by those used in array management: *write*, *set*, *insert* and *push*. These algorithms write the objects in local areas of a given document without any management of its free space. As this process continues, the fragmentation of free space will make the positioning harder at every step and may eventually make it impossible. The stop criterion is therefore based on positioning failures. This approach allows us to preserve a partitioning between objects and document background. In what follows we present our algorithms in subsections 3.3 and 3.4, having first discussed how objects are handled by our algorithms in subsection 3.2. Finally, we discuss the stop criterion in subsection 3.5.

3.2. Object Handling

To handle the objects in the positioning algorithms we first compute their bounding boxes. Indeed, this is a common way to handle graphic objects in document analysis systems. Computing a bounding box from a set of straight lines is an easy thing to do. However, in our system in addition to straight lines we also use arcs and circles. Moreover, our primitives are given thickness attributes. In order to take into account these specificities we use the two methods presented in Figure 6. The first (a) is a classical projection method to move a point according to a given length and direction. This method allows us to find the corners of thick straight lines, and then to compute their bounding boxes. When processing arcs we also search for the corners but also the cardinal points (west, east, north and south). Indeed, these ones are necessary to compute an arc's bounding box (b). To do it we use a direction test which detects if right angles $\{0, \frac{\pi}{2}, \pi, \frac{3\times\pi}{2}\}$ belong to the arc.

3.3. The Write and Set Algorithms

We present here our two first positioning algorithms: *write* and *set*. They use two opposite approaches as shown in Figure 7. The *write* algorithm puts an object inside the document only if its target zone (ie. bounding box centered on a random position) is empty. Otherwise, it is deleted. In contrast, in the *set* algorithm all existing objects around the target zone are deleted. In both cases, these algorithms



x',y'
l
d_y
α
x,y
d_x
$x' = x + \cos(\alpha) \times l$
$y' = y + \sin(\alpha) \times l$
γ
β
α
0
2×Π
arc is directed to α
because β is lower than γ
(a)
(b)

Figure 6. Computation of bounding box
(a) point projection (b) direction test

test for overlaps between bounding boxes. This test is computed in three steps as explained in Figure 8: first between a line and a point (a), then between two lines (b) and at last between the two bounding boxes (c).

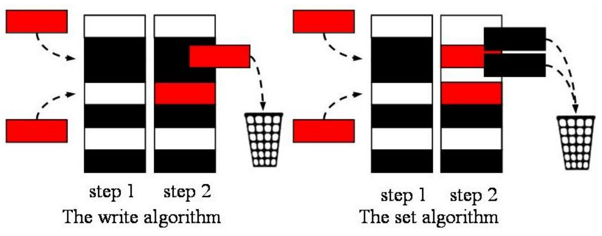
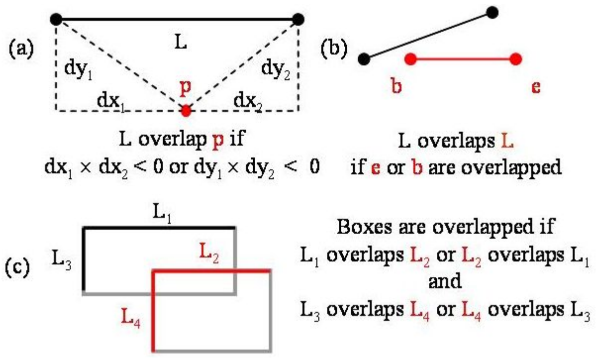


Figure 7. write and set algorithms



(a)
L
dy_1
dy_2
dx_1
dx_2
p
L overlap p if
$dx_1 \times dx_2 < 0$ or $dy_1 \times dy_2 < 0$
(b)
b
e
L overlaps L
if e or b are overlapped
(c)
L_1
L_2
L_3
L_4
Boxes are overlapped if
L_1 overlaps L_2 or L_2 overlaps L_1
and
L_3 overlaps L_4 or L_4 overlaps L_3

Figure 8. overlapping test
(a) line-point (b) line-line (c) box-box

3.4. The Insert and Push Algorithms

Our next two algorithms, *insert* and *push*, try to move objects in the document in order to satisfy constraints on positioning. To achieve this they use the competing approaches shown in Figure 9. In the case of the *insert* algorithm the current object is moved around its target zone in order to preserve the positioning of existing objects. On the contrary, the *push* algorithm moves the existing objects around the target zone of the current object. It therefore forces the positioning of the current object. In both cases, if objects can't be moved successfully they are deleted.

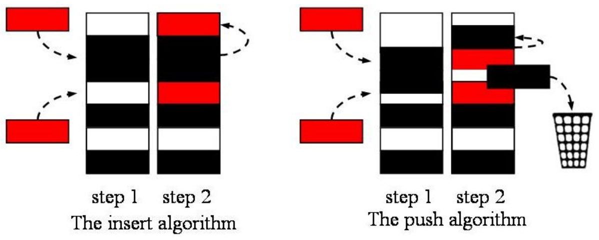
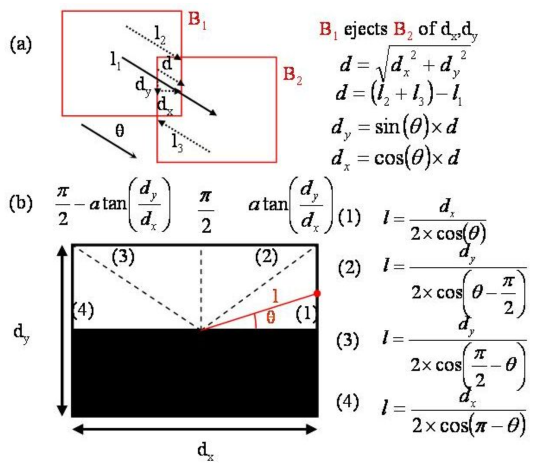


Figure 9. insert and push algorithms

To do this positioning the *insert* and *push* algorithms use the eject method presented in Figure 10. This computes the ejection parameters $\{d_x, d_y\}$ of a bounding box B_2 in order to move it to a free space (a). The computation is based on one of three line lengths $\{l_1, l_2, l_3\}$ joining the centres of gravity and borders of bounding boxes. The euclidean distance between the centres of gravity gives l_1 while $\{l_2, l_3\}$ are computed using the method described in (b): either in the θ direction for l_2, either in the $\theta + \pi$ direction for l_3. From the values of $\{l_1, l_2, l_3\}$ the value of d can be obtained. It is then used in addition to θ to compute the $\{d_x, d_y\}$ parameters as shown in (a).



(a)
B_1
B_2
l_1
l_2
l_3
d
d_y
d_x
θ
B_1 ejects B_2 of d_x,d_y
$d = \sqrt{d_x^2 + d_y^2}$
$d = (l_2 + l_3) - l_1$
$d_y = \sin(\theta) \times d$
$d_x = \cos(\theta) \times d$
(b)
$\frac{\pi}{2} - a\tan\left(\frac{d_y}{d_x}\right)$
$\frac{\pi}{2}$
$a\tan\left(\frac{d_y}{d_x}\right)$
(1) $l = \frac{d_x}{2 \times \cos(\theta)}$
(2) $l = \frac{d_y}{2 \times \cos\left(\theta - \frac{\pi}{2}\right)}$
(3) $l = \frac{d_y}{2 \times \cos\left(\frac{\pi}{2} - \theta\right)}$
(4) $l = \frac{d_x}{2 \times \cos(\pi - \theta)}$
(3)
(2)
(4)
(1)
l
θ
d_y
d_x

Figure 10. eject method
(a) computation of d_x, d_y (b) computation of l_2, l_3

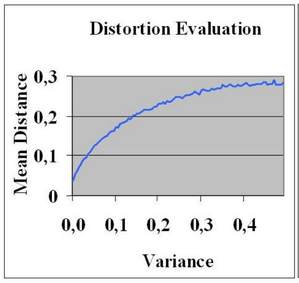
3.5. Stop Criterion

As explained in the subsection 3.1, should the user set constraints that are hard to satisfy it may be necessary to stop the positioning. To do this we use the number of objects per document (defined in the setting file) as stop criterion. We compare it with the number of building failures (ie. when a new positioning step doesn't increase the object number of document). When the failure number becomes larger we stop the process.

4. Experiments and Results

We present here our initial experiments and results about our system. These ones concern the evaluation of our distortion and positioning algorithms. To do it we have built databases of synthetic document to analyze their contents, using a model database composed of 150 electrical and architectural symbols [4]. The Figure 11 presents our results.

In a first step we have evaluated the noise levels produced by our distortion algorithm in regard to its setting. We have applied this algorithm from models without any scaling, rotation and positioning. We have generated 10 distorted symbols per model and next compared them together by using the method describes in [3]. It computes a distance between set of graphic primitives based on their projection in twos. We have done it with 100 different variance values and compute for each of them a mean distance for whole symbols (10×150). So our experiments have been done on 150 000 distorted symbols. The obtained curve highlights that our algorithm distorts the objects in a logarithmic way. Indeed, the noise level grows a lot for the weak variance values (below 0.05). Our algorithm must be then set consequently in order to limit the distortion impact on the objects.



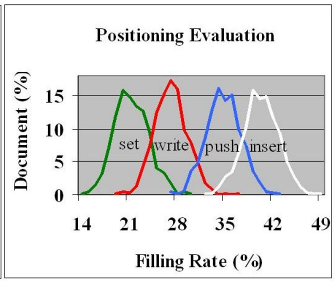


Figure 11. evaluation of algorithms

The second evaluation concerns the filling abilities of our positioning algorithms. We have generated from each of them a database composed of 1000 synthetic documents with our symbol models. The documents are 1024×1204 pixels with an average 41.74 symbols per document. Our experiments have been done on an overall number of symbol of 166 959. These symbols have been rotated (from 0 to $2 \times \pi$) with a gap of $\frac{1}{1000}$. We have computed next a filling rate per document (ie. sum of bounding box sizes of objects on the whole size of document) and then computed a distribution curve for each database. Our results show that our algorithms fill the documents in a gaussian way. All the curves look like a single gaussian one shifted with a regular gap. The stronger filling rates are obtained by algorithms preserving objects in document (*write* & *insert* vs *set* & *push*) as well as the ones using a moving strategy (*insert* & *push* vs *write* & *set*).

5. Conclusions and Perspectives

In this paper we have presented a system to build synthetic graphic documents for performance evaluation using two main components. The first one performs a vectorial distortion on the graphic objects. It is based on a gaussian model allowing to work without any previous learning step. The other one uses a set of positioning algorithms inspired by those used in array management. They use filling strategies and a stop criterion to preserve a partitioning between the objects and the document background. Experiments done on more of 300 000 symbols show that our algorithms distort and fill in a logarithmic and gaussian way.

These works open two main perspectives. A first concerns the improvement of our distortion algorithm. We plan to extend it in order to distort the objects in an hand-sketch way. The use of *Bezier* model seems to be a solution to do it. Our second perspective concerns the rule management to connect the objects together. We plan to investigate the use of adjacency grammars to deal with the topological constraints between objects as well as the domain rules.

References

[1] D. Lopresti and G. Nagy. Issues in ground-truthing graphic documents. In *Workshop on Graphics Recognition (GREC)*, volume 2390 of *Lecture Notes in Computer Science (LNCS)*, pages 46–66, 2002.

[2] D. Madej and A. Sokolowski. Towards automatic evaluation of drawing analysis performance: A statistical model of cadastral map. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 890–893, 1993.

[3] I. Phillips and A. Chhabra. Empirical performance evaluation of graphics recognition systems. *Pattern Analysis and Machine Intelligence (PAMI)*, 21(9):849–870, 1999.

[4] E. Valveny and P. Dosch. Performance evaluation of symbol recognition. In *Workshop on Document Analysis Systems (DAS)*, pages 354–365, 2004.

[5] L. Wenyin and D. Dori. Principles of constructing a performance evaluation protocol for graphics recognition algorithms. In *Performance Characterization and Evaluation of*

Computer Vision Algorithms, pages 97–106. Springer Verlag Publisher, 1999.

[6] J. Zhai, L. Wenyin, D. Dori, and Q. Li. A line drawings degradation model for performance characterization. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 1020–1024, 2003.

The authors wish to thank Sébastien Adam, Murielle Ramangaseheno and Karim Zouba (Rouen University, France) for their contributions to this work.