Textual summarisation of flowcharts in patent drawings for CLEF-IP 2012

Andrew Thean, Jean-Marc Deltorn, Patrice Lopez and Laurent Romary
INRIA - Humboldt Universität zu Berlin - Institut für Deutsche Sprache und Linguistik
andy.thean@gmail.com laurent.romary@inria.fr

Abstract

The CLEF-IP 2012 track included the Flowchart Recognition task, an image-based task where the goal was to process binary images of flowcharts taken from patent drawings to produce summaries containing information about their structure. The textual summaries include information about the flowchart title, the box-node shapes, the connecting edge types, text describing flowchart content and the structural relationships between nodes and edges. An algorithm designed for this task and characterised by the following method steps is presented:

- Text-graphic segmentation based on connected-component clustering;
- Line segment bridging with an adaptive, oriented filter;
- Box shape classification using a stretch-invariant transform to extract features based on shape-specific symmetry;
- Text object recognition using a noisy channel model to enhance the results of a commercial OCR package.

Performance evaluation results for the CLEF-IP 2012 Flowchart Recognition task are not yet available so the performance of the algorithm has been measured by comparing algorithm output with object-level ground-truth values. An average F-score was calculated by combining node classification and edge detection (ignoring edge directivity). Using this measure, a third of all drawings were recognized without error (average F-score=1.00) and 75% show an F-score of 0.78 or better. The most important failure modes of the algorithm have been identified as text-graphic segmentation, line-segment bridging and edge directivity classification.

The text object recognition module of the algorithm has been independently evaluated. Two different state-of-the-art OCR software packages were compared and a post-correction method was applied to their output. Post-correction yields an improvement of 9% in OCR accuracy and a 26% reduction in the word error rate.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: H.3.1 Content Analysis and Indexing; H.3.3 Information Search and Retrieval; H.3.4 Systems and Software; H.3.7 Digital Libraries

General Terms

Measurement, Performance, Experimentation

Keywords

Patent, Prior Art Search, Image Processing, Flowchart

1 INTRODUCTION

Certain inventions are most effectively summarised in pictorial form and patent professionals sometimes rely heavily on patent drawings for performing prior art searches. Searching patent drawings is currently a labour-intensive, error-prone task which would be facilitated by automatic indexing methods. Generic methods for automatically indexing patent drawings have been reported (Hanbury et al., 2011; Vrochidis et al., 2010; Codina et al., 2009) but the heterogeneity of patent drawings means that it is difficult for a one-size-fits-all approach to reach high levels of performance on all image classes. Flowcharts represent a large and useful subclass of patent images for which specially-adapted methods will improve indexing performance. They have a special role in the patents domain since they offer a way to graphically represent a *process* (a *process* is one of four legally-recognised categories of patent claim alongside *products*, *apparatus* and *use* e.g. Rule 43(2) EPC (2000)) and often contain information-rich text that is prohibited from other types patent drawing (they are usually considered an exception to rules that generally prohibit text from patent drawings e.g. Rule 46(2)(j) of EPC (2000) and Rule 6.2 (b) of PCT (1970)). Flowcharts were one of the image categories targeted in the Patent Image Classification task of CLEF-IP 2011 (see Piroi et al. (2011) and references therein), but the summarisation of flowcharts in patent drawings has not been discussed previously in the literature.

Few studies have addressed the automatic analysis of flowcharts in general. Ito (1982) used corner detection and edge-based block shape classification to derive FORTRAN code directly from a flowchart bitmap (see also US5386508). In Abe et al. (1986), a method for discriminating between the various components of a flowchart diagram (symbols, lines and characters) was discussed. Kasturi et al. (1990) proposed an automatic line drawing analysis system (including flowcharts) based on the reduction of the image into segments and closed loops. Yu et al. (1997) described a general framework for the analysis of engineering drawings characterised by alternating instances of symbols and connections lines. The particular case of flowchart summarisation was also discussed in Futrelle and Nikolakis (1995) in the context of automatic document analysis. More recently, Szwoch (2007) applied template matching to the recognition and description of hand-drawn flowchart components for the purpose of automatic diagram aestheticization. Vasudevan et al. (2008) described a prototype system of flowchart knowledge extraction from images based on chain code boundary descriptors and neural network classifier. Finally, in Lemaitre et al. (2010) the characteristic causal relationship between flowchart components (described as line strokes) was exploited through a set of syntactic constraints that helped address the problems of flowchart diagram recognition as well as text-graphics segmentation.

The paper is set out as follows: in Section 2 the problem addressed in the CLEF-IP 2012 Flowchart Recognition task is defined in detail; in Section 3 a method of solving this problem is described; in Section 4 the performance of the method is reported and discussed.

2 Problem Statement

2.1 Input image data

A training set of 50 training images were released in March 1012, followed by a set of 100 test images three months later: results were submitted on July 10th 2012. The input image data consists of binary images of flowcharts taken from patent drawings. Each image contains a single figure and has been cropped from a patent drawing to exclude irrelevant information such as patent metadata and page numbers. As such the task does not include figure segmentation: in general a single patent drawing may contain multiple figures and a single flowchart may be spread over multiple drawings/figures. In principle, the appearance of any patent drawing is constrained by the rules applied at large patent offices (although not all patent drawings are compliant). Examples of the most important of these rules as applied by the EPO are as follows (Rule 46(2) of EPC (2000), Section A-X, 7.5.1 of GL (2012)):

- only black lines drawn with drafting instruments are allowed;

- the scale of the drawing should allow for two thirds reduction in size;
- the character height should be larger than 0.32 mm;
- reference signs (*no-box* nodes) should be clear and without brackets;
- no text should be included in drawings unless it is indispensable (this rule is intended to minimise the need for language translation in drawings, however text in flowcharts is usually considered indispensable);
- leading lines (*wiggly* edges) should be as short as possible and extend to the feature indicated.

2.2 Text output

The aim of the Flowchart Recognition task was to produce textual summaries in a predetermined format containing information about the flowchart structure in the following 3 categories.

- Metadata describing the number of nodes and edges in the flowchart and, optionally, the 'title' of the flowchart (usually the figure number).
- Node data describing the type of each node and, optionally, the text appearing in the node. The following node types are possible: *oval, rectangle, double-rectangle, parallelogram, diamond, circle, cylinder, no-box, unknown* and *point*. All types of node correspond to boxes or image regions that may contain text apart from *point nodes* which are the junction points where two or more edges meet.
- Edge data describing node connectivity relationships including, optionally, any text attached to the edge. Edges are either undirected or directed (in the sense indicated by arrow symbols) and have the following possible types: *plain, dotted, or wiggly*. The type *wiggly* refers to elements referred to as *leading lines* in patent law: *wiggly* edges are not necessarily wiggly in shape, but serve to connect *reference signs* to nodes (e.g. see Section A-X, 7.5.1 of GL (2012)).

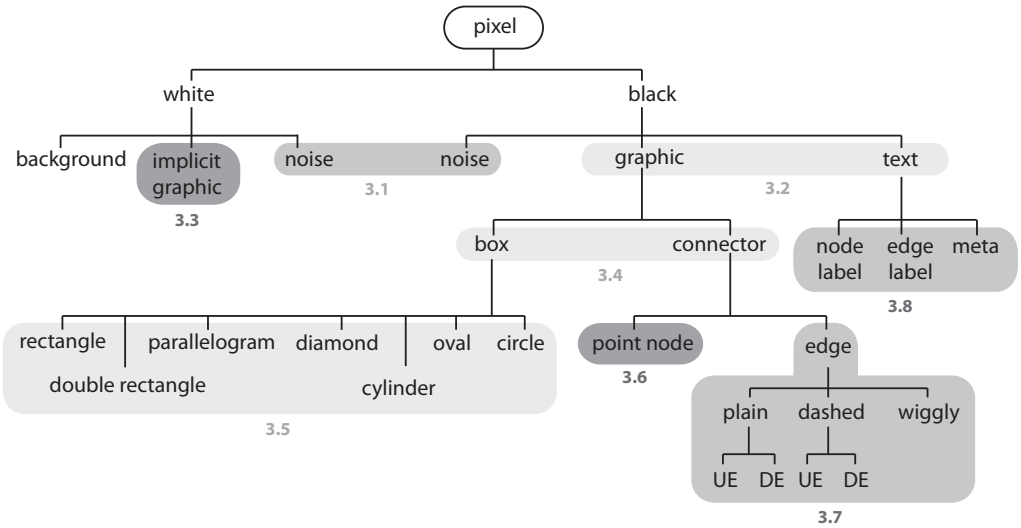

Figure 1: Possible membership states of image pixels in a flowchart. Numerical labels refer to the section headings below which describe how the various membership values are assigned. Note that joint membership states are, in principle, possible but are not allowed in the approach chosen.

3 Method

Figure 1 shows the possible membership states of image pixels in a flowchart. The method steps of the algorithm implemented are shown in Figure 2.

3.1 Preprocessing and de-noising

The input images may be scanned (and binarised) versions of hard copies filed with the patent application. Note that online filing of patent applications in electronic form now dominates submissions and leads to less noisy images, but scanned images dominate archived patent image databases. To remove some of the defects introduced by the digitisation process the following two-step method is applied. Firstly, a morphological opening with a kernel of 3×3 pixels removes isolated black dots. Secondly, a sequence of local median filters is applied in order to remove isolated line breaks and small 1-to-2-pixel gaps and to perform de-jagging of the lines. The choice of a median filter rather than a morphological closing, is motivated by the aim of bridging line breaks without artificially connecting text characters with the neighboring flowchart outline.

3.2 Text-graphic segmentation

For 70% of the training set it is possible to separate text from graphics (nodes and edges) by assuming that graphics are the single largest connected component. Based on this observation, text-graphic segmentation was performed by analysing the spatial extent of all connected components in the image. A mixture of three two-dimensional Gaussians was fitted to the height-vs-width distribution of the connected components using the Expectation Maximisation (EM) algorithm. The cluster with the largest height and width was deemed to contain all graphics components.

3.3 Finding implicit-graphics pixels

Certain types of connectivity between black pixels in a flowchart are merely implied i.e. the connectivity of dotted/dashed line elements, the connectivity of edges and nodes that do not physically touch and the outlines of broken box nodes (for example diamonds that are broken to accommodate text). The chosen method defines *implicit-graphics* pixels as white pixels that should be treated as black pixels in order to physically establish the connectivity implied by the flowchart author.

While some of the smaller line breaks are removed by the first preprocessing step (see Section 3.1), larger gaps can remain that cannot be resolved by simple morphological operations. Wedge-shaped masks, defined by an opening angle and a radius, are used to bridge such line breaks and connect dashed line segments. Firstly, a skeleton of the isolated graphic component is obtained and an estimate of the orientation in the neighborhood of each endpoint is derived. A wedge-shaped mask is placed at each endpoint and aligned with the local skeleton orientation. All intersecting pairs of masks are identified and two endpoints are reconnected by a straight line if their respective wedge masks intersect. This method is similar in nature to the dashed line detection algorithm presented in Kasturi et al. (1990).

3.4 Segmentation of closed regions

A connector is defined as a line in a flowchart that does not define a box node. Usually a connector pixel belongs to an edge, but it may also belong to a point node. The approach chosen for node-connector segmentation relies on a first step of box node classification. Closed regions in the graphics may correspond to either box nodes or *loops* (a *loop* is defined as a background region enclosed by edge lines). Node-connector segmentation is performed as follows. Firstly, loops are identified using the box classification technique described below. Secondly, the interior region of any remaining box node is dilated and used as a mask to separate nodes from connectors.

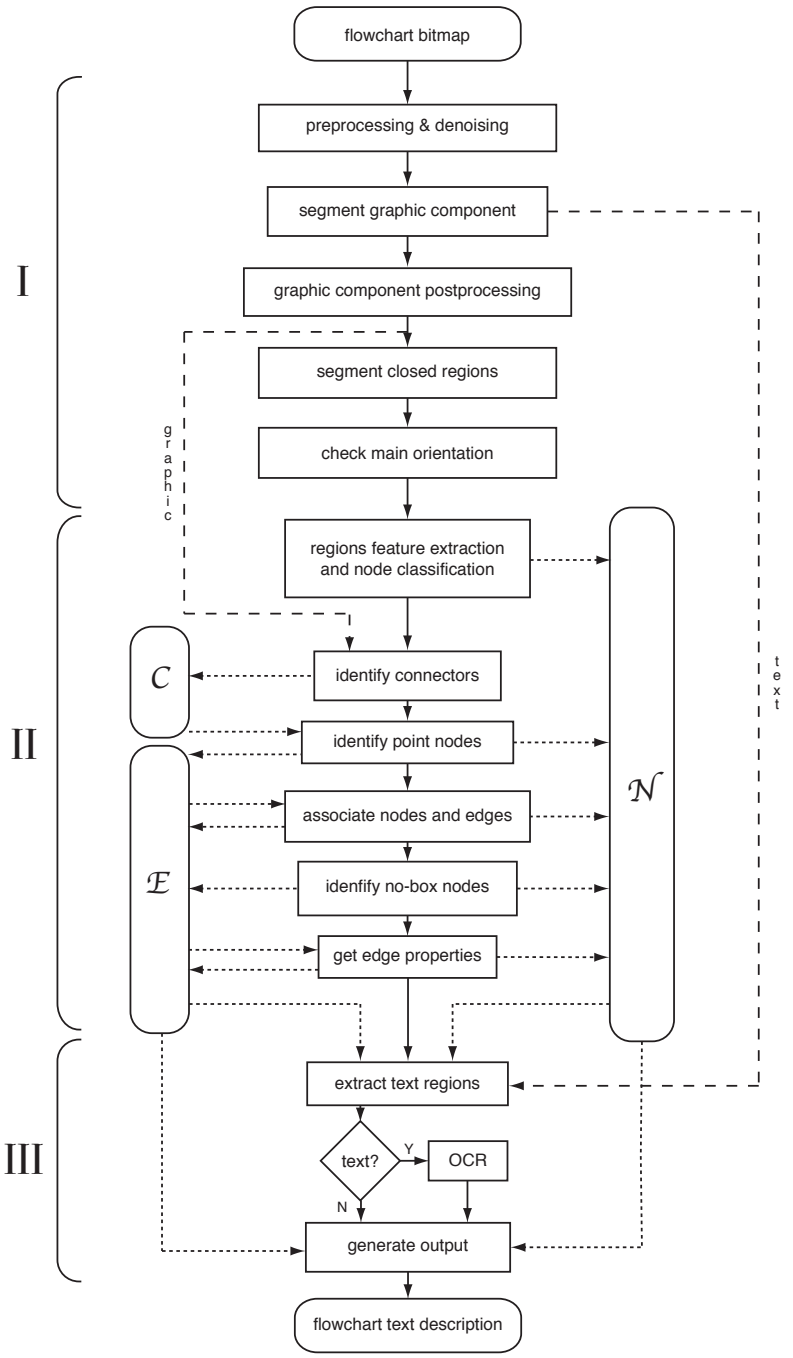

Figure 2: Method steps of the present flowchart summarisation process.The present processing architecture can be separated in three main components: (I) a preprocessing and segmentation stage (see sections 3.1 to 3.4), (II) a node and edge classification stage (sections 3.5 to 3.7), and (III) a text extraction and recognition stage (sections 3.8 and 3.9).In addition to the processing steps, the rounded vertical boxes represent the data structures aggregating the output of the connector identification (box "C"), the edge classification (box "E", which also contains the edge-node relationships) and the node classification (box "C"). Dashed lines represent a delayed transfer of data: Either the passage of the graphic component of the flowchart for the detection of the lines connections between nodes, or of the text layer used to extract individual text regions prior to OCR.

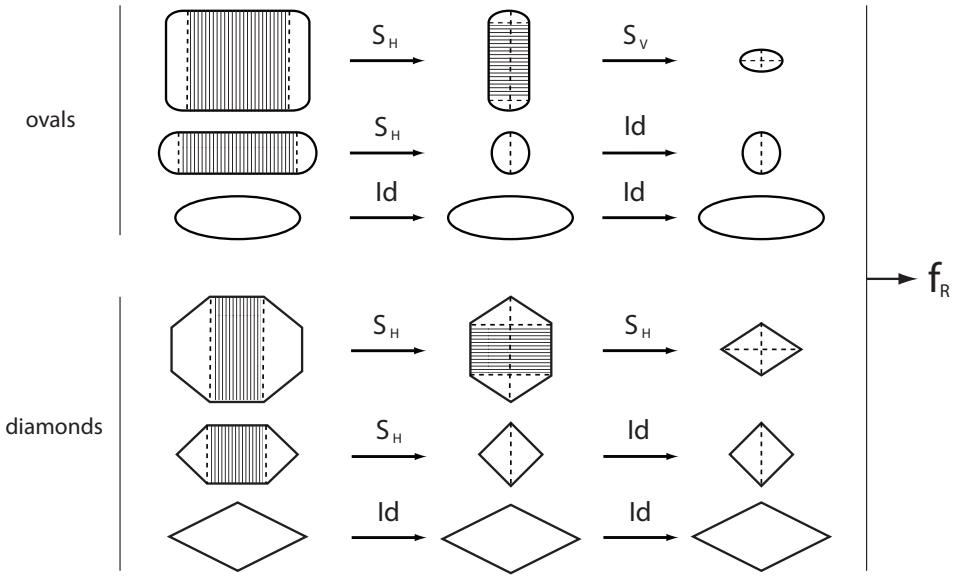

Figure 3: Squeezing operations resulting in shape primitives.A sequence of horizontal (S_H) and vertical (S_V) shape simplifications was applied to the original box regions prior to deriving the reduced set of image features, f_R. Examples of the transforms applied to various instances of the *oval* and *diamond* families are shown to illustrate how this step leads to comparable primitive shapes (i.e. an ellipsoid for the *oval* family, a *rhombus* for the diamond family).

3.5 Box classification

Estimating the global orientation of the flowchart was necessary to allow accurate box-node classification and OCR processing. The aspect ratio of the box node candidates identified after closed-region segmentation was measured and the entire image was rotated clockwise by 90 degrees if the fraction of box node candidates for which the height exceeded the width was greater than a fixed threshold.

Classifying box node candidates involved reducing each closed shape to a shape primitive. This step exploits the observation that during the flowchart authoring process individual box nodes may be squeezed or stretched in the horizontal and vertical directions according to the amount of text they contain. Since node classification should be invariant to such transformations, each candidate was subjected to a squeezing operation in the horizontal and vertical direction by application of the operators S_H and S_V, where

$$S_H = \frac{\partial}{\partial x}\left(\int_y B(x,y)\,\mathrm{d}y\right) \quad \text{and} \quad S_V = \frac{\partial}{\partial y}\left(\int_x B(x,y)\,\mathrm{d}x\right),$$

and discarding the segments for which S_V or S_V are below a predefined threshold.

The result of the squeezing operations is a shape primitive which characterises the basic shape family of each node type (see Figure 3). This shape simplification step is particularly suitable for the present definition of shape categories. In the present task, the class *oval* does not only encompass elliptical outlines (ovals in the simplest sense) but also half-circles joined by two horizontal lines and even rounded rectangles (see Figure 3). Similarly, the class *diamond* includes, in addition to the standard four-sided parallelograms, hexagons and octagons (each having two parallel sides of the same size and aligned with the horizontal or the vertical). Reducing each of these shapes to a single primitive (an elliptical shape in the case of *ovals*, a quadrilateral shape for *diamonds*) enables each family to be characterised.

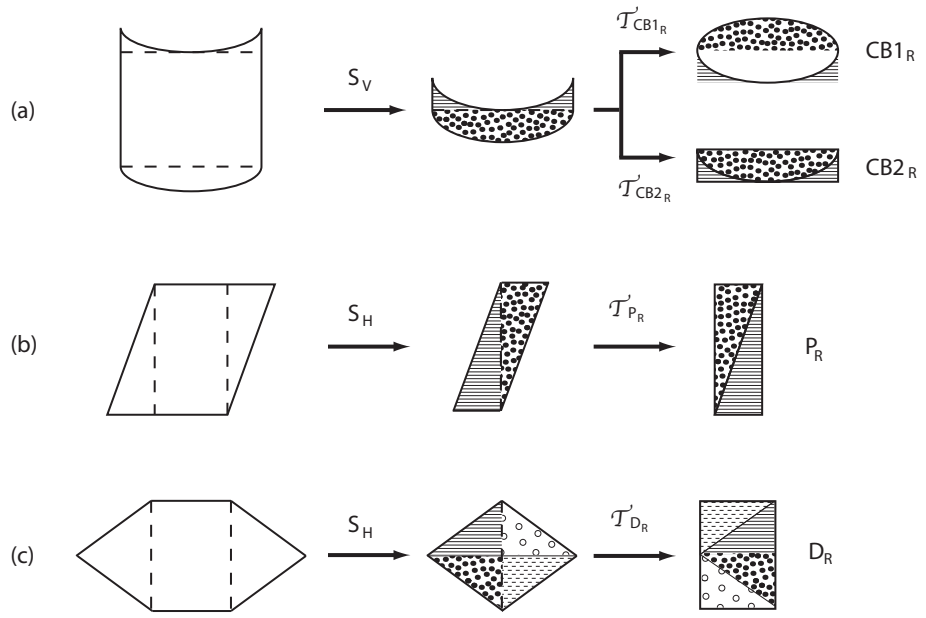

Figure 4: Derivation of the symmetric features for the *cylinder* bodies (a), the *parallelograms* (b) and the *diamonds* (c). After applying squeeze simplifications (S_H or S_V) to the original box regions, the resulting primitive shapes are decomposed in subcomponents and re-arranged using shape-specific transforms (T_{CB1_R} and T_{CB2_R} for the *cylinder* bodies, T_{P_R} for the parallelograms and T_{D_R} for the diamonds) prior to measuring the associated geometric features (ellipticity ($CB1_R$) and rectangularity ($CB2_R$, P_R and D_R).

Multiple shape features were extracted from the box nodes: a first set of features, f_0, was derived from the original box node, while a second set, f_R, was derived from the horizontally and/or vertically reduced shape. Features in the f_0 set include area (A_0), ellipticity (E_0), solidity (S_0: the ratio between the area within the bounding rectangle and the area within the box node perimeter) and the horizontal and vertical symmetries (SyH_0 and SyV_0). Reduced shape features include area (A_R), ellipticity (E_R), solidity (S_R) as well as symmetry measures aimed at characterising specific node shapes: *diamond*, *parallelogram* and *cylinder* bodies (see Figure 4).

D_R measures the rectangularity (here estimated by the solidity, as defined above) of a reduced shape after decomposing it into four quadrants and reassembling them as in figure 4(a). Diamonds are thus expected to be formed by complementary shapes and can be characterised by a value of D_R close to 1 (indicative of a high degree of rectangularity of the recomposed shape). In the same way P_R offers a measure of rectangularity of a recomposed shape obtained by cutting in two halves a horizontally-squeezed node box and repositioning its two portions (Figure 4(b)). Here again a high degree of rectangularity is expected from a horizontally-sheared rectangle (i.e. a *parallelogram* under the present task definitions). $CB1_R$ and $CB2_R$ measure the symmetries and complementaries expected from typical cylinder body shapes: $CB1_R$ measures the ellipticity of a composite shape obtained by adjoining the body half and the horizontally reflected top half of a vertically squeezed node box (Figure 4(c)), whereas $CB2_R$ measures the rectangularity of the two recombined halves. For an input node box region $B(x, y)$ the set of features given in Tables 1 and 2 is generated.

Based on the features defined in Tables 1 and 2, the box nodes are first classified into the following set of shape families, as defined in the CLEF-IP task: *circle, oval, parallelogram, rectangle, diamond*. On top of these required categories, we introduced the following additional classes: *loop* (to identify closed areas defined by edges and box outlines) and *small* (to flag regions having an

area below a fixed threshold). The two other node classes that are not defined by a closed outline shape (i.e. *point node* and *no box node*) are discussed in Sections 3.6 and 3.8.

A simple hierarchical classification scheme is used to discard the *small* regions and separate asymmetrical shapes (*loops*) from the other classes. The *rectangles* are then identified. The remaining shapes (*oval, circle, parallelogram, diamond, cylinder-body*) are then classified using heuristic thresholds on feature values. Statistical pattern recognition techniques have not been applied due to time constraints and the lack of image-level annotations in the training set. Obviously, the application of such techniques is expected to significantly improve classification performance.

Composite shapes (*double rectangles* and *cylinders*) are identified in a second pass as a combination of primitive shapes (*double rectangles* being formed by a collection of three adjoining rectangles, and *cylinders* by an oval and a cylinder body).

3.6 Point node detection

Point nodes, defined as the junctions where edges meet, were identified as follows. Firstly, horizontal and vertical lines in the connector-only image were detected using a simplified version of the Hough transform by projecting pixel values onto the vertical and horizontal axes and identifying maxima. Secondly, *point* node candidates were identified as the image coordinates where horizontal and vertical lines crossed. Thirdly, a box centered on the point node candidates was used for validation: candidates were accepted if the box contained a branch-point and 3 or more lines crossed the box perimeter.

3.7 Edge classification

The main challenge in classifying edge types surrounds the classification of *dashed* and *wiggly* edges. Dashed edges are not common and were identified as those containing multiple implicit-graphics segments (see Section 3.3). Counterintuitively, *wiggly* edges ('leading lines') are often straight and indistinguishable from plain undirected edges in the absence of context. These edges are classified solely according to their relationship to *no-box* nodes. Since wigglies correspond to *leading lines* and *reference signs* are of type *no-box*, any edge connecting a box node to a *no-box* was a assumed to be a *wiggly*. Preliminary trials suggested that truly curved *wiggly* edges could be detected using features describing the orientation of their endpoints which are neither horizontal nor vertical, such features could be useful for improving $no-box$ classification in future.

Edge directivity was classified using a simple set of features. Firstly, the two endpoints of each edge were identified and a fixed region around each endpoint was analysed. Next, the number of erosions required to remove all black pixels from the endpoint region was measured and a binary classification tree with heuristic thresholds was applied. Again, the use of statistical pattern recognition techniques was not applied due to time constraints and the lack of image-level annotations.

Feature	Operator
A_0	Area
E_0	Ellipticity
S_0	Solidity
SyH_0	Horizontal symmetry
SyV_0	Vertical symmetry

Table 1: Features and operators for the original box regions

Feature	Operator
A_R	Area o S_H o S_V
E_R	Ellipticity o S_H o S_V
S_R	Solidity o S_H o S_V
D_R	Solidity o T_{DR} o S_H o S_V
P_R	Solidity o T_{PR} o S_H
$CB1_R$	Ellipticity o T_{CB1_R} o S_V
$CB2_R$	Solidity o T_{CB2_R} o S_V

Table 2: Features and operators for the shape primitives

3.8 Text object segmentation and classification

The text-graphic segmentation method (Section 3.2) results in a set of connected components representing characters that must be further grouped into text objects and classified into box-node labels, edge labels, no-box nodes (*reference signs*) and metadata. Segmenting text objects involved the hierarchical, agglomerative clustering of characters according to the proximity of their centroids. Since characters are generally grouped horizontally, before clustering, vertical distances were weighted with a positive scaling constant which favoured horizontal associations.

Any text within the closed region of a box node was classified as a node label in a straightforward manner. To classify edge labels and no-box nodes each text object was dilated and checked for overlap with known edges. If the overlapping edge connects to two nodes (box nodes or *point nodes*) the text object is classified as an edge label, whereas if the overlapping edge connects to only one node the text object is classified as a *no-box* node and the edge is classified as a *wiggly*.

The fonts used for metadata (figure titles) tend to be larger than the other types of text objects and titles usually appear at the image edges. After classification of text objects into box-node labels, edge labels and no-box nodes any remaining text is re-clustered according to centroid positions. The new text objects were scored proportionally to their total area and their distance from the image center: the object with the highest score was classified as metadata.

3.9 Text object recognition

Reliable OCR of text in the patent flowchart collection is challenging for multiple reasons.

- The quality of image digitisation can be very low. This is mainly due to the low quality of the original scanning and the original paper support:
- The quality of patent flowchart draftsmanship is variable. Note that rules and guidelines discussed in Section 2.1 do not exclude hand-drawn flowcharts;
- The segmentation of textual content is difficult (see Section 3.8);
- The text used in flowcharts is a particular form of technical language, different from the general domain language addressed by the state of the art OCR engines.

These different issues have been addressed using flowchart-specific text segmentation followed by state-of-the-art OCR software (ABBYY FineReader, version 10 and Tesseract 3.0) and customised post-correction techniques based on a Shannon's noisy-channel model approach and language models.

It has been shown that post-correction of the recognition errors based on specialised models can provide a significant reduction of the word error rate produced by general OCR engines. The proposed technique is based on a Shannon's noisy-channel model approach, more precisely a probabilist modeling of the degradation of the original string by the OCR process. This approach is used by spell-checker systems and auto-correction/auto-completion systems for information retrieval systems. The model can then be applied to a noisy string for correcting recognition errors, as shown in Kolak and Resnik (2002).

3.9.1 Character Level Model

Following a probabilistic framework, finding the correct string c given an original string o corresponds to finding,

$$argmax_c P(c|o)$$

and, following Bayes' Rule,

$$argmax_c P(o|c)P(c)$$

Following Shannon's noisy-channel model approach, $P(c)$ is the source model estimating which string is likely to be sent, and $P(o|c)$ is the channel model estimating how the signal is degraded. In our case, $P(o|c)$ estimates the probability that the string o is recognised by the OCR when the original string present in the bitmap image is c. For the source and channel model, we introduce first an N-Gram model of characters denoted CM with N=6:

$$P(c) \simeq P(c|CM) = \prod_{m} P(c_m|c_{m-1}, ..., c_{m-5})$$

For the channel/error model, we use a probabilistic framework for edit operations following a confusion model.

$$P(o_{1..n}|c_{1..m}) = \prod_{1..m} P_e(c_{1..m} \rightarrow o_{1..n})$$

where P_e corresponds to the probability that the OCR recognised the string $o_{1..n}$ when interpreting the string $c_{1..m}$ based on the four following edit operations:

$$\begin{aligned}
&P_{match}(c_i \rightarrow o_j | c_i \in c_{1..m} \vee o_j \in o_{1..n}) = P_{match}(c_i) \\
&P_{substitution}(c_i \rightarrow o_j | c_i \in c_{1..m}, o_j \in o_{1..n}) = P_{substitution}(c_i \rightarrow o_j) \\
&P_{insertion}(c_i \rightarrow o_j | c_i \in c_{1..m}, o_j \in o_{1..n}) = P_{insertion}(c_i \rightarrow \varepsilon) \\
&P_{deletion}(c_i \rightarrow o_j | c_i \in c_{1..m}, o_j \in o_{1..n}) = P_{deletion}(\varepsilon \rightarrow o_j)
\end{aligned}$$

Having a probabilistic interpretation of the edit distance allows the probabilistic costs of the edit operations to be learned from a corpus of errors instead of setting costs manually/experimentally as is often the case with existing spell checkers based on noisy channel models. The probabilities $P_{substitution}(c_i \rightarrow o_j)$, $P_{insertion}(\varepsilon \rightarrow o_j)$ and $P_{deletion}(c_i \rightarrow \varepsilon)$ are learned from a corpus of OCR errors aligned with the expected correction following the Levenshtein distance:

$$\begin{aligned}
P_{match}(c_i) &= 1 - \frac{count(c_i \rightarrow \{c_j\}_{j \neq i}) + count(c_i \rightarrow \varepsilon)}{count(c_i)} \\
P_{substitution}(c_i \rightarrow o_j) &= \frac{count(c_i \rightarrow o_j)}{count(c_i)} \\
P_{insertion}(\varepsilon \rightarrow o_j) &= \frac{count(\varepsilon \rightarrow o_j)}{count(c)} \\
P_{deletion}(c_i \rightarrow \varepsilon) &= \frac{count(c_i \rightarrow \varepsilon)}{count(c_i)}
\end{aligned}$$

where c corresponds to any characters of the alphabet of the source.

A smoothing operation is then applied for estimating the edit operations unseen in the OCR-error training data. This was done by giving a very small additive probability to all character substitutions, insertions and deletions (Lidstone's additive smoothing).

Note that transposition is not used in our channel model because this sort of character error is relevant for spell-checker applications, but extremely rare in the case of OCR. The original Levenshtein distance was therefore preferred over the Damerau-Levenshtein distance generally used for spell-checking.

3.9.2 Language Model

N-gram language models are used to capture constraints on word sequences. For estimating P(c) a word trigram language model LM was combined with the character model CM:

$$P(c) \simeq P(c|CM)P(c|LM) = P(\langle c_p \rangle | CM) \prod_{p} P(c_p|c_{p-1}, c_{p-2})$$

where $\langle c_p \rangle$ denotes the complete concatenated character sequence with boundary characters corresponding to the p words c_p.

At this stage the main issue is the word segmentation of the corrected word sequence. The character model handles word splits naturally and merges by inserting, substituting or deleting character boundaries. Consequently, the application of the character model may result in different hypotheses having different numbers of words. The application of a language model on these different hypotheses will be biased by the fact that the probability of a longer word sequence is always penalised. For solving this problem, empty final words are introduced so that all hypotheses have the same word length (set to the maximum among the hypotheses). The transition cost to empty final words is computed dynamically as the average of the existing transitions.

Given a noisy word sequence recognised by the OCR, the final aim is to find, for a sequence of corrections of words:

$$\begin{aligned} argmax_c P(o|c)P(c) = argmax_c \prod_{0 \le p \le p_{max}} & \prod_m P_e(c_{1..m,p} \rightarrow o_{n..q}) \\ & \prod_m P(c_{m,p}|c_{m-1,p}, ..., c_{m-5,p}) \\ & P(c_p|c_{p-1}, c_{p-2}) \end{aligned}$$

where p_{max} is the maximum number of words in the segmentation introduced by the character model and n to q correspond to the character range of the corrupted character sequence corresponding to the word p in the correction.

The decoding of the intended sequence of words given the observed sequence of words recognised by the OCR is implemented using the Viterbi approximation.

3.9.3 Training

For the present experiments a set of approximatively 2000 corrections of error produced by ABBYY FineReader was used. A correction is simply the alignment of the erroneous strings and the corrected ones which allow the probabilistic costs of the edit operations introduced in the previous section to be learned. Since the corpus of errors was only produced for ABBYY FineReader, the resulting model has limited applicability to Tesseract.

For training the character and language model, a corpus of 10.000 patent descriptions and 2.000 scientific articles was used. In both cases, the resources were a mixture of English, German and French texts.

4 Results and discussion

4.1 Graph structure recognition

At the time of writing no official performance results are available for the algorithm test run submitted. Instead, the performance of the algorithm has been measured by manually comparing algorithm output with known ground-truth values. This method of evaluation, although time-consuming, allows a detailed analysis of the modes of failure of the algorithm. For each class of objects in the flowchart, Pr, the precision and, Re, the recall value of the classification scores over the whole test set were determined as

$$Pr = \frac{true\ positive}{true\ positive + false\ positive} \quad \text{and} \quad Re = \frac{true\ positive}{true\ positive + false\ negative}.$$

In addition, the balanced F-score, F, is calculated as

$$F = 2.\left(\frac{Pr.Re}{Pr + Re}\right).$$

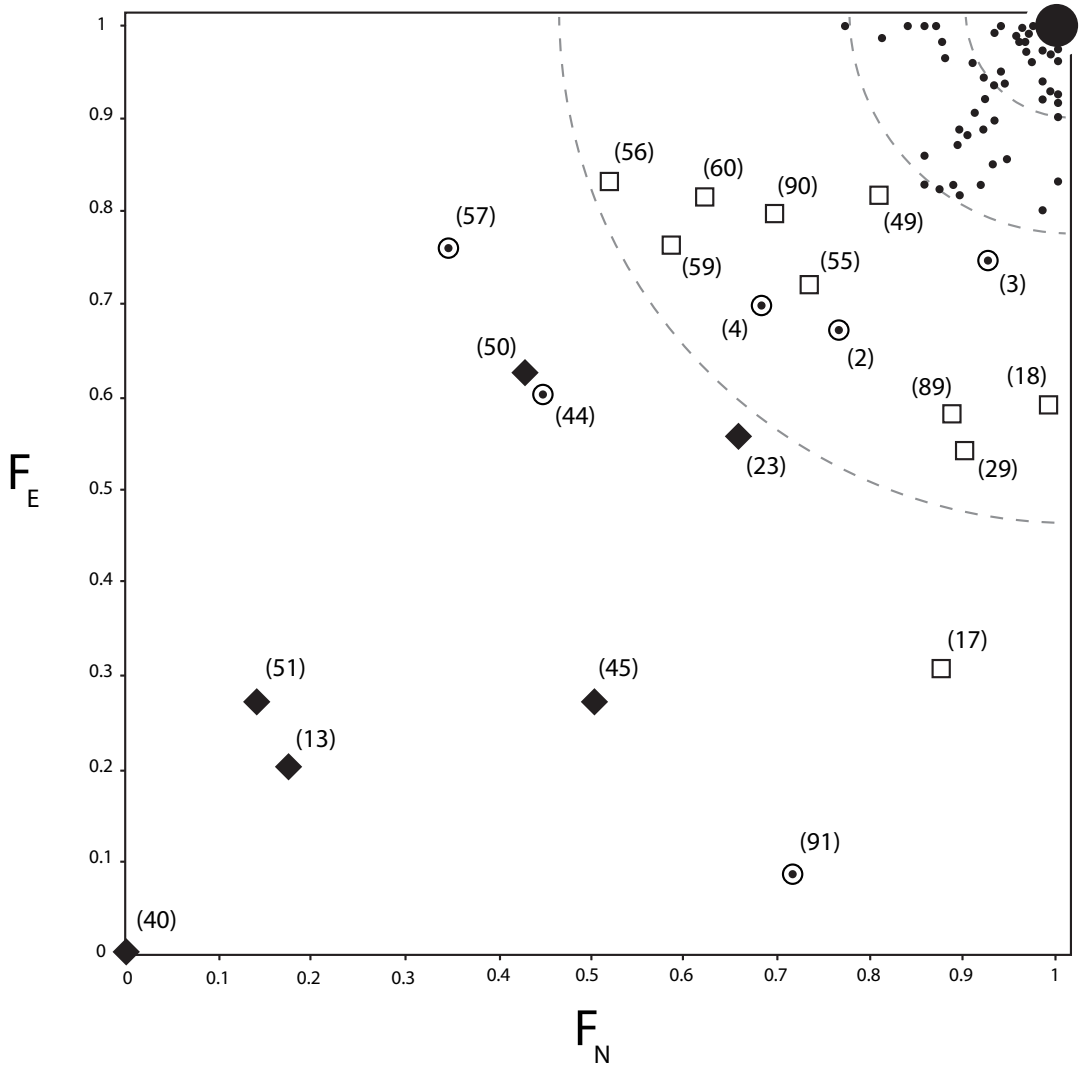

Figure 5: The distribution of F-scores for individual flowcharts F_E denotes the F-scores for all edges (omitting directionality), while F_N corresponds to the F-score of all nodes (except *no-boxes*) in each flowchart. Those flowcharts with low F-scores have been marked with numbers and symbols that indicate the mode of failure: Diamonds indicate poor text-graphic segmentation, dotted circles denote the problems caused by characters touching graphics, open squares relate to a substantial fraction of broken lines. All other flowcharts are represented by small disks. The larger black disk indicates a group of 34 flowcharts at F_N=F_E=1.00 . The grey dashed quarter circles mark the regions of the plot containing respectively 90%, 75% and 50% of the data points.

Class	Precision	Recall
All box nodes	0.91	0.83
rectangle	0.99	0.90
diamond	1.00	0.87
oval	0.97	0.81
circle	1.00	0.79
double rectangle	1.00	0.72
point	0.53	0.80
no-box	0.94	0.77
All edges (ignoring directivity)	0.90	0.89
DE	0.97	0.47
UE	0.31	0.81

Table 3: Precision and Recall scores for nodes and edges (node classes *cylinder*, *parallelogram* and *unknown* are not included since there are too few examples). The combined scores free all box nodes (excluding *point* and *no-box* nodes) and all edges (ignoring directivity) are also given.

Table 3 summarises the results of the evaluation for the various node and edge types. Note the following.

- The relatively high values of Pr and Re for box nodes is achieved despite the use of a simplistic classification approach. This indicates that the box features defined in Section 3.5 are discriminative.
- The relatively low values of Pr for directed edges and Re for undirected edges are probably caused by an overly simplistic approach to extracting directivity features and classifying them. A large fraction of directed edges were misclassified as undirected edges as shown by the relatively high values of Pr and Re when directivity is ignored.
- Since the identification of *wiggly* edges and *no-box* nodes occurs in conjunction, the performance scores for these two entities are almost identical: Pr_{wiggly}=0.96 and Re_{wiggly}=0.76.
- A recall score of 83% was measured for identifying the position of the title (prior to OCR).
- The relatively low value of Pr_{point} is partly explained by text touching characters which causes false positives.

In order to examine the patterns of algorithm failure, a single combined F-score, F_N, is calculated for all nodes of each flowchart (excluding *no-box* nodes since they are labels that do not define the flowchart logic). Similarly, a single combined F-score, F_E, is calculated for all edges of each flowchart (ignoring directivity). $\overline{F}$ denotes the average of the node or edge F-scores for a given flowchart.

The results of the combined F-score evaluation over the whole test set are summarised in Figure 5. Out of a total sample of 100 analysed flowcharts, 34 show a F-score of F_N=F_E=1.00. 50% of the diagrams have an average F-score superior to 0.90, while 75% show an average score of 0.78 or better. Only 10% of the test sample have an average F-score below 0.50. For the purpose of identifying the most significant sources of errors, the flowcharts associated with $\overline{F}$ <0.8 were we identified. Each of these flowcharts was assigned to one of three categories according to the main mode of failure: (i) initial text-graphics segmentation error, (ii) presence of text characters touching the flowchart graphics and (iii) discontinuities in the flowchart graphic structure (mainly due to broken lines).

Looking in turn at each of these classes, it was observed that poor segmentation of the main flowchart component at the early stage of processing was the main cause of very low F-scores. Six flowcharts have $\overline{F}$ <0.5: EP00000821886 (13), EP00000926609 (23), EP000010469970029 (40), EP000010702880135 (45), EP0000110761 (50) and EP00001113655 (51). The mode of failure for

this group of patents is associated with the text-graphic segmentation module (Section 3.2), in particular the assumption that the graphic component of the flowchart can be isolated from text by selecting the connected components with the largest heights and widths. A failure at this stage prohibits further node classification and edge identification.

Characters touching graphics (mostly text intersecting node boundaries) is also an important source of errors. Six flowcharts fall into category: EP000006215470107 (2), EP000006479080138 (3), EP000006879010063 (4), EP00001063844 (44), EP00001217549 (57) and EP00001526500 (91). Although this type of error is mostly due to a combination of design choices and scanning noise, it can also stem from the preprocessing steps designed to remove small gaps in the image components (Sections 3.1 and 3.3). This reflects the difficulty in bridging line breaks without creating unwanted pixel junctions between otherwise disconnected entities. It is noted that the problem of separating touching text from graphics remains a challenging issue for general OCR processing engines (Nagy (2000)). It has been studied in the document image analysis literature in relation to the analysis of maps (Fletcher and Kasturi, 1988; Luo et al., 1995)), form processing (Yoo et al. (1997)) and technical drawing understanding (Kasturi et al., 1990). An improved text-graphics segmentation method would benefit from the inclusion of an additional stage of detection and separation of touching characters.

The failure to correctly bridge broken lines (Section 3.3) is the leading source of error for values of $\overline{F}$ >0.5. A total of 10 flowcharts in the range 0.3< $\overline{F}$ <0.75 show significant errors that can be attributed to discontinuous graphical components. Line breaks can affect the connecting lines between nodes creating false positives in the edge components, as is the case for: EP00000866609 (17), EP000000884919 (18), EP00000098456 (29) and EP000001513121 (89)). Line breaks can also prevent a proper identification of the nodes as for: EP000001104172 (49), EP000001197682 (55), EP000001201195 (56), EP000001223519 (59), EP0000012274360 (60) and EP000001521176 (90). Improving gap bridging schemes would improve performance.

4.2 Recognition of text objects

In order to evaluate the performance of the text object recognition module (Section 3.9), text labels were transcribed manually for approximately 20% of the test set (412 out of a total of 2023 text boxes). The sentence and word scores given below are based on this sample.

The following scores are presented in Table 4.

- A *coverage* score, which is the percentage of cases where the OCR returns a result.
- An *accuracy* score which is the percentage of correct results produced by the OCR given the total expected results. When the OCR does not produce a result, it is considered that the result is incorrect. An accuracy evaluation is given at both *word* level and *sentence* level, i.e. if the complete text associated with a graphical object is correctly recognised.
- A *precision* score which is the percentage of correct results produced by the OCR given the total result produced by the OCR, similarly at word and sentence level.

Technique	ABBYY	ABBYY & post-correct.	Tesseract	Tesseract & post-correct.
Coverage (sentence)	99.76	99.76	61.84	61.84
Coverage (word)	96.81	96.70	77.84	78.25
Accuracy (sentence)	72.27	79.06	22.98	23.88
Accuracy (word)	76.13	82.43	44.43	46.36
Precision (sentence)	72.53	79.52	37.16	39.22
Precision (word)	78.63	84.13	54.03	57.08

Table 4: Result of OCR of text object with ABBYY FineReader and Tesseract with and without post-correction techniques.

As a comparison, the highest accuracy of current state-of-the-art OCR engines is considered to be around 98-99% correctly recognised words in optimal conditions. It is clear from the coverage evaluation that Tesseract filters out a large amount of results that it judges incorrect. Although the Tesseract confidence threshold affects its coverage, no exploration of this parameter was performed for lack of time. It can be observed that, although producing fewer results, the precision of Tesseract *out-of-the-box* remains significantly below that of ABBYY FineReader.

As the post-correction technique has been trained only on ABBYY FineReader errors, it is less effective at improving Tesseract results. The accuracy of ABBYY FineReader is improved by 9.4%, meaning a reduction of the word error rate of 26.2%. This reduction of the error rate is lower than the 60% reduction reported by Kolak et al. (2003) using a similar approach (but correcting an OCR engine with a significantly lower baseline performance). These results should be considered carefully because, due to time constraints, only a partial specialisation of the models was applied to the flowchart text problem:

- The texts considered for training the probabilistic costs of the edit operations, the character and language models were patent descriptions and scientific articles without text from flowcharts. Ideally, the text present on the flowchart of the training data (50 flowcharts) should have been combined with the other training resources.
- The errors used for training the costs of the edit operations were generated for ABBYY FineReader only. A fairer comparison would use Tesseract-generated errors for correcting Tesseract output.
- The ideal source for training the language model is the patent description of the patent where the flowchart figure appears. As the patent publication number was given with the flowchart images, the usage of the description text would have been possible, for instance by retrieving the ST.36 document via the 'Open Patent Service' of the EPO.

These issues suggest clear directions for future improvement. It should be possible to improve the post-correction model significantly by improving the way specialised training data is selected and exploited.

References

K. Abe, Y. Azumatani, M. Mukouda, and S. Suzuky. Discrimination of symbols, lines and characters in flowchart recognition. In *8th International Conference on Pattern Recognition*, pages pages 1071–1074, 1986.

J. Codina, E. Pianta, S. Vrochidis, and S. Papadopoulos. Integration of semantic, metadata and image search engines with a text search engine for patent retrieval. 2009. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.143.98`.

EPC. *European Patent Convention*. 14th edition, 2000. URL `http://www.epo.org/law-practice/legal-texts/epc.htm14l`. Note: the wording of Rule 46 EPC is almost identical to the wording used in PCT Rules 6.2 (b), 11.11 and 11.13 and US Manual of Patent Examining Procedure, Section 1.84.

L.A. Fletcher and R. Kasturi. A robust algorithm for text string separation from mixed text/graphics images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 10 (6):910 –918, nov 1988. ISSN 0162-8828. doi: 10.1109/34.9112.

R.P. Futrelle and N. Nikolakis. Efficient analysis of complex diagrams using constraint-based parsing. In *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, volume 2, pages 782 –790 vol.2, aug 1995. doi: 10.1109/ICDAR.1995.602019.

GL. *Guidelines for Examination in the European Patent Office*, 2012. URL `http://www.epo.org/law-practice/legal-texts/guidelines.html`.

A. Hanbury, N. Bhatti, M. Lupu, and R. Mörzinger. Patent image retrieval: a survey. In *Proceedings of the 4th workshop on Patent information retrieval*, PaIR '11, pages 3–8, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0955-4. doi: 10.1145/2064975.2064979. URL http://doi.acm.org/10.1145/2064975.2064979.

S. Ito. Automatic input of flow chart in document images. In *6th international conference on Software engineering*, pages 319–328, 1982.

R. Kasturi, S.T. Bow, W. El-Masri, J. Shah, J.R. Gattiker, and U.B. Mokate. A system for interpretation of line drawings. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(10):978 –992, oct 1990. ISSN 0162-8828. doi: 10.1109/34.58870.

O. Kolak and P. Resnik. OCR error correction using a noisy channel model. In *Proceedings of the Human Language Technology Conference (HLT)*, San Diego, California, USA, 2002.

O. Kolak, W. Byrne, and P. Resnik. A generative probabilistic OCR model for NLP applications. In *Proceedings of the Human Language Technology Conference (HLT-NAACL)*, Edmonton, Canada, 2003.

A. Lemaitre, H. Mouchere, J. Camillerapp, and B. Coasnon. Interest of syntactic knowledge for on-line flowchart recognition. In *9th IAPR International Workshop on Graphics RECognition (GREC 2011)*, pages 85–88, 2010.

H. Luo, G. Agam, and I. Dinstein. Directional mathematical morphology approach for line thinning and extraction of character strings from maps and line drawings. In *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, volume 1, pages 257 –260 vol.1, aug 1995. doi: 10.1109/ICDAR.1995.598989.

G. Nagy. Twenty years of document image analysis in pami. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(1):38 –62, jan 2000. ISSN 0162-8828. doi: 10.1109/34.824820.

PCT. *Patent Cooperation Treaty*. 1970. Done at Washington on June 19, 1970, amended on September 28, 1979, modified on February 3, 1984, and on October 3, 2001, (as in force from April 1, 2002).

F. Piroi, M. Lupu, A. Hanbury, and V. Zenz. *CLEF-IP 2011: Retrieval in the Intellectual Property Domain*, 2011. URL http://clef2011.org/resources/proceedings/Overview-CLEF-IP_Clef2011.pdf.

W. Szwoch. Recognition, understanding and aestheticization of freehand drawing flowcharts. In *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, volume 2, pages 1138 –1142, sept. 2007. doi: 10.1109/ICDAR.2007.4377093.

B.G. Vasudevan, S. Dhanapanichkul, and R. Balakrishnan. Flowchart knowledge extraction on image processing. In *Neural Networks, 2008. IJCNN 2008. (IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, pages 4075 –4082, june 2008. doi: 10.1109/IJCNN.2008.4634384.

Stefanos Vrochidis, Symeon Papadopoulos, Anastasia Moumtzidou, Panagiotis Sidiropoulos, Emanuelle Pianta, and Ioannis Kompatsiaris. Towards content-based patent image retrieval: A framework perspective. *World Patent Information*, 32(2): 94 – 106, 2010. ISSN 0172-2190. doi: 10.1016/j.wpi.2009.05.010. URL http://www.sciencedirect.com/science/article/pii/S0172219009000489.

J-Y. Yoo, M-Ki. Kim, S. Y. Ban, and Y-B. Kwon. Line removal and restoration of handwritten characters on the form documents. In *Document Analysis and Recognition, 1997., Proceedings of the Fourth International Conference on*, volume 1, pages 128 –131 vol.1, aug 1997. doi: 10.1109/ICDAR.1997.619827.

Yuhong Yu, A. Samal, and S.C. Seth. A system for recognizing a large class of engineering drawings. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(8):868 –890, aug 1997. ISSN 0162-8828. doi: 10.1109/34.608290.