Decomposing Document Images by Heuristic Search

Dashan Gao[1] and Yizhou Wang[2]

[1] Dept. of Electrical and Computer Engineering
University of California, San Diego
9500 Gilman Drive
La Jolla, CA 92093-0409 USA
dgao@ucsd.edu(**Not the correspondence author.**)
Tel. 1-(858)-534-4538

[2] Palo Alto Research Center (PARC)
3333 Coyote Hill Rd.
Palo Alto, CA 94304-1314 USA
Yizhou.Wang parc.com (**The correspondence author.**)
Tel. 1-(650)-812-4772
Fax: 1-(650)-812-4334

Abstract. Document decomposition is a basic but crucial step for many document related applications. This paper proposes a novel approach to decompose document images into zones. It first generates overlapping zone hypotheses based on generic visual features. Then, each candidate zone is evaluated quantitatively by a learned generative zone model. We infer the optimal set of non-overlapping zones that covers a given document image by a heuristic search algorithm. The experimental results demonstrate that the proposed method is very robust to document structure variation and noise.

Note: the 2nd author is the correspondence author.

Decomposing Document Images by Heuristic Search

Dashan Gao[1] and Yizhou Wang[2]

[1] Dept. of Electrical and Computer Engineering
University of California, San Diego
9500 Gilman Drive
La Jolla, CA 92093-0409 USA
dgao@ucsd.edu
Tel. 1-(858)-534-4538

[2] Palo Alto Research Center (PARC)
3333 Coyote Hill Rd.
Palo Alto, CA 94304-1314 USA
Yizhou.Wang parc.com
Tel. 1-(650)-812-4772
Fax: 1-(650)-812-4334

Abstract. Document decomposition is a basic but crucial step for many document related applications. This paper proposes a novel approach to decompose document images into zones. It first generates overlapping zone hypotheses based on generic visual features. Then, each candidate zone is evaluated quantitatively by a learned generative zone model. We infer the optimal set of non-overlapping zones that covers a given document image by a heuristic search algorithm. The experimental results demonstrate that the proposed method is very robust to document structure variation and noise.

1 Introduction

Document decomposition is a basic but crucial step for many document related tasks, such as document classification, recognition, and retrieval. For example, given a technical article, after it is decomposed into zones, the zones' properties can be used as indices for efficient document retrieval. Hence, an accurate, robust and efficient framework for document decomposition is a very important and demanding module of document analysis, which ensures success of subsequent tasks.

The goal of document image decomposition is to segment document images into zones. Each zone is a perceptually compact and consistent unit (at certain scale), e.g. a paragraph of text, a textural image patch. Methods for document image decomposition can be classified into three categories: bottom-up methods[5, 9, 15], top-down methods[1, 7, 8], and combination of the two[13]. Typical examples of bottom-up methods utilize detected connected components, and progressively aggregate them into higher level structures, e.g. words, text lines, and paragraphs (zones). Conversely, top-down methods decompose larger components into smaller ones. A typical top-down approach is the X-Y tree method [14], which splits a document image into rectangular areas (zones) recursively by alternating horizontal and vertical cuts along spaces. Usually,

both approaches heavily depend on detecting connected components, separating graphics and white space, or certain document generating rules and heuristics. Both parameters for detecting document components and rules/heuristics used to segment documents are often manually tuned and defined by observing data from a development set. Thus, the adaptability and robustness of these methods are limited. It is hard for them to be generalized from document to document. When there exist ambiguities (e.g. text in document is noisy or has accidental proximity), neither type of method decomposes pages reliably. Methods based on statistical pattern analysis techniques [6, 10] are generally more robust. However, current reported methods, so far, are still relatively naive. For example, features are ad hoc, and computation engines are greedy.

This paper proposes a novel approach to decomposing document images using machine learning and pattern recognition techniques. More specifically, given a document image, it first proposes over-complete overlapping zone hypotheses in a bottom-up way based on generic visual feature classifiers. Then, each candidate zone is evaluated and assigned a cost according to a learned generative probabilistic zone model. Finally, a zone inference module implemented as a heuristic search algorithm selects the optimal set of non-overlapping zones that covers the given document image corresponding to the global optimal page decomposition solution.

The most outstanding advantage of the proposed method is that it organically combines a convenient document representation, an elaborated computational data structure, and an efficient inference algorithm together to solve the page decomposition problem. In other words, it seamlessly incorporates data(documents), models(representation) and computing (data structure and algorithm) into an integrated framework. Thus, it makes the model effective for data representation and computation; and it also makes the computation efficient due to the convenient model and computational data structure. Moreover, this method is one of the very few methods providing globally optimal multi-column document decomposition solutions, besides the X-Y-tree-like context free grammar methods. Based on page decomposition results, further document analysis tasks, such as meta-data tagging, document recognition and retrieval, are expected to be more convenient.

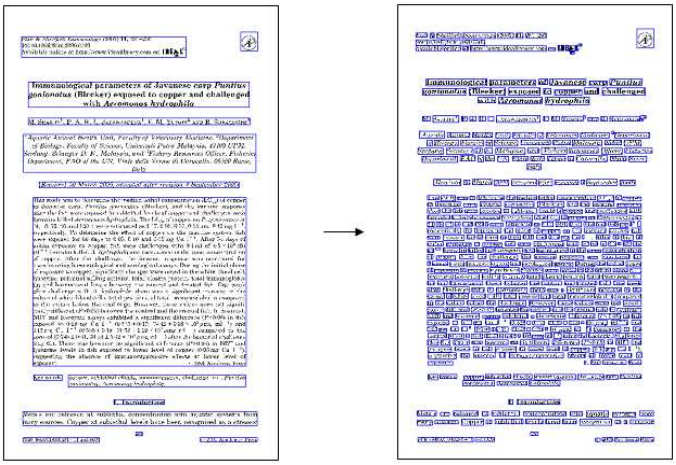
We first introduce a document image representation in Section 2. We discuss data preparation, document models and learning in Section 3 and 4. In Section 5, we implement zone inference by a well informed heuristic search algorithm. Some results are shown in Section 6. Finally, we summarize the proposed method in Section 7.

2 Document Image Representation

We represent a document image by a 2-layer hierarchical model. The first layer is called the *primitive layer*. Given a document image I, we apply standard techniques, such as [17], to detect "words" as atomic primitives, and connect these words into a word-graph, denoted as G_w.

$$G_w =< V, E >,$$

where $V = \{v_i; i = 1, \ldots, N_w\}$, each "word" is a graph node v. N_w is the number of "words" in a document. The edge set, $E = \{(e = (i, j), w_{ij}) : v_i, v_j \in V, w_{ij} \in$



(a) (b)

Fig. 1. Two layers of a document image model. a) Layout layer - segmented zones. b) Primitive layer - detected word bounding boxes.

$\mathbb{R}\}$, tells the neighborhood relation of pairs of "words." Each edge is associated with a weight, w_{ij}, representing bounding force between a pair of "words."

Note that these detected "words" need not be lexical words, a fraction of a word or an image patch is fine. The only purpose of this step is to reduce the image representation from pixels to a compact atomic "word" representation for the sake of computational efficiency.

The second layer is the *layout layer*, where the detected "words" are grouped into *zones* and form a zone-map, denoted as Z.

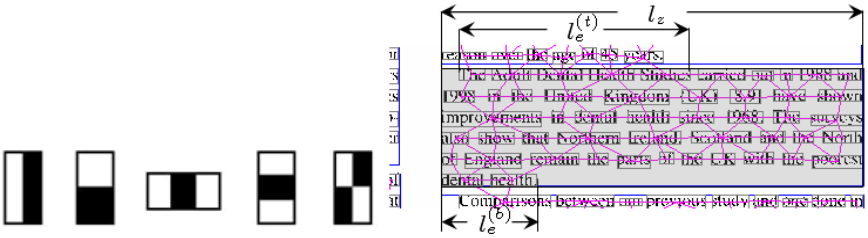
$$Z = (N_z, \{z_j : j = 1, \ldots, N_z\}), \tag{1}$$

where N_z is the number of zones. Each zone is defined as

$$z_j = (\{\mathbf{c}_i^{(j)} : i = 1, \ldots, n_{cj}\}, \{v_k^{(j)} : k = 1, \ldots, n_{wj}\}), \tag{2}$$

which is a polygon representation; $\mathbf{c}_i^{(j)}$ is a corner of a zone bounding polygon. n_{cj} is the number of vertices/corners of zone-j's bounding polygon. n_{wj} is the number of "words" comprising zone-j. Fig.1 shows the hierarchical representation of a document image.

Most conventional zoning algorithms heavily depend on connected component and white-space analysis, which involve *ad hoc* parameter tuning and rigid rule based reasoning. Consequently, the adaptability and robustness of the algorithms are limited. Our zone representation is based on corners, which is a well-known generic low-level robust visual feature, and it is independent of language. (Corners of the textural image bounding polygon is still obtained by connected component analysis.) Note that this



(a) (b)

Fig. 2. (a) Harr-wavelet features. (b) An illustration of zone bounding box edge cutting spans.

polygon representation for zones is not necessarily a rectangle. Our method is capable of handling diverse layout styles under a common generic zone model as shown in Fig.6.

From generative model point of view, we have the following causal dependence $Z \rightarrow G_w$. We integrate the two layers into a joint probability of G_w (derived from an input document image I) and the hidden representation Z:

$$p(G_w, Z) = p(G_w|Z)p(Z), \tag{3}$$

where $p(G_w|Z)$ is a generic zone likelihood model, and $p(Z)$ is a prior model for zone relations.

3 Data preparation

3.1 Features

Generic visual features In this project, we adopt 21 Harr-like filters to extract features from document images. These 21 filters are derived from 5 prototype Harr-filters (shown in Fig.2.(a), including a horizontal step edge, a vertical step edge, a horizontal bar(ridge), a vertical bar, and a diagonal blocks) by varying their size and scale. These features are generic and important visual features, and the filter responses can be computed in constant time at any scale and location using integral images[16].

"Word" related features "Word" related features are very important and convenient features for document analysis. In this project, we identified six types of such feature on the word-graph g. We introduce the definition of each feature as follows.

1. "word" compactness in a zone: $f_w^{(0)}(g) = \frac{\sum_{i=1}^{k} A(g_i)}{A(z)}$ where g_i is the ith connected component of the word-graph within a candidate zone. Usually $k = 1$ in a zone, i.e. words are highly connected to one another within a zone. $A(\cdot)$ is the area of a connected component bounding box. This word-graph connected component is not the pixel-based connect component adopted by conventional document image analysis methods. $0 < f_w^{(0)}(g) \leq 1$.

2. "word" height(size) consistency in a zone: $f_w^{(1)}(g) = n_w^d(g)/n_w(g)$, where $n_w(g)$ is the number of "words" in a zone, and $n_w^d(g)$ is the number of "words" with dominant height in the zone. Usually, $0 < f_w^{(1)}(g) \leq 1$. This feature tells the ratio of dominant sized "words" in a zone, which indicates the font size consistency of the zone.
3. zone bounding box top border edge-cutting span: $f_w^{(2)}(g) = l_e^{(t)}/l_z$, where l_z is the width of a zone, and $l_e^{(t)}$ is length of part of the zone bounding box top border that cuts word-graph edges. A graphical illustration of this feature is shown in Fig.2.(b). $0 \leq f_w^{(2)}(g) \leq 1$.
4. zone bounding box bottom border edge-cutting span: Similar to the above, $f_w^{(3)}(g) = l_e^{(b)}/l_z$, where $l_e^{(b)}$ is length of part of a zone bounding box bottom border that cuts word-graph edges as shown in Fig.2(b). $0 \leq f_w^{(3)}(g) \leq 1$.
5. zone bounding box vertical border average edge-cutting weight: $f_w^{(4)}(g) = \frac{\sum_{i=1}^{n_e^{(v)}} w_e^{(i)}}{n_{tl}}$, where $n_e^{(v)}$ is number of edges cut by the two vertical borders of a zone bounding box. $w_e^{(i)}$ is the ith edge weight. n_{tl} is the number of text lines in the zone. This feature indicates the connection force of a proposed zone with its surroundings. The larger edge weight cut, less likely it is a valid zone.
6. text line alignment in a zone: $f_w^{(5)}(g) = \min(\mathrm{var}(\mathbf{x}_l), \mathrm{var}(\mathbf{x}_c), \mathrm{var}(\mathbf{x}_r))$. It gives the minimum variance of the text lines' left, center and right x coordinates in a zone. The smaller the variance, the better the alignment.

These features are heuristic but independent of languages and layout style, as we try to avoid extracting syntax information from document to make our model more generalizable. These features are not necessarily independent, and they are utilized to evaluate the "goodness" of proposed zones.

3.2 Generating word-graph – the primitive layer

Given a document image I, we first compute a word-graph G_w using a neighbor finding algorithm based on the Voronoi tessellation algorithm[9] (Fig.3.(a)). Then, we compute edge weights, which tell how likely a pair of connected words are to belong to the same zone. The edge weights are posterior probabilities returned by an edge classifier discussed below. We adopt Support Vector Machines (SVM) to learn the binary edge classifier from word-graphs of training images as follows:

1. Data Preparation: We manually label zone bounding boxes on the training word-graphs. Positive edge samples are the edges within zone bounding boxes; negative samples are those cut by bounding box borders.
2. Feature Extraction: We extract a 22-dimensional feature vector including a feature accounting height difference of a pair of "words", and the 21 Harr-like filter responses (described in Section.3.1) from an image patch. The image patch is cut from I centering at the mid-point of an edge, and its area is four times large of the union of the two "words" bounding boxes.
3. SVM Training: We train a LibSVM[2] classifier on the extracted feature vectors.

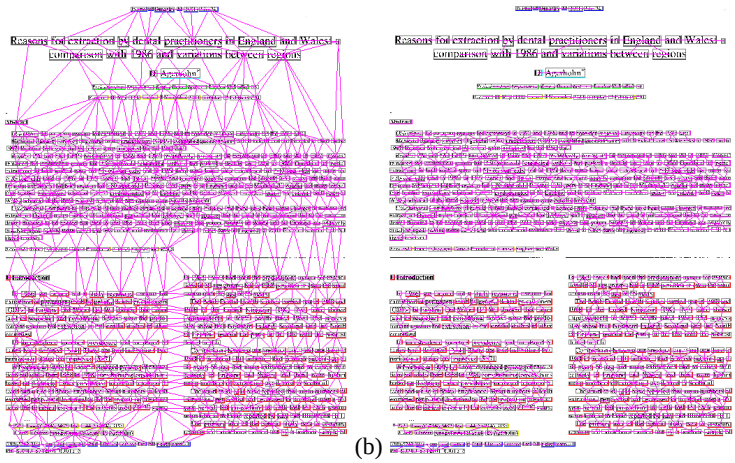


Fig. 3. (a) A Voronoi word-graph. (b) The same word-graph after prune edges whose weighs are less than 0.5 assigned by an edge classifier. Note that the edges between the paragraphs that are vertically adjacent to each other are not cut by the edge classifier.

Fig.3.(b) shows a word-graph after pruning edges whose weighs are less than 0.5 assigned by the SVM edge classifier. As the connection of a pair of "words" is computed based on generic features, the measure is more robust than pre-defined adhoc heuristic rules. Note that, in the figure, the edges between the paragraphs that are vertically adjacent to each other are not cut by the edge classifier.

3.3 Generating zone hypotheses

In Eqn.2, the zone representation is a generic polygon. In this paper, we demonstrate the power of the representation by simply using rectangles for zones without losing much generality due to the data set. Thus, Eqn.2 is reduced to $z_j = (\mathbf{c}_{ul}, \mathbf{c}_{lr}, \{v_k^{(j)} : k = 1, \ldots, n_{wj}\})$, where $\mathbf{c}_{ul}$ and $\mathbf{c}_{lr}$ are upper-left and lower-right corners of a zone bounding box.

In order to propose candidate zones efficiently, we train two classifiers, which detect upper-left and lower-right corners in document images, as follows:

1. Data Preparation: We obtain positive samples of zones' upper-left and lower-right corners using the labeled zones' corners in training word-graphs; Negative samples are collected by randomly selecting "word" bounding boxes' corners, which are not the corners of labeled zones.
2. Feature Extraction: We extract a 21-dimension generic visual feature vector (described in Section.3.1) from an image patch, which is cut from I centering at an upper-left or lower-right corner, and its size is 400×400 pixels.
3. SVM Training: We train LibSVM corner classifiers on the extracted feature vectors.

We augment the corner set by including bounding box corners of word-graph connected components in order not to miss any possible corners. Fig.4.(a) shows detected zone

Fig. 4. (a) Detected corners. (b) Proposed top candidate zones.

corners. We propose all possible candidate zones by pairing all the detected upper-left with all lower-right corners. If heuristics are used in this process, candidate zones can be proposed more effectively by ruling out some bad configurations such as a candidate zone cannot cross line separators, etc. Fig.4.(b) shows the top 51 candidate zones with least costs proposed by this method. For the sake of computation efficiency, the rest of zones with higher costs are discarded. The zone costs are assigned by a learned generative zone model introduced below.

4 Models and Learning

4.1 A likelihood model for zones

In Eqn.3, $p(G_w|Z)$ can be factorized into

$$p(G_w|Z) = p(g_{\bar{w}}) \prod_{i=1}^{N_z} p(g_i|z_i),$$

where $g_{\bar{w}}$ is sub-graphs of "words" not covered by any zone. $p(g_{\bar{w}}) = \exp(-|g_{\bar{w}}|)$, and $|\cdot|$ denotes the cardinality function. g_i is sub-word-graph(s) subsumed in zone-i, and $p(g_i|z_i)$ is a generative model for zones.

Intuitively, $p(g|z)$ governs how "words" are organized in zones in terms of the features $f_w^{(j)}(\cdot)$ described in Section.3.1. We want to construct a probabilistic model p on word-sub-graphs, such that the expected value of each feature is the same as its average

value extracted from training data. That is, given n labeled zones,

$$E_j[f_w^{(j)}(g|z)] = \sum_{i=1}^{n} p(g_i|z_i) f_w^{(j)}(g_i|z_i) = \frac{1}{n} \sum_{i=1}^{n} f_w^{(j)}(g_i|z_i) = \mu_j, \quad j = 0, \ldots, 5, \tag{4}$$

where j indexes the zone features of Section.3.1. The observed feature statistics serve as constraints. Thus, based on *maximum entropy* principle, the likelihood model for zones is derived as

$$p(g|z) = c \exp\{-\sum_{j=0}^{5} \lambda_j f_w^{(j)}(g|z)\}, \tag{5}$$

where λ's are Lagrange multipliers or, in this case, feature weights to be estimated. c is the normalizing constant. Note that as the features $f_w^{(2)}, f_w^{(3)}, f_w^{(4)}$ are "context sensitive," the zone model encodes a certain amount of contextual information.

Learning feature weights λ_j In Eqn.5, generally, there is no closed form *Maximum Likelihood Estimation* (MLE) solution for $(\lambda_0, \ldots, \lambda_5)$. We adopt a numerical method called *Generalized Iterative Scaling* (GIS) proposed by [4] to solve them iteratively as follows:

1. Given n labeled zones, compute each feature of each zone: $f_w^{(j)}(g_i|z_i), \ (j = 0, \ldots, 5, \ i = 1, \ldots, n)$.
2. Compute the average of each feature extracted from the training data,

$$\mu_j = \frac{1}{n} \sum_{i=1}^{n} f_w^{(j)}(g_i|z_i), \ \ j = 0, \ldots, 5.$$

3. Start iteration of GIS with $\lambda_j^{(0)} = 1, \ j = 0, \ldots, 5$.
4. At iteration t, with current parameter $\lambda_j^{(t)}$, use Eqn.5 to compute

$$E_j^{(t)}[f_w^{(j)}(g|z)] = \sum_{i=1}^{n} p^{(t)}(g_i|z_i) f_w^{(j)}(g_i|z_i), \ j = 0, \ldots, 5$$

   for each feature.
5. Update parameters

$$\lambda_j^{(t+1)} = \lambda_j^{(t)} + \frac{1}{C} \log \frac{\mu_j}{E_j^{(t)}}, \quad j = 0, \ldots, 5,$$

   where C is the correction constant chosen large enough to cover an additional dummy feature[4]. ($C = 8$ in this project.)
6. Continue iteration from Step.4 until convergence.

4.2 A prior model for zone-map

The prior model of zone-maps governs not only each zone's shape, but also spatial distribution of zones in a page, e.g. similarity, proximity, symmetry. It is characterized

by a statistical ensemble called *Gestalt ensemble* for various Gestalt patterns[18]. The model makes zone evaluation context sensitive. However, learning such a prior model is very expensive. In this project, we take advantage of the specificity of the document set by simply enforcing that each zone is a rectangle and there is no overlap between any two zones, such that $p(\{z_1, \ldots, z_{N_z}\}) = \prod_{i \neq j} \delta(z_i \cap z_j)$, where $\delta(\cdot)$ is the Dirac delta function. Thus,

$$p(Z) = p(N_z) \prod_{i \neq j} \delta(z_i \cap z_j), \tag{6}$$

where $p(N_z)$ is prior knowledge on zone cardinality, which we assume to be a uniform distribution.

In summary, the joint probability of a word-graph G_w and zone partition Z is

$$\begin{aligned} p(G_w, Z) &= p(G_w|Z)p(Z) \\ &= p(g_{\bar{w}})\{\prod_{i=1}^{N_z} p(g_i|z_i)\} \cdot p(N_z) \prod_{i \neq j} \delta(z_i \cap z_j) \end{aligned} \tag{7}$$

5 Zone Inference by Heuristic Search

Document image decomposition can be formulated into a Maximum A Posteriori (MAP) zone inference problem ($p(Z|G_w)$). However, to find the global optimal solution in this high dimensional space can be very expensive. In this paper, we propose a novel approach, which converts this challenging statistical inference problem into an optimal (covering) set selection problem by turning learned data statistics into costs and constraints. We design a well informed heuristic search algorithm, i.e. A^* search, to seek the global optimal page decomposition solution.

5.1 Generating costs and constraints from learned statistics

Instead of assigning costs and defining constraints in an *ad hoc* way, we derive them based on learned probabilistic models. In the page decomposition problem, we learn the following probabilistic models in Eqn.7: 1) a generative zone model, $p(g|z)$, and 2) a prior model about pairwise zone relation, $p(\{z_1, \ldots, z_{N_z}\})$.

We convert a probability $0 < P(\cdot) < 1$ to a cost as

$$c(\cdot) = \rho(-\log P(\cdot)), \tag{8}$$

where $\rho(x)$ is a robust function cutting off extreme values. When $P(\cdot) = 0$ or $P(\cdot) = 1$, there generates a binary constraint for that event. As a result, in this project, we have the following costs and constraints generated from the learned models: 1) individual cost for each zone, 2) a binary constraint that selected zones cover all "words" in a page, and 3) a binary constraint of no overlap between any pair of zones.

5.2 The A^* algorithm

A^* algorithm is a best-first graph search algorithm, which finds a path from an initial node to a goal node. It maintains a set of partial solutions, i.e. paths through the graph starting at the start node, stored in a priority queue. The priority assigned to a path passing node x is determined by the function,

$$f(x) = g(x) + h(x), \tag{9}$$

where $g(x)$ is a *cost function*, which measures the cost it incurred from the initial node to the current node x, and $h(x)$ a *heuristic function* estimating the cost to the goal node from x. To ensure the search algorithm find the optimal solution, $h(x)$ must be admissible.

In this project, after candidate zones are proposed, page decomposition can be formulated as a weighted polygon partitioning problem in computational geometry: given a polygon (document page) and a set of candidate sub-polygons (zones), each with a weight(cost), the goal is to partition the polygon into a subset of *disjoint* sub-polygons in the candidate set so as to cover every "word" in a document image with minimum cost. This problem can be solved by an A^* search algorithm, which exploit heuristics from data to improve search performance.

As A^* search algorithm is a standard algorithm in the search literature, here we only introduce each term in the algorithm in the context of document decomposition.

State Variable $\mathbf{x}$: Suppose that there are n candidate zones, we introduced a binary state vector $\mathbf{x} = (x_1, \ldots, x_n)$, where $x_i = 1$ means zone-i is selected; 0, otherwise. Any specific choice of 0's or 1's for the components of $\mathbf{x}$ corresponds to selecting a particular subset of the candidate zones.

The Goal State is every "word" in a given document is covered by only one zone.

The Cost Function $g(x)$: The cost of each path to $\mathbf{x}$ is defined as,

$$g(\mathbf{x}) = \mathbf{c}_z^T \, \mathbf{x}, \tag{10}$$

where $\mathbf{c}_z = (c_{z1}, \ldots, c_{zn})^T$ is the vector of individual zone costs, which was computed by Eqn.5 & Eqn.8 immediately after candidate zones are proposed.

The Heuristic Function $h(x)$: To insure the A^* algorithm admissible (or optimal), $h(\mathbf{x})$ must never overestimate the actual cost of reaching the goal. To achieve this and given the fact that both the document and the zones are represented by rectangles, the h-value of a path, from $\mathbf{x}$ to the goal state, is estimated by finding the minimum number of non-overlapping rectangles to partition the rest of the document page that has not been covered by the selected zones, $n_z(\mathbf{x})$,

$$h(\mathbf{x}) = n_z(\mathbf{x}) * c_{min}, \tag{11}$$

where c_{min} is the minimum zone cost learned from the training data. The estimate of $n_z(\mathbf{x})$ involves partitioning the complementary polygon of a state $\mathbf{x}$, which is created by removing the selected zones (rectangles) from the document page (a rectangle), into minimum number of non-overlapping rectangles. Partitioning arbitrary polygons is NP-complete, but becomes tractable if it is restricted to only *orthogonal* polygons (whose edges are either horizontal or vertical), and can be estimated by the following theorem [11].

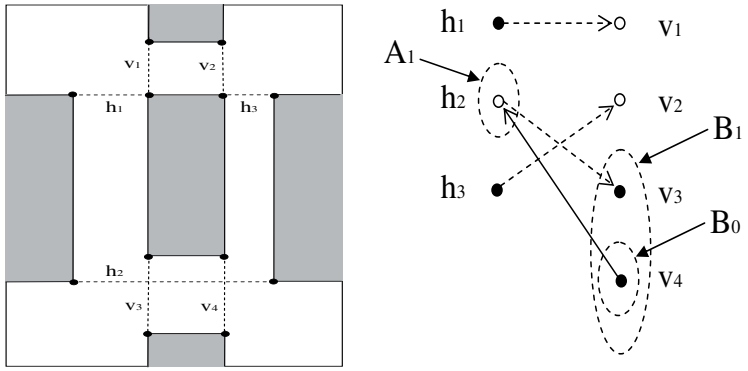


Fig. 5. An illustration of estimating a minimum rectangular partition of an document image. (a) A document image (the outer rectangle) with selected zones removed (shading rectangles). The black solid dots are reflex vertices. The vertical and horizontal chords between reflex vertices are labeled, each represented by a vertex of the bipartite graph in (b). (b) The bipartite graph constructed from the example in (a) and the derivation of a maximum independent set from a maximum matching (dashed edges are in the matching): vertices in the independent set shown as solid dots.

Theorem 1 *An orthogonal polygon can be minimally partitioned into $N - L - H + 1$ rectangles, where N is the number of reflex vertices*[3] *H is the number of holes and L is the maximum number of non-intersecting chords that can be drawn either horizontally or vertically between reflex vertices.*

One key computation in the theorem is that of finding L, the maximum number of non-intersecting chords that can be drawn either horizontally or vertically between reflex vertices. In this section we will show that this is equivalent to the problem of finding the *maximum number of independent vertices in the intersection (bipartite) graph of the vertical or horizontal chords between reflex vertices*, and derive a solution from a maximum matching of the bipartite graph [11, 12].

As illustrated in Figure 5(a), the complementary of a document image (the outer rectangle), to be partitioned, is generated first by removing the selected zones (the shading rectangles) in a path $\mathbf{x}$. To estimate L for this orthogonal polygon, a bipartite graph is first constructed in the following steps:

1. find all possible horizontal and vertical chords that can be drawn between all reflex vertices of the polygon (the black dots shown in Figure 5 (a));
2. include all horizontal chords obtained from Step.1 in a set H, and vertical chords in a set V;
3. construct a bipartite graph from the two sets by setting a vertex for each chord and drawing an edge between vertices of two sets if their corresponding chords intercept. Each edge is represented by a duplet, e.g. (h_1, v_1) for edge between vertices

[3] A reflex vertex is a vertex with interior angle greater than 180^o.

$h_1 \in H$ and $v_1 \in V$, and the set of all edges in the graph is E. The constructed bipartite graph is represented as $G = ((H, V), E)$.

Figure 5 (b) illustrates the bipartite graph constructed from Figure 5 (a), where $H = \{h_1, h_2, h_3\}$, $V = \{v_1, v_2, v_3, v_4\}$, and $E = \{(h_1, v_1), (h_2, v_3), (h_2, v_4), (h_3, v_2)\}$. The problem of finding L converts, now, to that of finding the maximum independent set of the bipartite graph G.

Definition 1 Maximum independent set problem *Given a graph G, an* **independent set** *is a subset of its vertices that are pairwise not adjacent (connected by an edge). The problem of finding the largest independent set in a graph G is called a* **maximum independent set problem**.

The independent set problem of an arbitrary graph is known to be NP-complete, but a polynomial solution exists when the graph is a bipartite graph. This can be done by solving a *maximum bipartite matching* problem [11, 12].

Definition 2 Maximum bipartite matching problem *Let $G = ((A, B), E)$ be an undirected bipartite graph, where A and B are sets of vertices, and E the set of edges of the form (a, b) with $a \in A$ and $b \in B$. A subset $M \subseteq E$ is a* **matching** *if no two edges in M are incident to the same vertex, that is no two edges share a common vertex. A vertex is called* **matched** *if it is incident to an edge in the match, and* **unmatched** *otherwise.* **Maximum bipartite matching** *is to find a matching M that contains the maximum number of edges of all possible matchings.*

The maximum bipartite matching problem is a well studied topic in graph theory, and often appears as a algorithm textbooks (e.g. [3]). As it is a standard algorithm, we omit the details for the sake of space limit.

In this project, we adopt graph-cut algorithm to solve the *maximum bipartite matching* for $G = ((H, V), E)$, so as to find the *maximum independent set* to solve L. Consequently, according to Theorem 1, the heuristic function $h(\mathbf{x})$ can be computed by Eq. 11. As the heuristic estimation is grounded on a solid theoretical foundation and generally gives a tight upper bound, the proposed algorithm becomes very efficient and a well-informed search strategy, which is also verified by the experiments.

Convenient Data Structures for Search To enforce zones' non-overlapping constraint, we create a matrix called *candidate zone overlapping matrix* O to encode candidate zones' overlap. It is very easily generated as follows: Say, there are n candidate zones. O is initialized as a $n \times n$ matrix, each entry is set to 0. Then, for each pair of overlapping zones, say zone-i and zone-j, we set $O(i, j) = 1$ and $O(j, i) = 1$. During the search, whenever a candidate zone, say zone-i, is selected as a part of the solution, we check the i-th row of O, and exclude every candidate zone-j where $O(i, j) = 1$ from future selection. In this way, it is guaranteed that no overlapping zones can possibly appear in the solution.

Another factor making the search efficient is due to a data structure called *word-to-candidate-zone index* $I_{w \to z}$. After n candidate zones are proposed, for each word in the document, there are pointers to the candidate zones that cover it. Note that each word can be covered by a number of overlapping candidate zones. But in the final solution,

only one of them is selected. At each search state, say $\mathbf{x}$, we check each word which has not been covered by selected candidate zones so far to see any available candidate zones covers it. The number of available candidate zones is decreasing when more and more candidate zones as selected because the more candidate zone that are selected, the fewer available candidates; and the more overlapping candidate zones are excluded from the candidate list. $I_{w \to z}$ needs to be updated dynamically at each search step. If there exists an uncovered word neither covered by any available not-selected-yet candidate zones, $h(\mathbf{x}) = \infty$. It means that this search path is terminated earlier, even though there are possible additional non-overlapping rectangles to be partitioned solely based on the current shape of polygon (current configuration of decomposed document). This step makes the search much more efficient by pruning spurious search paths at an earlier stage.

6 Data Set and Experimental Results

We train and test our model on the first page of articles from NLM's MEDLINE database. We randomly select a set of first pages for training and a different set for testing. Some inference results are shown in Fig.6. We can see that our results are very accurate and robust to document layout variations and noise, and it is also free from connect component restriction. Moreover, we claim that due to the convenient document representation and computational data structure, together with the well informed heuristic search strategy, the algorithm is capable of efficiently solving most page decompositions within a second. This efficiency is also because we limit the number of candidate zones by only considering the top 20 to 50 proposals from the generative zone model. Otherwise, the search space grows exponentially with the number of candidates.

7 Conclusions

We have proposed a novel, generic and efficient learning and inference framework to solve a fundamental challenging problem of document analysis. It organically integrates data, models and computing to search for globally optimized multi-column document decomposition solutions. The learning part learns robust probabilistic models based on generic features. The inference module casts an expensive statistical inference to a well informed heuristic search problem. As a result, the proposed framework is very general, and it can be extended to a lot of machine learning applications.

Acknowledgements

We'd like to thank Dr. Eric Saund for his numerous discussions and insightful suggestions to this project.

References

1. H.S. Baird, Background Structure In Document Images, *Document Image Analysis, World Scientific*, 1994.

2. C-C Chang and C-J Lin, LIBSVM: a library for support vector machines, 2001. Software available at http://www.csie.ntu.edu.tw/∽cjlin/libsvm
3. T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms*, Second Edition. MIT Press and McGraw-Hill, 2001. pp.664C669.
4. J. N. Darroch and D. Ratcli, Generalized Iterative Scaling for Log-Linear Models, *The Annals of Mathematical Statistics, Vol.43, 1972.*
5. *D. Drivas and A. Amin, Page Segmentation and Classification Utilizing Bottom-Up Approach,* ICDAR, *1995.*
6. *F. Esposito, D. Malerba, and F.A. Lisi, Machine Learning for Intelligent Processing of Printed Documents.* J. Intell. Inf. Syst. *14(2-3): 175-198, 2000.*
7. *H. Fujisawa and Y. Nakano, A Top-Down Approach for the Analysis of Documents,* ICPR, *1990.*
8. *R. Ingold and D. Armangil, A Top-Down Document Analysis Method for Logical Structure Recognition,* ICDAR, *1991.*
9. *K. Kise, A. Sato, and M. Iwata, Segmentation of page images using the area Voronoi diagram,* CVIU, *1998.*
10. *K. Laven, S. Leishman and S. Roweis, A Statistical Learning Approach to Document Image Analysis,* ICDAR, *2005*
11. *W. Lipski, E. Lodi, F. Luccio, C. Mugnai, and L. Pagli, On two dimensional data organization II.* Fundamental Informaticae, *2, 227-243, 1977.*
12. *W. Lipski, Jr. and F. Preparate, Efficient algorithms for finding maximum matchings in convex bipartite graphs and related problems,* Acta Informatica, *15, 329-346, 1981.*
13. *J. Liu, Y. Tang, Q. He, and C. Suen, Adaptive Document Segmentation and Geometric Relation Labeling: Algorithms and Experimental Results,* ICPR, *1996.*
14. *G. Nagy, S. Seth, and M. Viswanathan, A Prototype Document Image Analysis System for Technical Journals,* Computer, *25(7), 1992.*
15. *L. O'Gorman, The Document Spectrum for Page Layout Analysis,* PAMI, *15, 1993.*
16. *P. Viola and M. Jones, Rapid Object Detection using a Boosted Cascade of Simple Features,* CVPR, *2001.*
17. *Y. Wang, I.T. Phillips, and R. Haralick, Statistical-based Approach to Word Segmentation,* ICIP, *2000.*
18. *C. Guo, S.C. Zhu and Y. Wu, Modeling Visual Patterns by Integrating Descriptive and Generative Methods,* IJCV, *2003.*

Fig. 6. Page decomposition results. Note that the 1st result shows that the method is robust to noise and free from connected component restriction.