A Graph Lattice Approach to Maintaining and Learning Dense Collections of Subgraphs as Image Features

Eric Saund, *Member*, *IEEE*

Abstract—Effective object and scene classification and indexing depend on extraction of informative image features. This paper shows how large families of complex image features in the form of subgraphs can be built out of simpler ones through construction of a graph lattice—a hierarchy of related subgraphs linked in a lattice. Robustness is achieved by matching many overlapping and redundant subgraphs, which allows the use of inexpensive exact graph matching, instead of relying on expensive error-tolerant graph matching to a minimal set of ideal model graphs. Efficiency in exact matching is gained by exploitation of the graph lattice data structure. Additionally, the graph lattice enables methods for adaptively growing a feature space of subgraphs tailored to observed data. We develop the approach in the domain of rectilinear line art, specifically for the practical problem of document forms recognition. We are especially interested in methods that require only one or very few labeled training examples per category. We demonstrate two approaches to using the subgraph features for this purpose. Using a bag-of-words feature vector we achieve essentially single-instance learning on a benchmark forms database, following an unsupervised clustering stage. Further performance gains are achieved on a more difficult dataset using a feature voting method and feature selection procedure.

Index Terms—Graph lattice, subgraph matching, document classification, line-art analysis, CMD distance, weighted voting

1 Introduction

Effective object and scene classification and indexing depend on exploitation of informative image features. Recent trends in computer vision emphasize the use of large numbers of relatively simple features used for feature vector comparison, bag-of-words voting, and image database access using inverted indices [17], [26], [27]. An important issue is the information captured by a feature. While purely appearance-based features measure local shape and texture properties, additional information is found in spatial relationships among feature measurements sampled at keypoints or interest points.

One way of encoding spatial configuration is through graphs. Objects and scenes are modeled as parts (nodes) and relations (links) [4], [11], [28]. An observed image generates a graph of observed parts and their relations to other parts in the local neighborhood, and recognition is performed by subgraph matching. Subgraph matching poses two difficulties. First, it is known to be exponentially expensive. This problem is to some extent alleviated by use of attributed graphs, that is, graphs whose nodes contain properties that constrain possible matches. Nonetheless, subgraph matching has been limited to relatively small subgraphs due to the second difficulty, namely, noise and variability cause observed graphs to deviate from ideal models. This variability demands use of error-tolerant, or inexact, graph matching techniques, which drastically increases matching cost and largely removes the advantages of attributed graph matching because possible matches of differently labeled nodes must now be explored. Conte et al. provide a thorough review of these issues [6].

This paper shows that it is possible to exploit a rich vocabulary of image features that encode local geometrical structure through exact attributed graph matching, which can be made extremely efficient through the use of a graph lattice data structure. Features correspond to subgraphs, each of which encodes a limited amount of information about spatial configuration in a local neighborhood. By supporting large numbers of possibly overlapping and redundant feature subgraphs, image structure can be captured through exact graph matching even in the presence of noise and variability. The approach is analogous to shingling in text retrieval [12]; this paper extends the concept to two dimensions.

Fig. 1 illustrates the key concept of organizing subgraphs as a lattice while regarding them as image features which are measured by counting mappings (exact matches) to observed data graphs.

The idea is applicable to practical problems in classification and indexing of documents containing consistent subgraphs. Our focal problem is forms documents that have sparse and inconsistent textual content due to variable filling in of data fields, but which usually contain a network of rectilinear rule lines serving as region separators, data field locators, and field group indicators (Figs. 2 and 14). Rule lines intersect each other in well-defined ways that form junction and free-end terminator graph nodes. These are natural candidates for the nodes of an attributed graph

• *The author is with the Intelligent Systems Laboratory, Palo Alto Research Center, Palo Alto, CA 94304.*

Manuscript received 27 Feb. 2012; revised 4 Sept. 2012; accepted 1 Dec. 2012; published online 19 Dec. 2012.
Recommended for acceptance by M. Brown.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-2012-02-0148.
Digital Object Identifier no. 10.1109/TPAMI.2012.267.

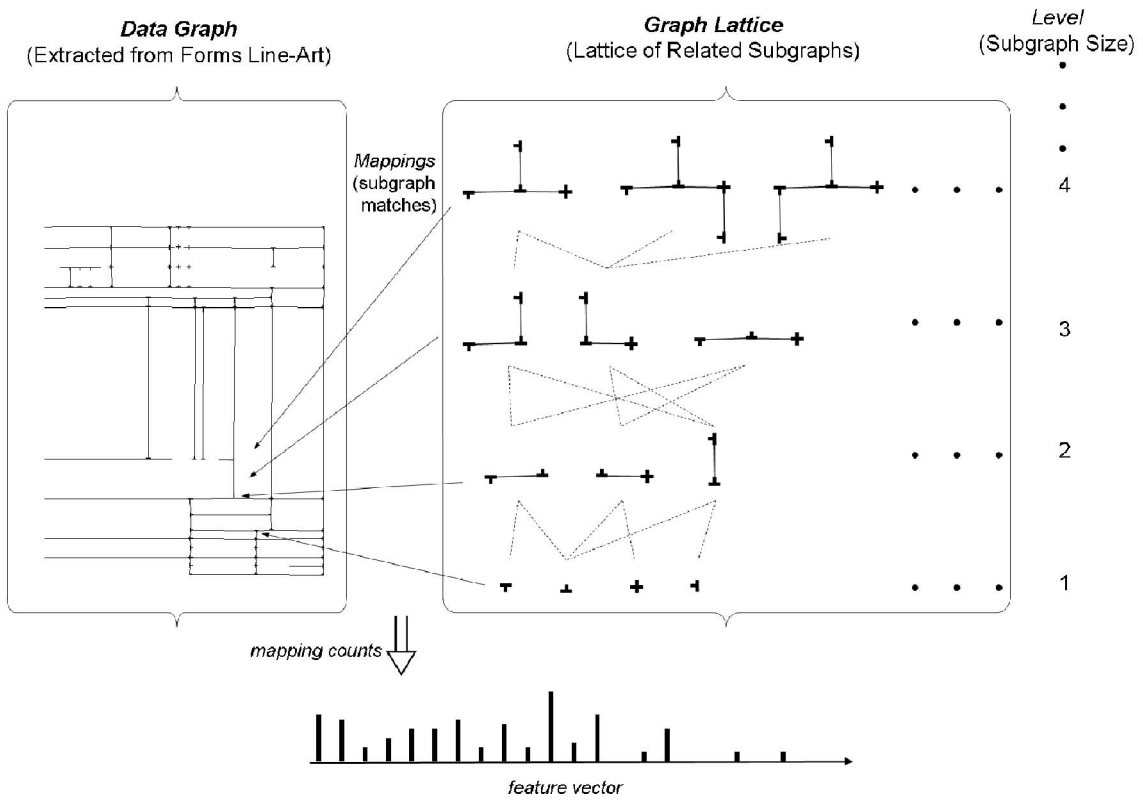


Fig. 1. Overview of the main idea. For the application of document forms categorization and clustering, image line art is processed to produce *data graphs* whose nodes are line-art junctions. A feature vocabulary is constructed, comprised of subgraphs. Mapping (exact matching) of these subgraphs to a data graph is made efficient by organizing the subgraphs as a *graph lattice*. Parent-child relations in the graph lattice build larger subgraphs from smaller ones. This feature vocabulary can be used in a variety of ways; one way is to construct feature vectors based on counts of mappings of subgraphs onto observed data graphs; a second way is through indexing and weighted voting (not depicted). Combinatorial explosion in the number of possible subgraphs of increasing size (level) is managed by selecting from actually observed subgraphs in a sample dataset.

representation (a *data graph*), with graph links arising from line segments linking junctions.

Document type classification[1] is a principal step in virtually all automatic document processing systems. Features used for forms classification include text content, connected component attributes, Haar filters, zonal densities, texture filters, and line-art locations and junctions. Methods for forms classification include string matching, template matching, nearest neighbor, decision trees, generative feature density models, and neural networks [16], [19], [20], [24]. Chen and Blostein [5] provide an extensive survey. By and large, most prior work emphasizes sophisticated learning algorithms applied to simple, easy-to-compute features.

One of the important issues in pattern classification is the number of training samples required to build a model. In commercial scenarios, customers frequently provide only one or a small number of examples of each doctype. Yet existing methods require at least 10 exemplars per category to build distributional models leading to robust performance given the simple features they are based on [20]. Our practical aim is to enable single-exemplar doctype learning, which may entail automatic unsupervised clustering of forms images into their correct types.

Rectilinear lines are also found in abundance in other image domains, for example, organization charts, maps, and engineering drawings. Every architectural firm has its own layout for its drawing title blocks that define various fields such as project name, drawing name, drawing number, and so on. The ability to train with one example and then recognize the distinctive layout identity and fields of a title block is of significant practical interest.

A graph matching approach to document genre classification was used by Bagdanov and Worring. Each document category is modeled as a configuration of layout elements (e.g., paragraphs and other zone-level items, which can be problematical to find) whose (variable) spatial relations are modeled with a single first-order random graph (FORG). FORGs model distributions of families of graphs and evaluation against an observed data graph is even more complex than inexact subgraph matching [2].

Two notable examples of encoding geometry in localized features have been demonstrated in the document image analysis domain. Nakai et al. [15] perform indexing of large document image databases using localized features that encode the spatial configurations of word blobs. Amit and Geman [1] perform character shape classification using forests of simple features that encode configurations of edge events; the vocabulary of such features is grown from simple to complex based on training data. The present work follows this vein. The idea of growing a vocabulary of larger subgraphs from smaller ones was raised for the document character and symbol recognition problem by Sidere et al. [25]. Other work in bringing bag-of-words text retrieval methods to image retrieval observes that a large vocabulary

1. Known in the trade as *doctype classification*.

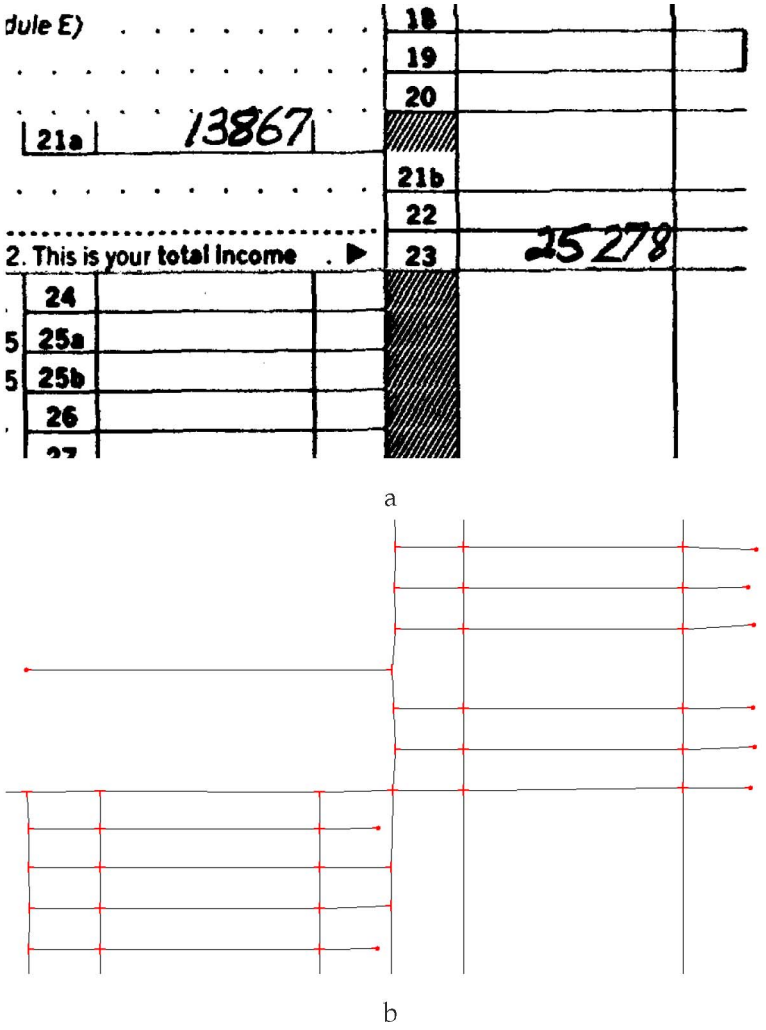


Fig. 2. (a) Section of a NIST tax form image. (b) The data graph (junctions and links) extracted from it. Note errors of two missing T-junctions.

of structural features does provide for robustness through the use of statistical methods for matching [9].

Messmer and Bunke [13], [14] introduced an approach to matching related model subgraphs to data graphs by precomputing a linked representation for shared subgraphs. This was applied to error-tolerant subgraph matching. The present work can be viewed as taking this approach to an extreme by representing a multitude of overlapping and related subgraphs linked in lattice organization. Shervashidze et al. [23] considered the set of subgraphs up to size k as kernels for graph comparison, but did not leverage a lattice organization nor node attributes associated with an application domain like document image analysis.

The paper is organized as follows: Section 2 describes the use of subgraphs as image features for bag-of-words clustering and classification using fixed-length feature vectors. We develop methods for reweighting feature counts and for comparing feature vectors. On a benchmark forms dataset, we show that a feature representation gains discriminative power when it incorporates larger features formed by grouping primitives into subgraphs. Unsupervised clustering is successful such that following the clustering stage, a single labeled instance of each category yields a correct forms classifier. Section 3 shows how processing may be made efficient by use of a graph lattice architecture. Efficiency is important to maintaining large vocabularies of graph-lattice features. Organized as a lattice with appropriate bookkeeping, each subgraph requires only a small incremental amount of work to compute its mappings onto a data graph. Section 4 introduces a more difficult dataset obtained under production document processing conditions. For this data, we demonstrate a voting approach to using subgraph features that exploits their location and dimensional attributes, and we introduce a method for advantageously extending the subgraph vocabulary to include larger subgraphs representing more precise features.

2 Data Graphs and Subgraph Features for Rectilinear Line-Art Analysis

2.1 Noise and Variability in Line-Art Data Graphs

This paper refers to two data corpora. A standard benchmark set for forms classification and data capture is the NIST tax forms database [8] consisting of 11,185 images of size $2,560 \times 3,300$ pixels. These are representative of 20 categories of scanned hand filled and typed United States Tax Forms. A second dataset consists of approximately 6,000 standardized dental forms scanned under

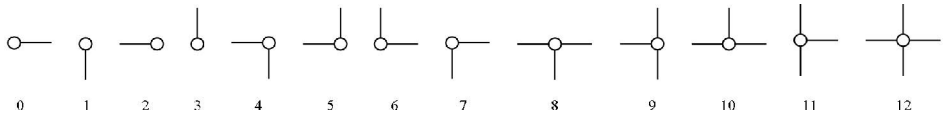


Fig. 3. The 13 junction types in rectilinear line art.

production document processing conditions. The properties of this dataset are discussed in Section 4.

The NIST dataset contains sufficiently distinct categories that existing supervised classifier methods are able to perform 100 percent correct classification [19], [20], [24]. However, these systems require tens or hundreds of labeled training samples. Instead, it would be highly preferable to achieve single-instance training, perhaps preceded by an unsupervised clustering stage.

Effective handling of image noise and variability is one of the key underlying issues in all domains of computer vision. Fig. 2a shows, in a NIST tax forms image, that line art is often noisy and distorted. Using standard as well as specialized image processing approaches [18], we can extract rule lines and their intersections at junctions with approximately 95 percent reliability. In a typical form of 250 terminations and junctions, we will therefore see about 10-15 errors. Typical errors are substitution of T-junctions and X-junctions (actually + junctions in upright orientation) and vice versa, introduction of spurious terminations where line art is broken by image noise, and failure to identify L-junctions.

By enumeration of possible configurations, it is clear that rectilinear line art gives rise to 13 junction/termination types, as shown in Fig. 3. For an observed line-art image, after extraction of rule lines it is straightforward to build a data graph consisting of attributed nodes and links among them, as shown in Fig. 2b. A data graph node's attribute is the index of its junction/termination type.

Junction-link relations define the topology and directional geometry of an observed line-art image. Dimensional geometry—the lengths of rule lines between junctions—can be important to encode as well. Dimensional properties are continuous valued in our target domain and therefore complicate mapping and matching in discrete graphs. Hence, in this study we encode dimensional attributes of feature subgraphs, as well as their locations in an image, separately from their node-link structure and junction attributes. Dimensional geometry and location attributes of subgraph features are exploited in Section 4.

2.2 Primitive Junctions Are Not Effective Image Features

As a baseline step toward considering subgraph features, we may ask whether the counts of primitive junction types (i.e., subgraphs of size 1) are sufficiently informative to distinguish NIST tax forms. Fig. 4 shows that they are not—at least not in a straightforward manner. This figure shows a histogram of pairwise similarities between 1,000 of the 11,185 NIST images based on comparing vectors of junction counts. Two primary modes are apparent. A high-similarity mode derives from forms of the same type, while a low-similarity mode derives from forms of different type. Note that the modes are not clearly separated, which means that there is a range of confusion

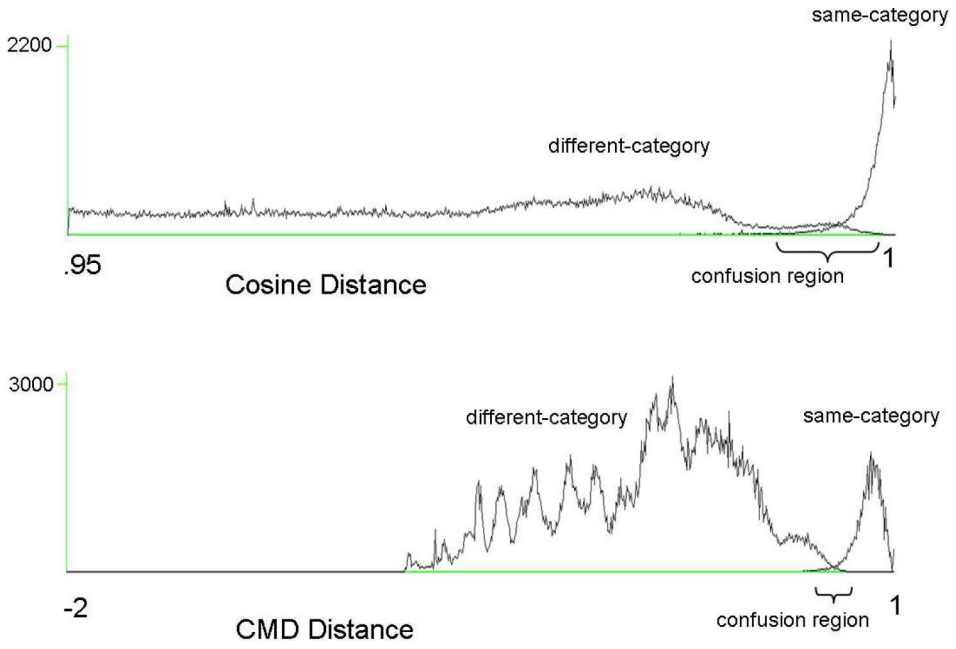


Fig. 4. Histograms of pairwise similarity scores based on primitive junction type counts for 1,000 US tax forms images, comprising 20 categories, drawn from the 11,185-image NIST corpus. In each graph, two histograms are plotted, one for within-category similarity scores, the other for between-category similarity scores. Two measures for vector similarity are shown. Top: cosine distance; Bottom: CMD similarity measure described in Section 2.5. In each graph, the number of histogram buckets is 600. The range of similarity values shown is $[0.95, 1]$ for cosine distance, $[-2, 1]$ for CMD similarity, where 1 (rightmost histogram bucket) means identical feature vector. The confusion region indicates that no threshold similarity score can be set to correctly classify the images based on primitive junction type counts alone.

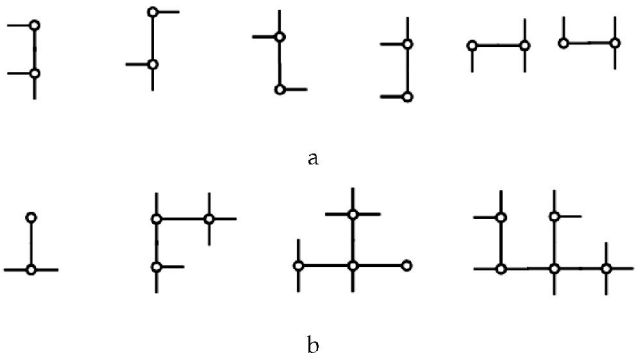


Fig. 5. (a) Six of the 98 size 2 groupings of primitive junctions. (b) A few larger subgraphs found on tax forms.

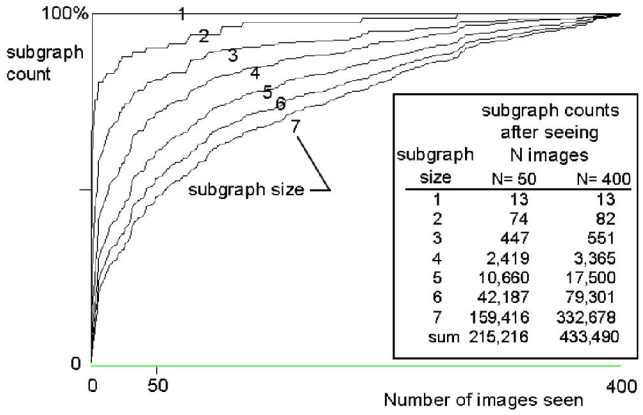


Fig. 6. Consideration of the number of distinct subgraphs contained in data graphs as a function of number of data graphs sampled from a larger corpus. The table inset shows counts of distinct subgraphs of sizes 1 through 7 for number of NIST document samples equal to 50 and 400. The graph plots number of distinct subgraphs encountered after N documents as a percentage of the number of subgraphs encountered in 400 documents. Plateauing of these curves indicates when enough samples have been seen to capture a large fraction of the consistent subgraphs present in the corpus.

about whether two images should be assigned to the same category. This confusion region is due to the noise and variability in these images, given that some pairs of different forms happen to be designed to contain approximately the same number of terminations, L-junctions, T-Junctions, and crossings (X-junctions). There are many possible ways to assess the similarities/differences between feature vectors—in this case between 13-dimensional vectors of junction type counts. The similarity measure used here, called common-minus-difference (CMD), is described below. Euclidian distance and cosine distance both produce less distinct histogram modes.

Ideally, highly informative feature vectors and similarity measures would produce two well-separated modes in the histogram. This would enable a very straightforward greedy algorithm for clustering and classifying images, namely, if pairwise similarity is less than a threshold, then the images are in the same category, and if greater than a threshold, they are in different categories. If the same-category and different-category modes were well separated, threshold values could be determined from the histogram automatically.

2.3 Subgraph Mappings as Image Features

More complex features are obtained by considering not simply counts of primitive junctions, but counts of pairs, triples, and larger collections of junctions. This is in some ways analogous to the use of higher order statistics in image texture analysis [10]. Junction pairs correspond to second-order, triples to third-order statistics, and so on. Given the constraints of how junctions may link to one another, there are 98 ways that the 13 primitive junction types may be grouped pairwise, as suggested in Fig. 5a. Beyond size two, the number of possible subgraphs grows exponentially with subgraph size. Larger subgraphs provide increasing constraints on how the primitive junctions in the observed data graph are linked, up to the largest possible subgraph, which is the data graph itself. Feature vectors incorporating larger subgraphs can therefore be more informative than primitive junction counts.

The number of possible subgraphs increases exponentially with subgraph size, but fortunately the space of possible subgraphs is only sparsely populated by subgraphs actually observed in data. In adding subgraph features, it is useful to utilize only these, and we may comfortably omit nonobserved subgraphs from the feature vector. Fig. 5b presents a few of the many subgraph features of different sizes observed in NIST tax form data.

Section 4 introduces a strategy for entertaining larger—and therefore potentially highly discriminative—subgraphs based on observed data graphs. But first let us consider the results of extending the feature vector to count subgraph matches for subgraphs of sizes two, three, and four. The first step in the process is to build the feature representation. This starts by choosing at random a subset of sample observed data graphs, which would be expected to include representatives of most subgraphs that the full dataset will contain. This subset should contain one or a few examples of each image category. For these, we extract and catalog every subgraph of the target size. Fig. 6 shows the number of subgraphs encountered per subgraph size as a function of sample size for NIST images. The counts plateau sufficiently at 50 samples that this number of images suffices to build a useful vocabulary of subgraphs for clustering and classification.

2.4 Reweighting Mapping Counts for Overlapping Subgraphs

Given a feature measurement vector defined by subgraphs of size 1 to D, where D is a maximum subgraph size, we measure counts of matches to an observed data graph. Each match produces a mapping of subgraph nodes to data graph nodes. One such mapping is illustrated in Fig. 10. Efficient means for performing the requisite subgraph matching is described in Section 3.

In our experience, straight subgraph match counts do not constitute a good feature vector for comparing data graphs. The reason has to do with overweighting of larger subgraphs. Fig. 7 illustrates. For larger subgraph features, a very large number of highly overlapping subgraphs are matched. Any node (line-art junction) in the data graph will participate in many more high-order subgraphs than low-order ones. This leads to instability in large numbers of

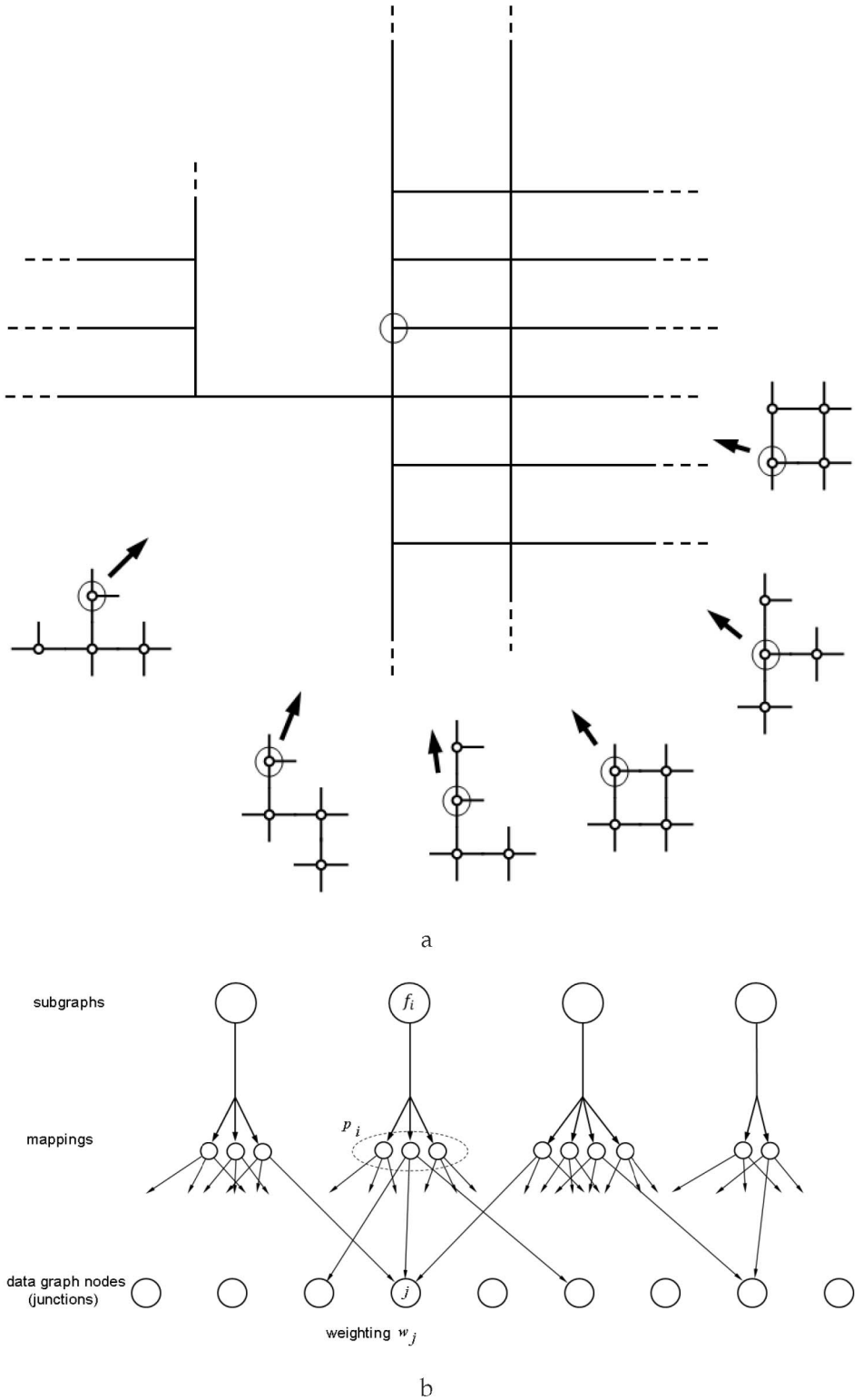


Fig. 7. (a) The circled junction in the data graph is covered in overlapping ways by mappings by the subgraphs shown, plus more not shown. Such overlaps cause junctions or regions to become unevenly represented in a mapping count feature vector. (b) In the junction-normalized reweighting method, the multiple mappings onto each observed junction are counted and used to reweight each mapping's contribution to the match count vector.

match counts as a result of even small numbers of errors in detecting primitive junctions.

A solution is to reweight subgraph match counts according to overlaps in the junctions they map to. Let ${}^{d}f_i$ denote the ith subgraph feature of size d primitive junctions, in a set of features. Let ${}^{d}P_i$ denote the set of mappings of ${}^{d}f_i$ onto an observed data graph. $|{}^{d}P_i|$ is the number of such mappings. Each junction in an observed data graph is then given equal weight toward building a feature vector of normalized counts of mappings of subgraph features. A junction's weight is distributed over the subgraph matches that cover it. The vector of mapping

TABLE 1
Notation and Terms

${}^d f_i$	The ith feature in a set of features, corresponding to a subgraph of size d (number of primitive junction nodes = d); therefore placed at level d in a graph lattice.
${}^d P_i$	The set of mappings of feature ${}^d f_i$ onto an observed data graph.
${}^d w_j$	Weighting of primitive junction j in an observed data graph. Any subgraph ${}^d f_i$ (of size d) will receive this much contribution to the *Junction Normalized Mapping Count* for feature i for each mapping of the subgraph that includes this junction.
v_i	*Junction Normalized Mapping Count* feature vector value for subgraph feature i.
$s(\bar{v_1}, \bar{v_2})$	CMD (Common-Minus-Difference) similarity measure for comparing two positive real valued vectors v_1 and v_2.
M	Set of model data graphs.
$\lvert f_i \mapsto M \rvert$	Number of model data graphs that feature f_i maps to in at least one location.
ω_i	Weighting factor for subgraph feature i, used for weighted voting indexing into a set of reference data graphs. Weighted voting is an alternative to feature vector comparison in the use of subgraph features.
ρ	Performance measure for comparing rankings of category assignments between a candidate algorithm and an imperfect "groundtruth" reference process.
z_i	Score given to candidate feature i for selective growing of a graph lattice based on features' contributions to discrimination among model data graphs.

counts is normalized through these weights. This *Junction Normalized Mapping Count* operates independently for sets of subgraphs in the graph lattice having the same size. In other words, all of the subgraph mappings ${}^d P_i$ for subgraphs of size d are computed, and these are used to normalize mapping counts for all mappings of these subgraphs. (Table 1 provides a summary of notation used in this paper.)

For subgraph size d, a weighting ${}^d w_j$ is computed for each junction j in the observed data graph:

$$ {}^d w_j = \frac{1}{|{}^d P : j|}, \tag{1} $$

where $|{}^d P : j|$ refers to the number of mappings from all subgraphs of size d that include junction j.

Then, the junction-normalized mapping count element v_i for subgraph f_i of size d is

$$ v_i = \sum_{{}^d P_i} \sum_{j \mapsto {}^d P_i} {}^d w_j, \tag{2} $$

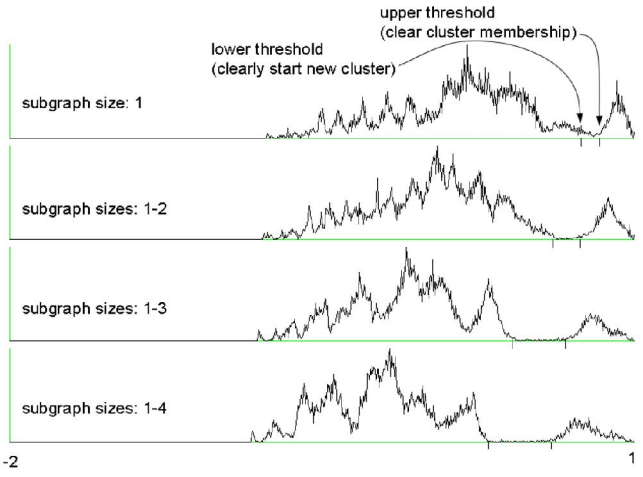
where $j \mapsto {}^d P_i$ means summation over data graph junctions j that are mapped to by subgraph f_i. Through this reweighting, if a junction is mapped only once or a few times, it gives a strong contribution to the counts. If, on the other hand, a junction is covered by many overlapping mappings, these mappings must all share that junction's contribution weight. The junction normalization formula prevents subgraph features from dominating the feature vector when they happen to have many overlapping mappings, which can occur, for example, in grids or other repeating line-art image structure.

2.5 CMD Distance for Comparing Mapping Count Feature Vectors

Another consideration is the similarity measure used to compare junction-normalized feature vector (subgraph match) counts. Obvious choices are euclidian distance and cosine distance. We have found that neither of these works as well as another similarity measure, CMD:

$$ s(\bar{v_1}, \bar{v_2}) = \frac{\sum_i (\min(v_{1,i}, v_{2,i}) - |v_{1,i} - v_{2,i}|)}{\max(|G_1|, |G_2|) * D}, \tag{3} $$

where G_k is the size (number of junctions) of data graph k and D is the number of subgraph sizes considered in the junction-normalized feature vector. While the popular cosine distance is designed to compare distributions or relative values of vector elements, CMD distance also compares absolute magnitudes on an element-by-element



a

subgraph size(s)	feature vector size	errors	spurious clusters	cluster quality (edit distance to G.T.)
1	13	6	18	24
1-2	87	3	5	8
1-3	534	0	1	1
1-4	2953	0	1	1

b

Fig. 8. (a) Histograms of pairwise CMD distances for 200 NIST documents based on feature vectors of subgraph match counts, ranging from size 1 subgraphs only through subgraphs of sizes 1 through 4. Small vertical lines show automatically computed thresholds (clearly within-cluster and start-new-cluster) for greedy clustering. (b) Clustering results for 11,185 combined NIST SpecialDatabase6 and SpecialDatabase2 images.

basis. Whereas cosine distance will give credit to any pairs of feature elements that both have positive count, the behavior of CMD is more strict. Positive credit is given to the extent the count is similar, but negative credit is given to the extent the counts differ. The normalization term of the CMD distance depends not just on the number of feature vector elements, but on the maximum possible value each element could take. Thus, it depends on the sizes of the data graphs being compared. The range of the CMD similarity measure is -2 (minimum, least similarity) to 1 (maximum, best similarity).

2.6 Subgraphs of Increasing Size Are Effective Image Features

Under the junction-normalized feature count and the CMD similarity score, we find that higher order subgraph features do indeed lead to improved discrimination. Fig. 8a presents pairwise similarity histograms for feature vectors up to subgraph size 4. For the NIST data, beyond subgraph feature size 2, different image categories are clearly separated. Under this representation using feature vectors comprising subgraphs of sizes 1-3 or 1-4, a simple greedy clustering algorithm correctly sorts all 11,185 NIST images into their respective 20 categories, with one category split into two. (This category is US Tax Form 2441, which contains only 65 junctions.)

Clustering results are presented in Fig. 8b. Clustering uses two automatically computed thresholds. If an observed data graph's feature similarity to a cluster's centroid exceeds the upper threshold, the document is added to that cluster. If feature similarity falls below a lower threshold for all existing clusters, a new cluster is started. Otherwise, the image is put aside into an "unknown" category until all images have been considered. This process is iterated with the unknown images until no more can be assigned to a cluster based on the upper threshold. Finally, each unknown image is assigned to the best-matching cluster.

Quality of clustering is scored as the edit distance to the ground truth correct assignment of images to categories. One edit operation is tallied for each incorrectly classified document, and one edit operation is tallied for merging any two clusters representing the same ground truth category. Under this procedure, forms clustering and classification is almost 100 percent correct for subgraphs of size 3 and larger; the only error is an extra cluster duplicating one of the ground truth categories. As reported in [22], to our knowledge this is the best result reported to date on the benchmark NIST forms dataset, improving even on supervised classifiers which by definition are supplied with a known number of categories [19], [20], [24].

3 Graph Lattice Data Structure

3.1 Nodes and Struts

A program architecture for computing features representing large numbers of subgraphs should support two main purposes: 1) efficient computation of subgraph (feature element) matches to observed data graphs, and 2) effective construction of more complex features (larger subgraphs) from smaller ones. The graph lattice fulfills these needs.

The basic element of a graph lattice is the graph lattice node, as distinguished from a node of a data graph (which is in this domain a line-art junction or termination). A graph lattice node represents a subgraph and its relations to larger and smaller subgraphs in the lattice. Additionally, it can maintain storage of its subgraph mappings onto data graphs. As a matter of terminology, let us call smaller subgraphs *parent* nodes, and larger subgraphs generated from them by adding junctions, *child* nodes. In general, two subgraphs of arbitrary size could be conjoined to create a larger subgraph, perhaps of size the sum of the sizes of parent nodes, or perhaps smaller if overlap is allowed. The present work does not consider such a general mechanism. Instead, we only support single-level links in the graph lattice representing the accretion of a single junction (data graph node) at a time. In other words, child graph lattice nodes are always subgraphs that are larger by one than their parents.

The relations among graph lattice nodes are maintained by *struts* (see Fig. 9). The purpose of a strut is twofold. First, it maintains the junction index mappings between a parent and child node. In general, any graph lattice node will index its component junctions in arbitrary order, and a strut keeps these organized between parent and child graph lattice nodes. Second, a strut indicates the primitive type, placement on the parent, and links for the junction that constructs the child from the parent.

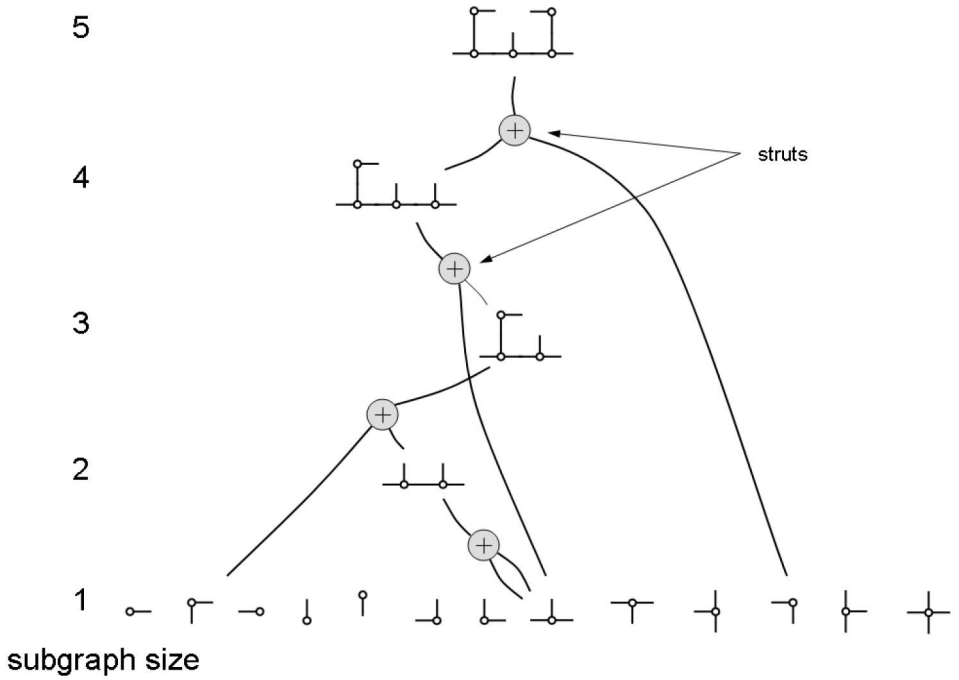


Fig. 9. A graph lattice consists of layers of subgraphs related by addition or deletion of primitives (junctions, in the case of rectilinear line art). These relations are maintained through *struts* maintaining the mappings and linkages formed by incrementally added primitives.

3.2 Sweep-Upward Matching

Matching of the complete set of nodes in a graph lattice to an observed data graph is efficient because each graph lattice node (subgraph) can be incrementally mapped based on mappings of its parents (see Fig. 10). Processing proceeds bottom up, layer-by-layer, in a *sweep-upward* algorithm. At layer subgraph-size $= d$, for each subgraph feature node ${}^{d}f_i$, subgraph mappings are computed as follows: Choose arbitrarily one parent node at layer $d-1$. The strut to this parent defines the possible mappings of ${}^{d}f_i$ onto the data graph, as these can be inherited and read off directly from the mappings stored by the parent. The strut also defines the single extra data graph junction that node ${}^{d}f_i$ contributes. This method essentially follows the approach used in the VF2 subgraph matching algorithm of Cordella et al. [7]; it is easy to test the data graph to see if it contains such a junction, properly linked, to support a complete mapping of ${}^{d}f_i$. In our Java implementation, using the sweep-upward algorithm, the computation of the junction-normalized feature vector for a graph lattice of 2,953 graph lattice nodes (subgraphs) averages 43 milliseconds per data graph[2] over a set of typical NIST tax form documents. Using the sweep-upward algorithm, subgraph matching takes approximately 1 microsecond per subgraph for subgraphs ranging up to size 4; this is a speedup of better than $10\times$ over brute-force attributed subgraph matching.

3.3 Growing a Graph Lattice

This approach trades storage necessary to hold the graph lattice against faster matching time via the sweep-upward algorithm. The number of *possible* subgraphs grows exponentially with subgraph size. This explosion is managed through the judicious selection of a relatively smaller number of subgraphs represented explicitly as graph lattice nodes, among the space of all possible subgraphs at various sizes. Any data domain will reflect characteristic patterns as more or less commonly occurring subgraphs, while many other possible subgraphs will not be observed at all. Thus, the selection of graph lattice nodes constitutes knowledge

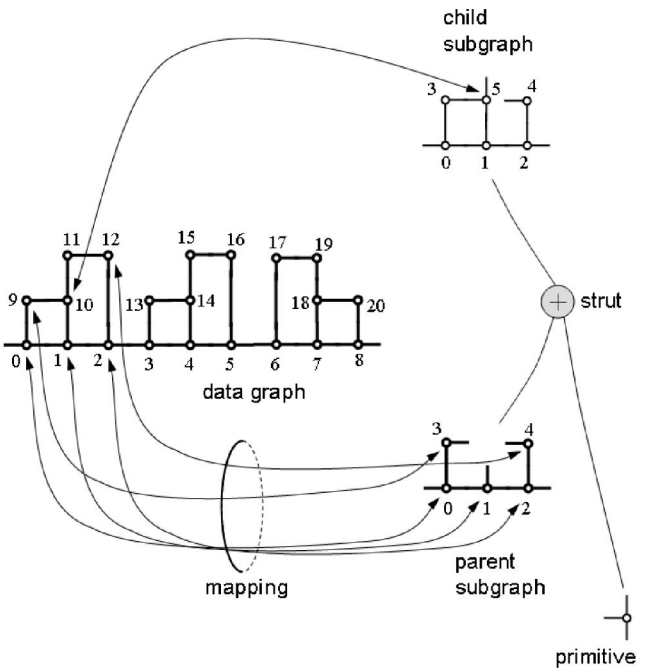


Fig. 10. In the sweep-upward algorithm, subgraph mappings are inherited from the previously computed subgraph mappings at the parent level. Struts maintain bookkeeping data structures, enabling only a quick check of whether the incremental junction is present in the data graph (upper mapping arrow).

2. Single core, 2.4 GHz PC, Java.

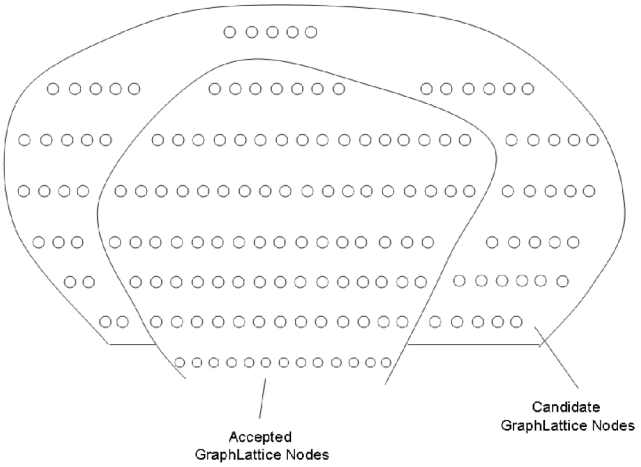


Fig. 11. A graph lattice can be constructed by promoting Candidate nodes to Accepted status. Accepted nodes are eligible to spawn more candidates.

about the data domain in the form of the subgraph features maintained in the graph lattice.

As a general framework, a suitable approach to constructing graph lattice representations is based on the following idea: Incrementally construct, evaluate, and selectively accept Candidate graph lattice nodes, which in turn accrete to an existing simpler graph lattice of Accepted nodes, based on observed examples (see Fig. 11). The algorithm is to iterate the following steps:

Step 1: Generate Candidate graph lattice nodes from Accepted graph lattice nodes and observed data.

Every mapping of a Level d subgraph ${}^{d}f_i$ onto an observed data graph can serve as a seed for spawning new Level $d+1$ graph lattice nodes that are supergraphs (children) of ${}^{d}f_i$. Each primitive junction linked to the perimeter of the subgraph's mapping can spawn one of these children. Step 1 of the graph lattice construction algorithm is to examine mappings of Accepted graph lattice nodes onto observed data graphs and spawn new Candidate graph lattice nodes based on actual supergraphs encountered in the data. This step may involve mapping the existing graph lattice to previously seen data graphs or to new, previously unobserved data graphs. This process involves some bookkeeping to follow subgraph mappings, detect duplicate Candidates nodes, and build struts to all of a Candidates nodes' parents at Level N.

Step 2: Select Candidate graph lattice to promote to Accepted graph lattice nodes

The second step of the graph lattice construction algorithm is to select certain Candidate graph lattice nodes for promotion to Accepted status. The forms image clustering work reported in Section 2 employs a trivial candidate selection strategy: Accept *all* Candidate nodes up to a certain level. Section 4 presents a feature selection strategy designed to accept many fewer Candidates, specifically only maximally distinguishing features for a set of model graphs.

Depending on the acceptance strategy used in Step 2, various stopping criteria may become meaningful, including:

1. The graph lattice contains a threshold number of Accepted graph lattice nodes at a given level.
2. The graph lattice contains a threshold number of Accepted graph lattice nodes in total.
3. The list of Candidate graph lattice nodes is exhausted.
4. Quality measures for Candidate nodes falls below a threshold.

For the NIST dataset, the time required to build a graph lattice up to a layer of subgraphs of size 4 from the data graphs of 50 randomly sampled images is 3 seconds. To build a graph lattice of size 4 from a sampling of 1,000 NIST images takes 50 seconds, resulting in 4,560 graph lattice nodes.

4 Feature Voting

4.1 Voting Motivations and Methods

An alternative in pattern classification to comparing fixed-length feature vectors is feature voting. Classification takes place by indexing into a set of exemplars, or *model* data graphs. One or more exemplars may be provided per category. Feature voting is an attractive way to use subgraph features for several reasons.

First, the graph lattice data structure and growing algorithm support an efficient mechanism to build and match large feature vocabularies that include larger subgraphs and, in particular, subgraphs specializing to unique or highly indicative features of the data categories. With increasing graph lattice size, most subgraph features will not match to any given test data graph, and hence the match feature vector will be sparse. Through feature voting the classification procedure avoids wasteful computation on large numbers of 0 entries in the feature vector, which would arise from absent features.

Second, by the use of inverted indices, feature voting facilitates scaling to large numbers of model categories by shifting from linear feature vector scoring of every category, or, alternatively, the use of complex hashing schemes, to vote-based consideration of only categories that obtain votes.

Third, under the graph lattice approach to subgraph matching, feature voting allows more precise exploitation of features through testing of location and dimensional attributes of subgraph features that share common topological structure.

Feature voting is illustrated in Fig. 12. Assume a graph lattice of subgraph features has been constructed. Initially, each model data graph representing a set of target categories is matched to the subgraph features using the sweep-upward algorithm. For each subgraph match, the mapping from line-art junctions and terminations of the feature subgraph to the line-art junctions and terminations of the model data graph is stored. Then, when a test data graph is presented, it is matched to the subgraph features using the sweep-upward algorithm. For every subgraph mapped, a vote is recorded for each of the model data graphs that were also mapped by that subgraph.

4.2 Increasing Feature Precision with Dimensional Geometry

Two modifications are introduced to enhance this basic method. The first is to constrain votes only to truly corresponding subgraph structure in the model and test

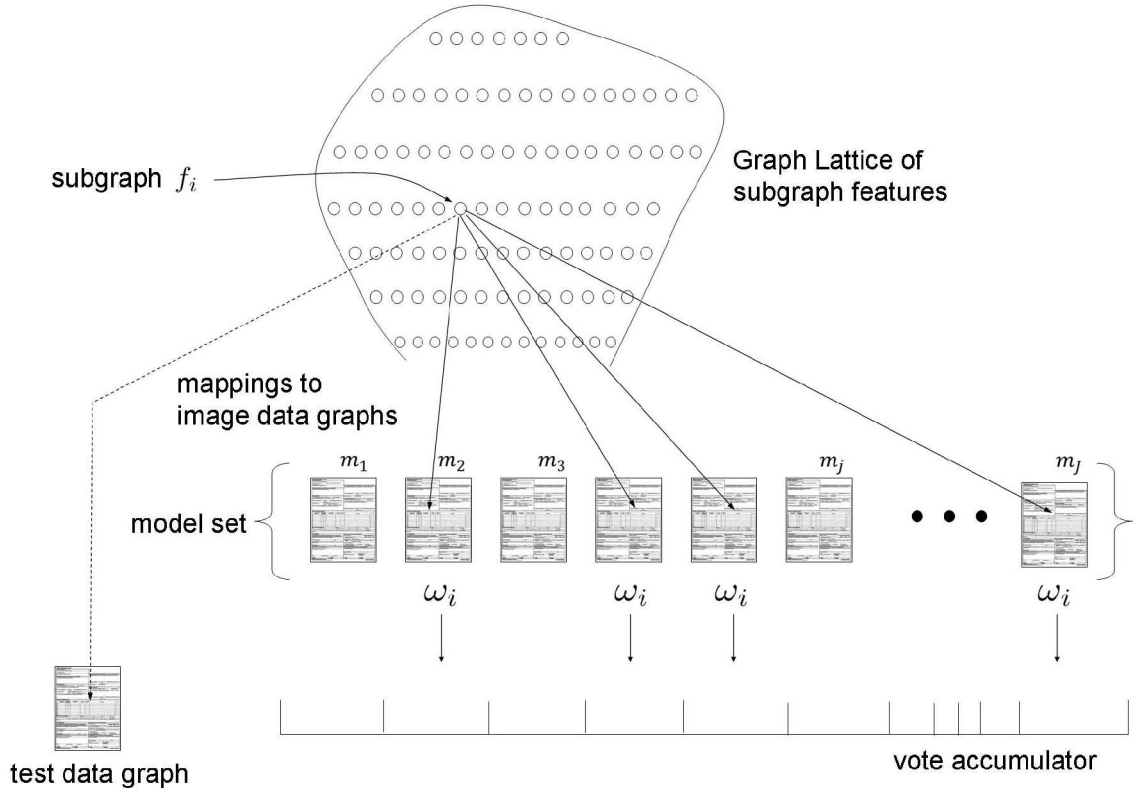


Fig. 12. Illustration of weighted voting. Each subgraph feature initially is matched to all model data graphs (solid arrows). The junction mapping of each match is recorded. Then, when a test data graph is presented, each subgraph feature is matched (dashed arrow), and each mapping is compared with mappings to the models. A weighted vote is tabulated for each model that holds a mapping meeting threshold tolerance on correspondence to the test data graph mapping on location and dimensional attributes.

data graphs. Constructed with struts, the graph lattice representation for families of subgraphs in the rectilinear line-art domain retains only topological and directional geometrical properties—that is, which typed junctions are connected to which others at their respective attachment points. Identical subgraph structure might be found on test and model data graphs corresponding to geometrically and positionally very different image features. It is important to prune these false matches when voting. This can be done by introducing dimensional geometry and location as filters on votes for corresponding subgraphs. Fig. 13 illustrates an effective method for doing this. Each mapping between a feature subgraph and a data graph retains a geometric signature and location for the configuration of line-art junctions and terminations of the subgraph as it occurs in the data graph. If a given subgraph feature maps onto the data graph in multiple places, each becomes a separate mapping that retains its own geometric signature and location. It has proven sufficient to maintain a configurational signature based on the absolute values of (x, y) displacement of each junction or termination from the centroid of junctions/terminations. The centroid itself is an adequate representation for global positioning of the subgraph on the model or test data graph. Filtering of Candidate nodes' mappings is performed based on thresholds of differences between the signature components between model and test data graphs. To achieve scale invariance, displacements are normalized by the maximum of height and width of the subgraph occurrence in the data graph.

4.3 Weighted Voting

A second modification is to perform weighted voting instead of uniform voting. This idea is based on the observation that in many datasets it is typical to encounter subgraph features representing image structure that is common across many models, while other subgraph features will be rare and therefore highly indicative of their respective models. It is possible to weight the latter more highly through the use of a weighting factor ω for each feature match vote:

$$\omega_i = \frac{1}{|f_i \mapsto M|}. \tag{4}$$

$|f_i \mapsto M|$ is the number of model data graphs that subgraph feature f_i maps onto. (Equivalent notation for $|f_i \mapsto M|$ is $\sum_{j:f_i \mapsto m_j} 1$, where M is the set of all model data graphs. If $|f_i \mapsto M|$ is 0, then some suitably small number is used in the algorithmic implementation.) In practice, we have observed that the use of weighted voting improves accuracy slightly over uniform voting, but it improves speed considerably because the selective graph lattice growing algorithm described in Section 5 chooses more selective features contributing fewer votes for each test image. Representative processing time for forms classification into 208 categories by weighted voting is 50 milliseconds per data graph for a graph lattice of 7,600 subgraph features, excluding image processing time to compute the data graph itself (dataset discussed below).

Some subgraph features, especially smaller ones, tend to map to sites common across very many model data graphs.

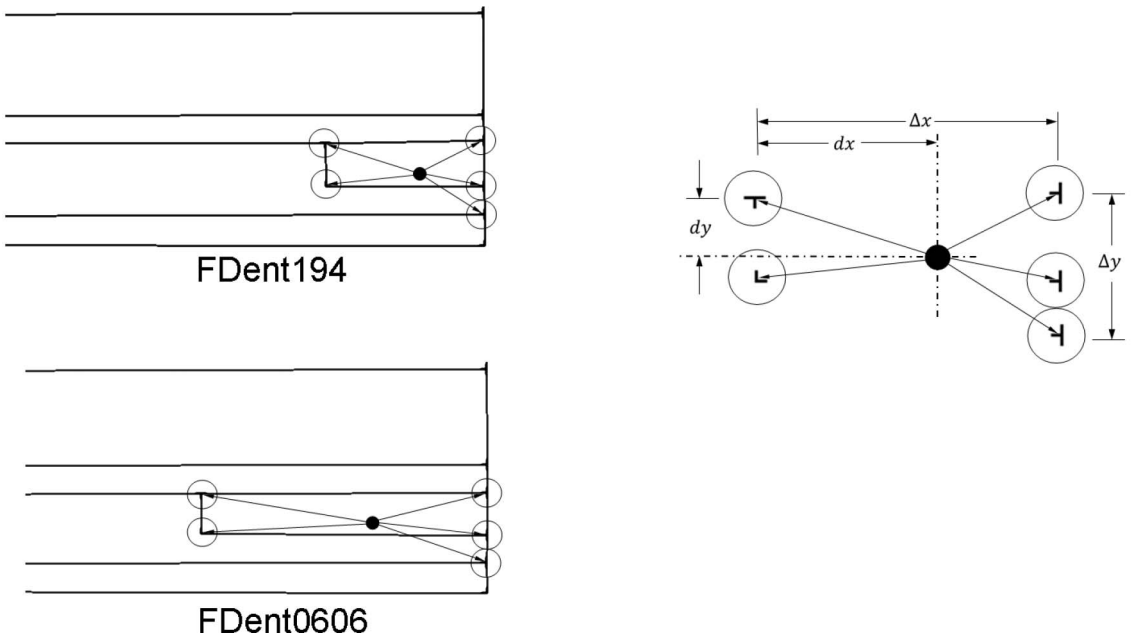


Fig. 13. Portion of two model data graphs (see Fig. 14) both of which match, topologically, a particular subgraph feature of size 5. Spatial configuration attributes of a subgraph feature mapped to a observed data graphs are captured by the absolute values $(|dx|, |dy|)$ of displacements of line-art junctions from their centroid, normalized by $\max(\Delta x, \Delta y)$.

Their vote weights are small, and the memory requirements for maintaining the mappings become large. Therefore, for practical purposes we set a threshold $(\text{value} = 100)$ on the maximum number of mappings stored and, hence, minimum feature weight for a subgraph feature to participate in voting. The mappings of features falling below this threshold are discarded.

4.4 Weighted Voting Performance: Simple Graph Lattice

While the subgraph feature vector comparison method of Section 2 performs extremely well on the benchmark NIST dataset, classification of real-world data can prove more challenging. One such dataset employed in this study is a corpus of 6,247 American Medical Association (AMA) dental claims forms encountered in a production document processing application. The line art from these forms is made available at [21].

The AMA dental forms dataset includes 208 labeled blank forms which are used to build the graph lattice feature vocabulary, and the line art from 6,247 scanned forms to be classified. The forms fall into four major categories, with a few dozen subcategories and a few dozen other minor categories. But, due to business requirements, the application conditions prescribe 208 categories of subtypes, for which one data sample, or model form, is provided for each category. It is necessary to classify the forms as accurately as possible because a subsequent processing step involves identifying data fields which depend on alignment with a correct form template. The problem is complicated because some of the subtypes are distinguished by very subtle differences, as seen in Fig. 14, and other subtypes are actually duplicate scans of the same underlying form subtype. We estimate that the true number of categories is approximately 156. Only about half of these subtypes are represented in the test dataset.

A ground truth ranking of model categories has been provided, the details of which lie outside the scope of this investigation. To summarize, however, a semi-automated process assigns one or more candidate categories to each document along with a confidence score. A manual verification step allows correction of assignments and adjustment of confidences. From the confidence scores, a ranking for ground truth category membership can be derived. But, as can occur in real-world applications, the ground truth is not 100 percent correct, nor can it be because of ambiguity due to apparent duplication of some of the model categories. The semi-automated scoring process includes a reject procedure such that a large fraction of images are given reject status, mainly due to poor scanned image quality, leaving a total of 4,943 images for testing and evaluation of our classification algorithms.

A suitable performance measure for classification accuracy is based on comparing the rank ordering of category model matches to image data graphs between ground truth and classifier outputs. The performance measure applies under either the feature vector comparison or weighted voting methods. Let $r_{g,c}$ be the ranking assigned by a classifier to the model assigned top rank in the ground truth. Let $r_{c,g}$ be the ranking in the ground truth of the model assigned top rank by the classifier. Then, the performance measure ρ is

$$\rho = \frac{1}{2}\left(\frac{1}{r_{c,g}} + \frac{1}{r_{g,c}}\right). \tag{5}$$

Under this performance measure a maximum score of 1 is given when the top-ranking categories agree, but some credit is given when the top ranking category of the ground truth or classifier output scores highly in the complement rankings (classifier or ground truth, respectively). Equation (5) applies to strictly ordered rankings, which normally obtains under weighted voting. Under uniform feature voting there may be ties, with two model categories receiving the same number of

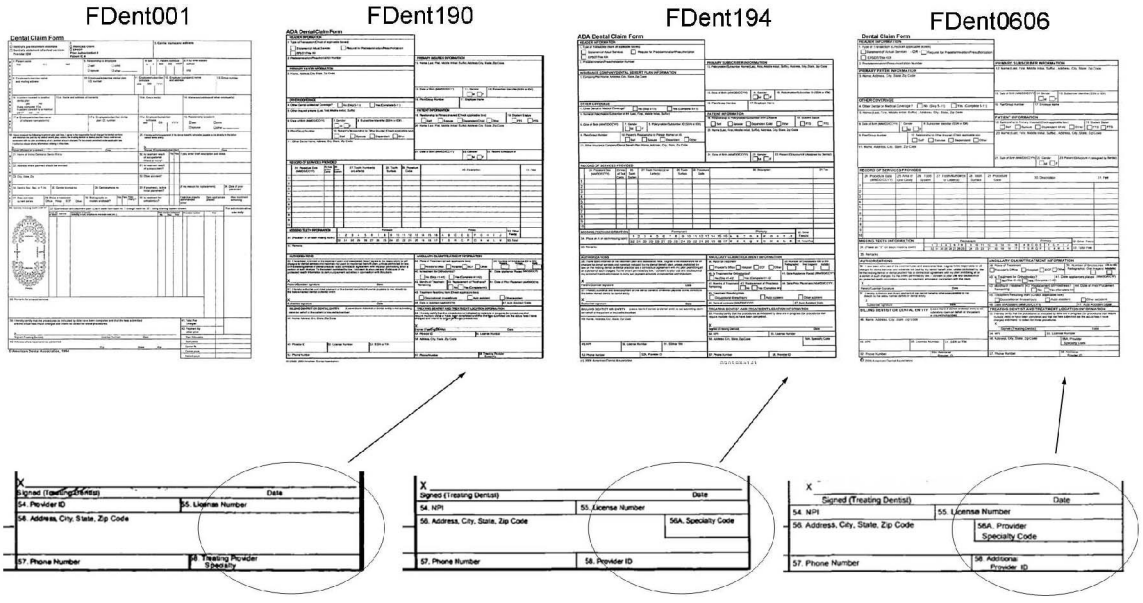


Fig. 14. Four exemplar model images from a set of 208 categories of AMA dental claims forms. While some forms are very distinct, others are different only in subtle details. Some subtle differences occur in the qualitative presence or absence of small line-art structure while others are minor dimensional differences in placement of rule lines.

votes. The performance measure can be extended to these cases by discounting each term by the number of models vying for the same rank as follows:

$$\rho = \frac{1}{2}\frac{1}{|r_{g,c}|}\left(\frac{1}{r_{c,g}} + \frac{1}{r_{g,c}}\right), \tag{6}$$

where $|r_{g,c}|$ is the number of categories assigned that ranking.

Table 2 presents classifier results on the AMA dental forms corpus for all methods described in this paper. Let us concentrate first on fixed feature vector comparison and weighted voting results for subgraph features constructed exhaustively up to a fixed size. Under the fixed feature vector comparison method described in Section 2, Test pairs 1/5, 2/6, 3/7, and 4/8 show that the CMD distance clearly outperforms cosine distance.

Classifier performance clearly improves as the size of the subgraph feature vocabulary increases from subgraph sizes 2 through subgraph size 5, which are obtained by expanding exhaustively to larger subgraphs (Tests 1-4).

Beyond this, the weighted voting method clearly shows an improvement in performance over the fixed-length feature vector comparison method, as shown in Test pairs 2/15, 3/21, and 4/27. (The degradation in performance of weighted voting at subgraph size 2 (Test pairs 1/9) is due primarily to the fact that many such subgraph features are discarded due to their low vote weight, as discussed above.)

5 Selectively Growing a Deeper Graph Lattice

Section 3.3 presents an overall method for growing a vocabulary of subgraph features in a graph lattice. Iteratively, Candidate subgraph features are promoted to Accepted status, then expanded to create new Candidate subgraphs based on observed model data graphs. This method leaves available many options for the selection strategy for deciding which Candidates to promote.

Clearly, candidate subgraph features should be promoted that will be most informative for the classification task. Brown [3] presents an information theoretic framework for feature selection for pattern classification. A principled objective is to maximize the conditional entropy of assignment of test samples to models, given measured features, but minimize interaction information, or redundancy, among the features themselves. This is again measured through conditional entropy given ground truth sample categorization.

This objective is aligned with the following strategy for prioritizing and iteratively accepting Candidate subgraph features based on their ability to distinguish among different model categories, given the current set of Accepted features. In particular, each Candidate feature f_i is given a score z_i:

$$z_i = \frac{\sum_{j,k} \begin{cases} \omega_i & \text{if } f_i \mapsto m_j \\ & \text{and } f_i \not\mapsto m_k \\ & \text{and } |f_i \mapsto M| >= 2 \\ 0 & \text{otherwise} \end{cases}}{v_{j,k}}, \tag{7}$$

$$v_{j,k} = \sum_{i'}\sum_{j,k} \begin{cases} \omega_{i'} & \text{if } f_{i'} \mapsto m_j \\ & \text{and } f_{i'} \not\mapsto m_k \\ 0 & \text{otherwise,} \end{cases} \tag{8}$$

where $f_i \mapsto m_j$ means that feature f_i does map onto model m_j under location and dimensional configurational constraints, and $f_i \not\mapsto /m_k$ means that feature f_i does not map onto model m_k. In other words, the presence of feature f_i is a distinguishing feature in favor of model m_j but not m_k. The expression $v_{j,k}$ sums the weights of distinguishing features among the vocabulary of already accepted subgraph features between models m_j and m_k, and thus indicates

TABLE 2
Performance Measure ρ for Classifying a Corpus of 4,943 AMA Dental Claims Forms into 208 Model Categories for Various Graph Lattice Classifier Test Conditions

Test Number	Test Condition	Extv. G.L. Level	Selective Growth Iterations	Feature subgraph nodes per level											Perf. Measure
				1	2	3	4	5	6	7	8	9	10+	total	ρ
1	CMD comp.	2	0	13	98									111	.411
2	CMD comp.	3	0	13	98	879								990	.467
3	CMD comp.	4	0	13	98	879	6110							7100	.507
4	CMD comp.	5	0	13	98	879	6110	33847						40947	.524
5	cos comp.	2	0	13	98									111	.341
6	cos comp.	3	0	13	98	879								990	.353
7	cos comp.	4	0	13	98	879	6110							7100	.371
8	cos comp.	5	0	13	98	879	6110	33847						40947	.377
9	wtd. voting	2	0	13	98									111	.177
10	wtd. voting	2	10	13	98	9	1							121	.260
11	wtd. voting	2	50	13	98	29	12	2	3	3	1			161	.492
12	wtd. voting	2	100	13	98	38	27	14	9	7	4	1		211	.599
13	wtd. voting	2	500	13	98	62	80	80	62	47	31	26	112	611	.630
14	wtd. voting	2	1000	13	98	78	110	117	117	89	78	77	334	1111	.630
15	wtd. voting	3	0	13	98	879								990	.680
16	wtd. voting	3	10	13	98	879	8	2						1000	.716
17	wtd. voting	3	50	13	98	879	27	15	6	1	1			1040	.730
18	wtd. voting	3	100	13	98	879	38	23	22	9	6	1	1	1090	.735
19	wtd. voting	3	500	13	98	879	89	85	77	53	39	41	116	1490	.741
20	wtd. voting	3	1000	13	98	879	132	144	133	109	95	79	308	1990	.736
21	wtd. voting	4	0	13	98	879	6110							7100	.743
22	wtd. voting	4	10	13	98	879	6110	5	3	1	1			7110	.743
23	wtd. voting	4	50	13	98	879	6110	16	9	8	4	4	9	7150	.740
24	wtd. voting	4	100	13	98	879	6110	29	22	12	12	9	16	7200	.740
25	wtd. voting	4	500	13	98	879	6110	66	66	59	63	49	197	7600	.746
26	wtd. voting	4	1000	13	98	879	6110	88	100	110	105	100	497	8100	.736
27	wtd. voting	5	0	13	98	879	6110	33847						40947	.745

In all cases, the graph lattice was built using a single exemplar per category. CMD comp. refers to feature vector comparison using the CMD distance. cos comp. refers to feature vector comparison using the cosine distance. wt. voting refers to the use of subgraph features by weighted voting. Extv. G.L. level refers to the level, or size of subgraph, grown by exhaustively accepting all Candidate subgraph feature nodes observed in the model set. Selective Growth Iterations refers to the number of Candidate subgraph feature nodes promoted under the selective growth algorithm.

the "need" for additional features to be recruited to distinguish these models. ω comes from (4). The graph lattice growing procedure iterates some number of times, each time promoting the highest scoring Candidate subgraph feature.

A third aspect to note in (7) is that an additional condition is imposed to gate the promotion of Candidate features to Accepted status. A subgraph feature is only promoted if it maps to at least two models. This criterion inhibits promotion of subgraph features that match to noise elements in the model samples on the assumption that different samples are unlikely to share noise in common.

5.1 Weighted Voting Performance: Selectively Grown Graph Lattice

Table 2 presents performance results for feature vocabularies grown by promotion of different numbers of Candidate subgraph features, starting with a graph lattice grown to accept all subgraphs observed in the model dataset exhaustively up to some size (2, 3, 4, or 5). (The explosion of size 6 subgraphs makes it prohibitively expensive to generate the Candidate node set that would be required to selectively add subgraph features from an exhaustive set of subgraphs up to size 5.) As shown clearly in Test sequences 9-14, 15-20, and 21-26, the feature selection strategy described above is generally effective in growing vocabularies that lead to better classification accuracy for a given feature set size.

However, we do observe that accuracy on this dataset does not always increase with the addition of more subgraph features. For example, best accuracy is achieved with a graph lattice grown exhaustively to subgraph size 4 and then grown selectively to add 500 more subgraph features (Test Number 25). This is slightly more accurate than selective addition of 1,000 features. Also, a subgraph

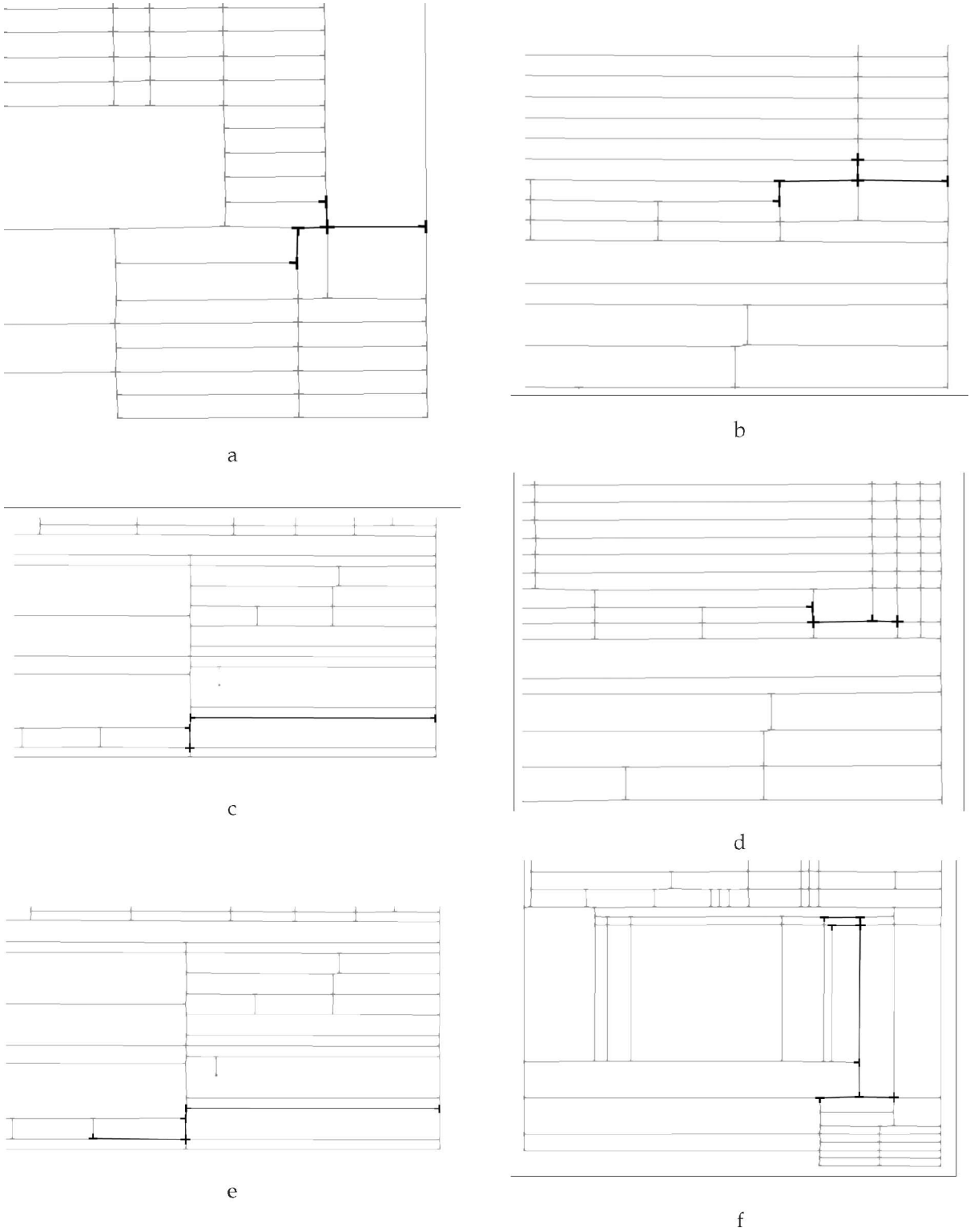


Fig. 15. A sampling of distinguishing subgraph features promoted to Accepted status by the selective learning algorithm, mapped onto model data graphs. (a) and (b) differ qualitatively only in the top-most junction (T versus Crossing). (c) (size 4) is a parent of (e) (size 5). Note that these serve to differentiate model FDent190 from models FDent194 and FDent0606, seen in Fig. 14.

grown exhaustively to only subgraph size 2 does not achieve the accuracy after adding 1,000 subgraph features selectively (Test Number 12, total 1,111 subgraph features) that is achieved by exhaustive growth to subgraph size 3 (Test Number 13, 990 subgraph features). In other words, the feature selective algorithm does not necessarily produce an optimal set of features, based on single exemplar models, for classification of the test data. Note that under the single-exemplar training paradigm, none of the test data are used in building the graph lattice. Degradation of performance with the addition of subgraph features can occur because, in this real-world dataset, the model samples are not fully

faithful representations of the test data images assigned to them by ground truth. After high scoring subgraph nodes under (7) have been added, many additional iterations of the algorithm can focus subgraphs onto small and spurious properties of very similar model exemplars that then bring excessive votes for incorrect model images in the test data. To develop reliable estimates for the optimal number of subgraphs will require a larger undertaking involving many large corpora, which is beyond the scope of this paper.

Fig. 15 presents some subgraph features that were generated as Candidates and subsequently accepted as useful discriminating features by the selective growth algorithm. These were among the first 100 subgraphs accepted after exhaustive growth to level 3 (Test 18 in Table 2).

6 Conclusion

This paper has demonstrated how to build and use large vocabularies of subgraphs as image features maintained in a graph lattice data structure. The graph lattice supports exact attributed subgraph matching and thus sidesteps traditional expensive error-tolerant approaches to graph matching in computer vision. The idea has been applied to the classification of forms documents containing rectilinear line art, where a matching algorithm, called the *sweep-upward* algorithm, efficiently computes the presence of informative subgraph features representing characteristic topological and dimensional structure of local regions of model exemplars and test images. Features can be used either by feature vector comparison or by weighted voting. The feature vocabulary is grown by exhaustive selection of observed subgraphs up to some small size, followed by selective growth based on scoring of subgraph features for their discriminative power. The goal of successful single-exemplar training has been demonstrated on two datasets, the benchmark NIST tax forms dataset and a more difficult corpus of production run dental claims forms.

Three central ideas underlie this work. First, graphs are natural data structures for describing spatial configurations of localized keypoints. Moreover, graphs of different sizes are related hierarchically, not only in trees, but in lattices, which are distinguished from trees by having many-to-many parent-child relationships. A graph lattice data structure comprised of graph lattice nodes (each of which is a subgraph) and struts is an efficient means to build complex structure from simpler constituents. Through struts, successively larger subgraphs are both represented, and matched to image data at the incremental cost of only a single data graph node per subgraph. Thus, a very favorable memory/speed tradeoff is obtained.

Second, knowledge in a domain can be effectively carried by the vocabulary of features used to represent observed instances. The space of possible subgraphs among rectilinear lines is astronomically large, but a given image domain, such as the forms images studied in this paper, produces only a small and tractable subset of these. Under appropriate feature vector comparison, or better yet, weighted voting methods, a feature vocabulary whose knowledge of domain line-art configurations derives from *single exemplar observations* of model categories can achieve superior results for practical real-world classification problems.

Third, under the right construction, exact subgraph matching can be an effective alternative to error-tolerant inexact subgraph matching for computer vision applications. Under the graph lattice data structure, exact subgraph matching is sufficiently memory and time efficient to afford large feature vocabularies of highly redundant and overlapping subgraphs. At least in the document forms domain, this redundancy can achieve noise and error tolerance normally expected for inexact graph matching methods.

To extend the graph lattice approach to other forms of image data would involve first establishing a vocabulary of primitive parts (nodes of a data graph) and relations (links). To exploit exact graph matching, it is advantageous for primitives to map onto natural categorical elements. For example, the primitives for road map data might consist of road terminations (degree 1) plus intersections of degree 2, 3, and so on, without regard to compass orientation. To take another example, for purposes of indexing images of text documents [15], a natural graph is formed by the Voronoi neighborhoods of word centroid locations. Additional attributes could further refine the primitive vocabulary, such as number of characters in a word node.

A interesting topic for further investigation is encoding of data graphs' continuous-valued attributes to support construction of a graph lattice. Section 4.2 shows how dimensional geometry can be used to filter votes for subgraph correspondences, but this takes place after subgraph matching in the lattice. A quantization step on continuous-valued attributes—for example, angles, orientations, and spatial distances between primitives—would entail thresholding to achieve categorical labels. Quantization introduces the potential for categorical errors near threshold boundaries, so methods are called for to introduce error tolerance without exploding computational cost.

Acknowledgments

Many thanks are due to Eugene Bart, Alex Brito, Christina Pavlopoulou, and Prateek Sarkar, and to members of the PARC Intelligent Systems Laboratory for numerous ideas, suggestions, and feedback. Appreciation is also due to Fang Liu for discussions and for building a preliminary implementation of the graph lattice machinery. The reviewers made numerous helpful suggestions for improvements to the paper.

References

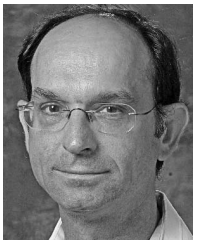
[1] Y. Amit and D.Y. Geman, "Shape Quantization and Recognition with Randomized Trees," *Neural Computation*, vol. 9, pp. 1545-1588, 1997.

[2] A.D. Bagdanov and M. Worring, "Fine-Grained Document Genre Classification Using First Order Random Graphs," *Proc. Sixth Int'l Conf. Document Analysis and Recognition*, pp. 79-83, 2001.

[3] G. Brown, "A New Perspective for Information Theoretic Feature Selection," *Proc. 12th Int'l Conf. Artificial Intelligence and Statistics*, Apr. 2009.

[4] H. Bunke, "Graph Matching: Theoretical Foundations, Algorithms, and Applications," *Proc. Int'l Conf. Vision Interface*, pp. 82-88, 2000.

[5] N. Chen and D. Blostein, "A Survey of Document Image Classification: Problem Statement, Classifier Architecture and Performance Evaluation," *Int'l J. Document Analysis and Recognition,* vol. 10, pp. 1-16, 2007.
[6] D. Conte, P. Foggia, C. Sansone, and M. Vento, "How and Why Pattern Recogntion and Computer Vision Applications Use Graphs," *Applied Graph Theory in Computer Vision and Pattern Recognition,* A. Kandel, H. Bunke, and M. Last, eds., pp. 85-135, Springer, 2007.
[7] L.P. Cordella, P. Foggia, C. Sansone, and M. Vento, "A (Sub)Graph Isomorphism Algorithm for Matching Large Graphs," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 26, no. 10, pp. 1367-1372, Oct. 2004.
[8] D. Dimmick, M. Garris, and C. Wilson, "Structured Forms Database," Technical Report Special Database 2, Nat'l Inst. of Standards and Technology, SFRS, Dec. 2001.
[9] S. Feng and R. Manmatha, "A Discrete Direct Retrieval Model for Image and Video Retrieval," *Proc. Int'l Conf. Content-Based Image and Video Retrieval,* pp. 427-436, July 2008.
[10] B. Julesz, E. Gilbert, and J. Victor, "Visual Discrimination of Textures with Identical Third-Order Statistics," *Biological Cybernetics,* vol. 31, no. 3, pp. 137-140, 1978.
[11] J. Llados, E. Marti, and J. Villanueva, "Symbol Recognition by Error-Tolerant Subgraph Matching between Region Adjacency Graphs," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 23, no. 10, pp. 1137-1143, Oct. 2001.
[12] U. Manber, "Finding Similar Files in a Large File System," *Proc. USENIX Winter Technical Conf.,* p. 2, 1994.
[13] B.T. Messmer and H. Bunke, "A New Algorithm for Error-Tolerant Subgraph Isomorphosm Detection," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 20, no. 5, pp. 493-504, May 1998.
[14] B.T. Messmer and H. Bunke, "Efficient Subgraph Isomorphism Detection: A Decomposition Approach," *IEEE Trans. Knowledge and Data Eng.,* vol. 12, no. 2, pp. 307-323, Mar./Apr. 2000.
[15] T. Nakai, K. Kise, and M. Iwamura, "Use of Affine Invariants in Locally Likely Arrangement Hashing for Camera-Based Document Image Retrieval," *Proc. Seventh Int'l Workshop Document Analysis Systems,* pp. 541-552, 2006.
[16] D. Neogi, S.K. Ladd, and V. Govindaraju, "Systems and Methods for Classifying Electronic Documents by Extracting and Recognizing Text and Image Features Indicative of Document Categories," United States Patent Application 20090116757, May 2009.
[17] D. Nister and H. Stewenius, "Scalable Recognition with a Vocabulary Tree," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 2161-2168, 2006.
[18] A. Pizano, "Extracting Line Features from Images of Business Forms and Tables," *Proc. 11th IAPR Int'l Conf. Pattern Recognition,* vol. III, pp. 399-403, 1992.
[19] K.U. Reddy and V. Govindaraju, "Form Classification," *Document Recognition and Retrieval XV,* Yanikoglu and Berkner, eds., SPIE 2008.
[20] P. Sarkar, "Image Classification: Classifying Distributions of Visual Features," *Proc. 18th Int'l Conf. Pattern Recognition,* pp. 472-475, Aug. 2006.
[21] E. Saund, "Ama Dental Forms Line Art Corpus," http://www2.parc.com/isl/groups/pda/data/DentalFormsLineArtDataSet.zip, 2012.
[22] E. Saund, "A Graph Lattice Approach to Maintaining Dense Collections of Subgraphs as Image Features," *Proc. 11th Int'l Conf. Document Analysis and Recognition,* pp. 1069-1074, 2011.
[23] N. Shervashidze, S. Vishwanathan, T. Petri, K. Mehlhorn, and K. Borgwardt, "Efficient Graphlet Kernels for Large Graph Comparison," *Proc. 12th Int'l Conf. Artificial Intelligence and Statistics,* pp. 488-495, 2009.
[24] C. Shin, D. Doermann, and A. Rosenfeld, "Classification of Document Pages Using Structure-Based Features," *Int'l J. Document Analysis and Recognition,* vol. 3, no. 4, pp. 232-247, 2001.
[25] N. Sidere, P. Heroux, and J.-Y. Ramel, "Vector Representation of Graphs: Application to the Classification of Symbols and Letters," *Proc. 10th Int'l Conf. Document Analysis and Recognition,* pp. 681-685, 2009.
[26] J. Sivic and A. Zisserman, "Video Google: A Text Retrieval Approach to Object Matching in Videos," *Proc. Ninth IEEE Int'l Conf. Computer Vision,* pp. 1470-1477, 2003.
[27] J. Yang, Y.-G. Jiang, A.G. Hauptmann, and C.-W. Ngo, "Evaluating Bag-of-Visual-Words Representations in Scene Classification," *Proc. Ninth ACM Int'l Workshop Multimedia Information Retrieval,* pp. 197-206, 2007.
[28] L. Zhu, Y. Chen, A. Torralba, W. Freeman, and A. Yuille, "Part and Appearance Sharing: Recursive Compositional Models for Multi-View Multi-Object Detection," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 1919-1926, 2010.



Eric Saund received the BS degree in Engineering and Applied Science from the California Institute of Technology and the PhD degree in Cognitive Science from the Massachusetts Institute of Technology. He is with the Intelligent Systems Laboratory, Palo Alto Research Center. His research in computational vision specializes in perceptual organization, with applications in document image analysis, perceptually supported image editing, and video analysis. He was honored to serve as an associate editor of the *IEEE Transactions on Pattern Analysis and Machine Intelligence* from 2006 to 2011. He is a member of the IEEE and the IEEE Computer Society.