Efficient Graph-based Image Matching for Recognition and Retrieval

Praveen Dasigi and C.V.Jawahar
Multimedia Research Laboratory
Center for Visual Information Technology, IIIT Hyderabad
Hyderabad - 500032
Email: {praveend@research.,jawahar@}iiit.net

***Abstract*—Graphs can be used for effective representation of images for recognition and retrieval purposes. The problem is often to find a proper structure that can efficiently describe an image and can be matched in reasonably low computational expense. The standard solutions to the graph matching problem are computationally expensive since the search space involves all permutations of the nodesets. We compare two graphical representations called the Nearest-Neighbor Graphs and the Collocation Trees, for the goodness of fit and the computational expense involved in matching. Various schemes to index the graphical structures have also been discussed.**

I. Introduction

Structural pattern recognition tasks in computer vision, such as classification and object recognition often require that a candidate image is matched to a model or another candidate. During matching, representative structures that are built for each of the images, are compared. In some cases these structures may contain groups of local features or segmented regions. When comparing these structures, it will be useful to encode the spatial arrangement or geometry of the groups of features. This step will bring better meaning to the matching process. One very powerful representation to encode structural properties is a graph. To use graphs for image matching has two key challenges;

- To find a representation that best discrimantes two graphical structures.
- To design a matching process that efficiently matches these representations.

Efficient image matching is the key component for a content based image retrieval(CBIR) system. Usually a CBIR system extracts structural features from images in a dataset and indexes them with the said feature vectors. The retrieval process happens through query by example, where a query image is given and the images which are semantically similar to the query image are expected to be retrieved. When images are represented by graphical structures, the pairwise graph matching scores between images play a very important role in retrieval. Thus, accurate and efficient matching of image graphical structures is a key component in a graph matching based CBIR system. It affects both the precision and efficiency for retrieval

Graph based representations have been employed in various areas such as image analysis, document processing and content based image retrieval(CBIR) ([1], [2], [3]). In most cases the solution will require the image to be represented as a set of "parts". The parts are usually segmented regions described by a local descriptor. The graphical structure encodes the relationship among the parts in terms of nodes and edges. Given two such graphs, the matching process will find the best mapping that maximizes a similarity score. Recent approaches that have aimed at capturing adjacency information used co-occurrence patterns and modelled groups of features as visual phrases [4], or have exploited the invariant properties of the data [3] and built graphs. In any case the solution should be able to satisfy the four properties. The representative structure should be repeatable under considerable distortions (repeatability). It should not overfit the candidate so that matching will be hindered (goodness of fit). The matching process has to be tractable and in most cases, cheap enough (complexity). It should be robust enough to be able to scale to larger applications (scalability). In this work we have used a new representation scheme called the collocation tree that models only the necessary adjacency information within groups of features. The matching process tends to be considerably cheaper than full-graph matching. This is because redundancy in node-relationships is eliminated by linking each node with only one nearest node and forming a hierarchical tree. The computational efficiency and performance of this approach

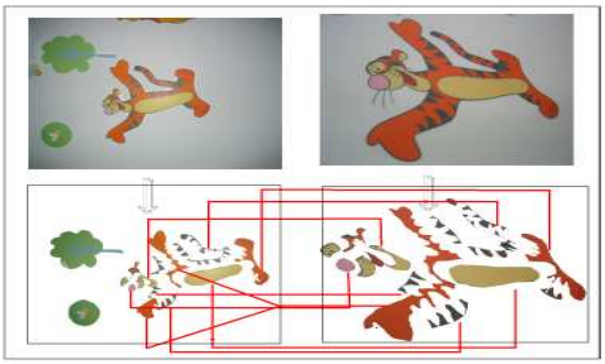


Fig. 1. Figure above depicts the division into parts and the requirement of the matching problem. The arrangement of parts is same even after distortion by view change

is compared with an approximate greedy solution to graph matching in the context of a representation called the nearest-neighbor graph.

Matching Paradigms	
Subgraph Isomorphism	Exact, highly sensitive to feature repeatability, NP-complete, approximations exist, gives only 0/1 match, impractical to real images
Maximal Common Subgraph [5]	Exact, sensitive to feature repeatability, NP-complete, approximations exist, gives % similarity, but scalability is low
Error-tolerant matching [6]	Inexact, all partial similarities treated as errors, decision of cost functions is major issue
Maximum Similarity [7]	Inexact, measures maximum similarity between graphs, comp. expense manageable for smaller graphs, not for large ones
Representation schemes	
Bag of Words (03) [8]	Very efficient feature descriptions, groups of features dealt with, no geometry information encoded
Visual-phrases (07) [4]	Best feature co-occurrences learnt by frequent itemset mining, training needed, dependant on training data
NN-Graphs	Full image structure modelled, redundancies in relationships since only criterion is neighborhood, complexity high for large nodesets, manageable for small ones, fair scalability
Shape adjacency graphs [2]	Sensitive to image distortions, low repeatability, low robustness, impractical on real datasets

TABLE I
TABLE SHOWING BRIEF TAXONOMY OF THE MATCHING ALGORITHMS AND THE REPRESENTATION SCHEMES IN VARIOUS APPLICATIONS, THE STATE OF THE ART

A. Related Work

Table 1 shows a brief taxonomy of literature and the state of art. The application of graphs to pattern recognition (PR) tasks have been dealt in two directions. The first one involves improving the efficiency of the graph matching process to increase its applicability to PR. The second one deals with finding better representations to model the feature sets so that the matching process becomes tractable. In the graph matching front, polynomial solutions to graph matching problems such as graph-subgraph isomorphism [9] and maximum common subgraphs [5] dealt with the problem of finding exact similarity. Since exactness would make the algorithm sensitive to descriptor repeatability, inexact techniques such as error tolerant search using the graph edit distance [6] have been proposed. One classic approach formulated the graph matching problem as a graduated quadratic assignment problem [10]. Another sub-optimal solution is a greedy approach to find the maximum similarity between labelled graphs and has been applied to compare CAD diagrams in [7] From the image retrieval perspective the visual words approach first proposed in [8] provides a very robust set of features which can be extended by encoding the geometry information. This problem has been addressed very recently in [4] where best co-occurrences between visual words are learned by frequent itemset mining and are used for retrieval. In our approach there is no requirement of training data and the CTree proposed here models the hierarchy of collocations. Modelling spatial adjacencies hierarchically is a popular concept in database metric-space literature [11]. The concept has been applied partially to content based retrieval in [12], where a shape is made into parts recursively and adjacency between pixel groups is modelled as a walkthrough.

II. GEOMETRY ENCODING AND MATCHING

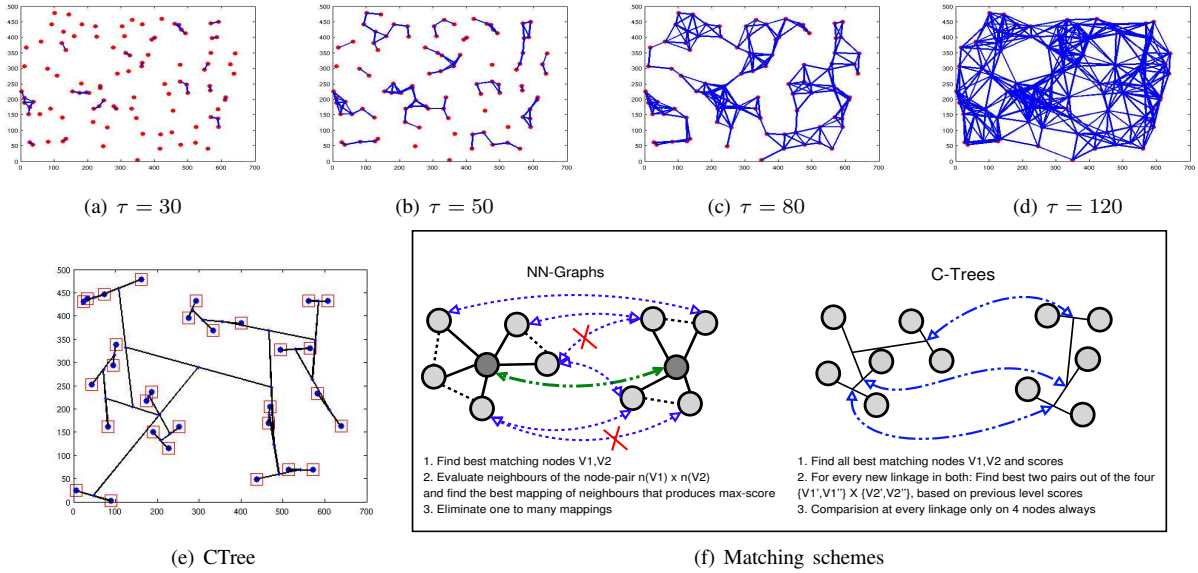
Encoding image-geometry in a graphical structure requires that the 'parts' (nodes) and the 'relationships' (edges) are properly defined. Once the geometry is encoded, matching a pair of graphs is the next challenge. A proper tradeoff needs to be achieved between encoding and matching, so that the matching scheme fully exploits the encoded properties. This section will explore two encoding schemes and their corresponding matching schemes. The two setups compared will emphasize various aspects of graph structure and their effects on the matching scheme. In either case, the parts are required to be invariant under various transformations and possible occlusions, such that the representation is repeatable. Segmenting patches and representation by a descriptor is proposed in [2]. To show invariance to projective distortions we use the MSER region detector [13], which is highly repeatable and robust to various transformations. To describe the detected regions we use the SIFT descriptor which returns a 128-dimensional vector describing the orientation of gradients in the detected region [14].

A. Encoding

Nearest-Neighbor Graphs: One graphical structure that can be used to encode the geometry is the Nearest Neighbor graphs (NN-Graphs). This is constructed by placing edges from each node to nodes in a fixed neighborhood threshold τ. The nodes contain the SIFT descriptor of the interest region depicted by the node. This configuration has the advantage that the relationships in the local neighborhood of a feature will be properly modelled. Even after reasonable projective transformations the local relationships will be maintained. The edge placement between two nodes $\{v_i, v_j\}$ which is represented by the entry in the graph adjacency matrix is

$$A_{ij} = 1 \Longleftrightarrow dist(v_i, v_j) < \tau$$

Here τ regulates the sparseness of the graph, $dist(v_i, v_j)$ is the euclidean distance between the detected regions. The role of τ is that for a very large value, edges are placed between each and every pair of nodes. Since the extra information about the arrangement is encoded only in the presence or absence of an edge, an edge between every pair indicates nothing about edge relationships and thus serves no purpose. Similar case holds with very low values of τ, resulting in almost no edges at all(See **Fig 2**). The most optimal value of τ is one that models only the necessary and sufficient number of edges such that the overall representation is discriminative. For an optimal value of τ, though a reasonable sparseness is achieved, if there are nodes in a large number of small groups scattered throughout the image then within these groups, a fully connected subgraph is formed, which is redundant. The



(a) $\tau = 30$ (b) $\tau = 50$ (c) $\tau = 80$ (d) $\tau = 120$

(e) CTree (f) Matching schemes

Fig. 2. The figure above (a)-(d) show NN-graphs built with different τ value, thus with different densities. (c) is well represented out of the lot but still there is lot of redundancy (e) shows a CTree built on 20 nodes, (f) shows the matching scheme in detail

complexity of matching these graph configurations to find the maximum similarity is approximately $O((|V^i| \times |V^j|)^2)$. For graphs with very large number of nodes this complexity turns out to be a computational bottleneck. This problem is addressed in the graphical model called the Collocation Tree as explained below.

Collocation Trees: The collocation tree (CTree) is a structure which models only the set of necessary relationships thus eliminating the redundancy seen in a NN-Graph. To do so, at every level of the CTree, each unit (node) is only linked to one other unit. Every level groups into pairs, the units formed in the previous level. This successive grouping is achieved through an agglomerative clustering over the initially detected regions. In the first phase of the clustering process pairs of nodes that are within a distance threshold are grouped together. In the next level pairs are made out of the initial groups and so on leaving two groups of nodes in the last-but-one step which are merged into one finally. This is inspired from the concept of distance based indexing in metric space literature [11]

The collocation linkage process can be formulated as follows. Let the total node-set be V consisting N nodes. Assume level l contains the node set V_l. The level $l+1$ is formed by applying a function $link$ over V_l.

$$V_l = \{v_k, \ldots, v_{k+m}\}$$
$$V_{l+1} = link(V_l) = \{v_{k+m+1} \ldots, v_{k+m+p}\}$$
$$\frac{N}{2^{l-1}} \leq m < \frac{N}{2^{l-2}}; \frac{N}{2^l} \leq p < \frac{N}{2^{l-1}}$$

Since the regions are grouped hierarchically based on the nearness, adjacency relationships will be preserved as good as in NN-Graphs. In addition to that each level of grouping will provide a different level of granularity from coarse to fine (See Fig 3). Thus direct region match will only happen at the lowest level. For each consecutive level, the score will be guided by the score from the previous level. The number of comparisons will be greatly reduced thus bringing the expense similar to a normal set-match. Each level provides a matching score at a different resolution. The efficiency of this process is emphasized in the fact that every node will be linked to another node with minimum redundancy thus the matching process will only involve one quadratic assignment (i.e., matching one pair with another) for each linkage instead of trying to match n nearest neighbors of a pair of nodes ([7], [10]).

B. Matching

Given two general graphical structures G_1 and G_2, the first possible method to match them is to find if both the graphs are isomorphic and output a $0/1$ similarity score. This would be exact graph matching when G_1 is a exactly similar to G_2 or is exactly a part of G_2. In complexity literature this is an NP complete problem and some approximate solutions have been found [15] to break the complexity. In this case, exact graph matching will not be able to solve the problem. Due to low repeatability of detectors when there is discrepancy in even one of the nodes, an exact graph matching algorithm will output a 0. The same problem applies using graph-subgraph isomorphism as well. The next possible approach is to find the maximum common subgraph(MCS). Given two graphs G_1 and G_2, the MCS is the graph g that is common to both graphs in terms of node and edge configurations. To find MCS all possible permutations are to be evaluated . Effective steps have been proposed to break the complexity of this problem in [5]. Using MCS will solve the problem to a certain extent as there is more flexibility to the size of the subgraph and the graphs could be of any sizes. However the problem of low repeatability still haunts, as even the parts of the graphs

which are almost similar will be left out. Judging from these aspects it is imperative that the exactness constraint has to be relaxed. Thus the requirement is that the algorithm has to find the best subgraphs in G_1 and G_2 that are as similar as possible and a score has to be given that depicts the similarity of these subgraphs. This is the guiding characterstic employed for matching both the encodings NN-graphs and CTrees)described above

NN-Graphs: Given two graphs $G_1 : \{V_1, E_1\}$ and $G_2 : \{V_2, E_2\}$ each node v_i in G_1 is associated with a set of nodes in G_2 by the function $m(v_i)$ which defines the mapping to identify the related nodes of v_i in G_2. Thus each node will be initially mapped to a set of nodes. The problem is to evaluate all the mappings of nodes, such that an overall mapping is found which results in a maximum similarity for the pairs of nodes within. To find the maximum similarity, the algorithm builds the set of mappings between the node-sets of two graphs G_1 and G_2 . After each of the mappings are found, the process is to evaluate all the combinations of mappings for the best overall similarity score such that this can be used as the maximum similarity between the two graphs. This would find the global maxima of the scores. However this process is also NP-complete. This is because, though all the combinations of all the mappings, will be less than the entire search space i.e., $2^{(|N^i| \times |N^j|)}$, since the mappings are for a reduced number of nodes, even then it will be highly expensive. Thus we adopt an approximate solution by the use of greedy evaluation of mappings. This is outlined briefly in the algorithm 1. This is a variant of the algorithm proposed for comparing CAD diagrams in [7].

To find the maximum similarity, each of the matches and their neighbors are evaluated. For every new match, the pair is eliminated from the search. At each level the match with best score for the node in G_i to the set of remaining nodes in G_2 is evaluated see figII. This greedy approach may not be able to find the most optimal solution. The matched sets and the similarity score will be sub optimal. It runs in polynomial time i.e, $O((|N^i| \times |N^j|)^2)$

CTrees: To match a pair of CTrees only a set of pairs at each level has to be compared. The cumulative score for each level is obtained by the individual similarity scores of the pairs in that level. To efficiently compute the final scores, an $m \times n$ matrix called the guide-matrix G is used. m and n are the number of nodes in either of the trees. Initially the k best matches for each of the first tree's leaf-nodes in the second tree's leaf nodes are computed, along with the matching scores. These are filled in the corresponding entries in the guide matrix. Whenever a linkage from each of the trees is to be compared, the comparison process has to find the best pairs between four nodes say v_{1i}, v_{1j} and v_{2p}, v_{2q}. The four pairs are evaluated and the best two are picked which result in a similarity score which is entered into the corresponding entry of G. For the next level this score is used as a group score instead of evaluating all similarities within the group. The algorithm for matching a pair of CTrees is given in algorithm 2. As mentioned earlier each matching step is a single quadratic assignment.

Algorithm 1 Calculate MaxSim from all mappings

$simmat \Leftarrow$ similarity matrix between all nodes $v_1 \times v_2$
$bestpairs \Leftarrow 0$
for i in all matches m **do**
  for j_a, j_b in all pairs of match i **do**
    $nb_1, nb_2 \Leftarrow$ immediate neighbors of j_a, j_b {P}hase 1
    Find best match bm_1, $nb_1(k)$ in nb_2; retain only global best
    Add $\{j, bm_1\}$ to bestmatches,Eliminate matched pairs
    {P}hase 2
    Find best match bm_2 for $nb_1(k)$ in nb_2
    Add $\{j, bm_2\}$ to bestmatches,Eliminate matched pairs
    {P}hase 3
    Retain best matched pair out of all repetitions
  end for
  $scr(i) \Leftarrow$ avg similarities of pairs in $bestmatches$
end for
$MaxSim \Leftarrow mean(scr)$

Algorithm 2 Calculate Similarity scores of CTrees CT_1 and CT_2

Initialize $gdmat_{m \times n} \Leftarrow 0$ {Guide matrix: $\langle m, n \rangle \Leftarrow$ no of elements in CT_1 and CT_2}
for $i, j \Leftarrow 1 : v_1, v_2$ (Level1) **do**
  $gdmat_{v_2 \times v_2} \Leftarrow L1 - Dist(v_1, v_2)$
end for
for all successive levels l **do**
  for each new node on CT_1 and CT_2 **do**
    $v_{i12} = link(v_{i1}, v_{i2})$ and $v_{j12} = link(v_{j1}, v_{j2})$
    $gdmat(i12, j12) = greedy - max(\langle i1, i2 \rangle \times \langle j1, j2 \rangle)$
  end for
  $sim(l) = avg(gdmat(i12, j12))$
end for

In cases with large number of nodes, where it is not necessary to link the regions right from the lowest resolution, a further approximation can be used. Prior to agglomerative linkage, the regions are clustered by a K-means into small groups. The cluster centers will be used as initial points in the linkage process. Performing this step prior to linkage will prune the bottom few levels of the tree based on the size of the initial clusters. This step can be used to considerably reduce the matching time in cases where very intricate level of detail is not needed.

III. Indexing graph-based structures

Pairwise matching scores within pairs of graphs will be able to classify the different classes of images represented by the graphs. However in the retrieval scenario, naively matching the graph built from the query image to all the graphs in the dataset will be computationally expensive and cannot be used in online retrieval. Thus the graph dataset has to be efficiently ordered with an index structure such that retrieval

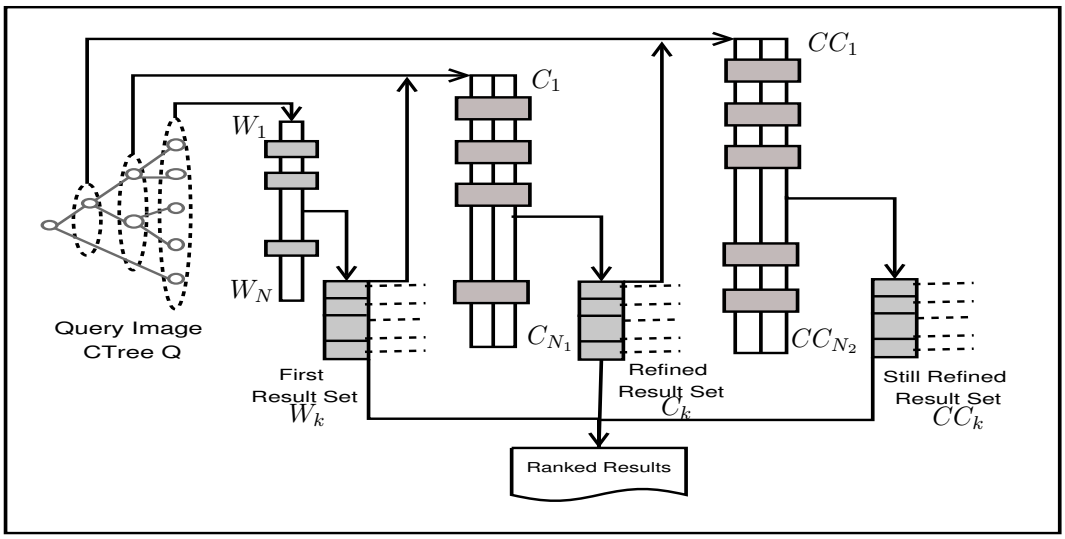


Fig. 3. The figure above depicts a schematic of the CTree indexing process as described in III. For the first level the result set arises from the words in the lowest level of CTree Q. For each new level, words are the collocations of units in previous levels. The refined set is formed from the collocations of that level in Q and the retrieved units from previous level. Finally the results are ranked.

happens with the same expense for any number of images in the dataset. This is essential for the scalability aspect of the system. To efficiently index a dataset of graphical structures D, there are two possibilities. One possibility arises from the concept of best describing feature vector for a class of graphs. The assumption for this approach is that, from the pairwise matching scores within the dataset, it is possible to distinctly cluster the graphs into separate classes. For each cluster, there is a possibility to extract a feature vector that best represents the cluster. The requirement of this feature vector is that it should be a precise representative of the set of graphs and it should be discriminative enough to discriminate between two clusters of graphs. The feature vector extracted from the query image-graph will be matched to the feature vectors of the clusters. This method will extract the best matching cluster that corresponds to the query image.

The field of spectral graph theory offers possible solutions to extract representative feature vector for a cluster of graphs. Based on the method described for 3D object recognition in [16] a possible approach is to extract spectral vectors for a cluster of similar graphs. In this method, the spectral vector is the representative feature vector. For a query graph that is isomorphic with any of the clusters will have the same spectral vector. The advantage with this approach is that it is elegant and will provide an efficient representation to index well-defined clusters. However there are some drawbacks which make the approach implausible for large scale retrieval. One major complaint in the spectral approaches is that they are too rigid to be used in situations where there are inexact similarity metrics in play. In cases where the cluster of graphs are not exactly isomorphic to each other(and thus contain different number of nodes), the permutation matrix cannot be built as is. To do so, the graphs have to be padded with zeros to normalize the size to the cluster. Doing so, the spectral vector will not be the best descriptive representation. Another complaint is that the representation is rigid and will not support the retrieval of partly similar graphs, since it will be indexing only well defined clusters.

The second approach is motivated from the bag of words based retrieval [17] and can be used to index the collocation-tree. The bag of visual words approach forms distinct visual-codewords from the dataset by clustering the sift descriptors into a fixed number of bins. The C-tree can be built from the visual words that are detected in the image and their spatial collocation information. In the classic bag of words based retrieval, the images are indexed with the visual-words. Given a query image with a set of visual words, all the images that are indexed by this set of words are retrieved and the results are ranked by performing a logical-AND operation. The collocation tree, has the set of visual words present in the image at the lowest level. For each higher level, the collocations form something similar to bigrams of the words in the lower level. Except here in this case only one bigram will be chosen for each word. The set of existing bigrams for the dataset will be greater in number than the words, yet finite. Thus the solution is to index the each level of collocations separately, and refine the retrieval result hierarchically over levels.

To do so for the first level, the documents will be indexed by the total number of visual words in the dataset. Let the set of words be $W = \{w_1, \ldots, w_p\}$ and the documents in the dataset be $D = \{d_1, \ldots, d_q\}$. For the first level in the index, each word w_k will have a set of documents D_{w_k} indexed. For the second level, the dataset is indexed again with the visual-word-collocations in the dataset. At the second level each entry will have significantly lesser number of documents than at the first level since the number of collocations are higher than the set of words. This process can be continued to approximately

half of the tree height. When the level of the collocations increase after a level, most index entries will have only one document indexed. At this point the indexing hierarchy can be truncated.

While retrieval, when a query graph is given, the set of documents that have the maximum number of visual words can be retrieved by the standard retrieval process of performing a logical AND between result sets for each word. After a set of potential candidates are extracted by this method, for the next level, the subset of collocations that correspond to the candidate words in the first level are extracted and only the documents that are indexed with the reduced subset of entries are used for comparison. This process continues till the last level. The major advantage with this method of indexing is that there will be an increase in the precision of retrieved results, without actually having to match the query CTrees with the entire dataset. With the method of hierarchical indexing only a small part of the index will be used at each level to refine the retrieved set.

IV. Discussion

The complexity for matching CTrees is considerably lower than for matching NN-graphs since the search space will be very directed. Where the NN-graph matching has to solve the quadratic assignment problem for a node pair and all its neighbors at each step, the CTree has to solve it only for a pair of node pairs. This aspect will greatly reduce the overall cost of matching. The complexity will be approximately $O(|V_1 \times V_2| log(|V_1 \times V_2|))$. Thus for a pair of graphs with, say, 30 nodes each, NN-graphs approach needs to perform a maximum of 810000 operations where as the CTree will need perform a maximum of only 2670 operations. The results will be discussed in detail in the next section. The NN-graph representation tries to encode a larger set of relationships consistently for a proper value of τ. The problem with this approach, as discussed above, is that there is lot of redundancy in this scheme in terms of the number of relationships modelled. Though the CTree aims at improving it by only modelling the right set of relationships, there is a small disadvantage in the latter scheme. One problem is that the collocation linkage process imposes a sort of rigidity on the nodes that are grouped together. If we think of the collocation linkage process as bigrams, the bigrams that are captured are controlled by the collocation process. This aspect will affect the matching process wherein an error in the match at the lowest level will be carried forward to higher levels. This problem can be corrected by employing a priority based linkage scheme where the best neighboring node to be linked to a node will be found out by a weighted priority that is calculated by the most linked neighbors from the rest of the dataset.

V. Results

For the experiments the dataset used is a subset of the University of Kentucky's ukbench database. We have used 400 images (100 classes x 4 instances) to run the matching experiments. The dataset contains images of scenes taken with small projective distortions within them. The dataset is processed and the MSER detector is run on it and the regions detected are given descriptions with the SIFT 128-dimensional vector. The two approaches explained in the previous section are applied in different settings of sparseness constraints.

A. NN-Graphs

The images chosen for the experiments, subjected to the MSER detector, output around 40 to 600 interest regions. Thus the graphs constructed on these configurations are quite varied in terms of densities and the time taken to match. There are mainly two variables that affect the computational time when it comes to the NN-graphs. The size of the node-set and the sparseness of the graph. To measure the effect of either of these factors on the matching time, graphs are constructed on all images with number of features ranging from 40 to 600 and is matched with graphs built on images of the same class. For each configuration the density was also varied by increasing τ the edge placement threshold distance. In figure 5 and table 2 time taken for NN-Graph matching at different node-set sizes are given. At 200 nodes the matching time is 5 seconds, but at 500 nodes, the time taken goes close to 2.5 minutes. The table shows the time taken for different values of the threshold τ it can be seen that at $\tau = 150$ and a nodeset size of 500 the matching takes close to 20 minutes.

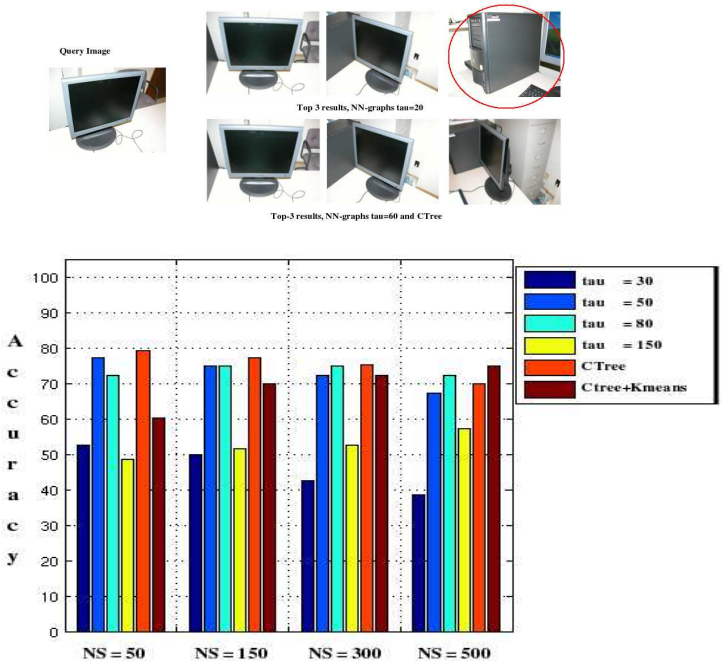


Fig. 4. The left figure shows top-3 retrieval results with a query image. The first set shows results with NN-graphs with a low τ value which retrieves a false result due to a considerable overlap with the descriptor set. The second set of results are the same as retrieved by NN-graph with τ = 50 and by the CTree approach. Bar graph showing retrieval accuracy at different node-set sizes for different values of sparseness of the NN-graph and CTree with and without initial K-Means

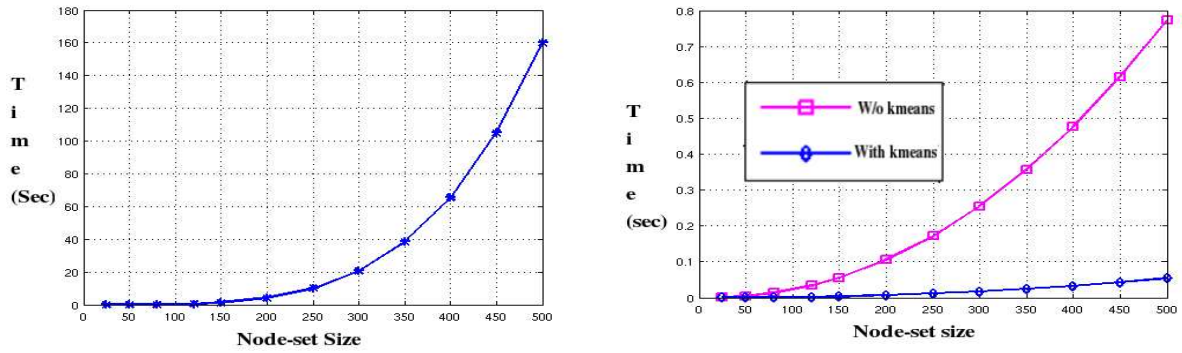


Fig. 5. Figure above the first plot shows the computational time taken for different nodeset sizes with the NN-graph approach, the second plot shows time taken without and with initial clustering in the CTree approach. Notice NN-graph takes 160 seconds at 500 nodes whereas CTree takes close to a second, see V-D

B. Collocation Trees

In similar settings as described for the NN-graphs, the collocation trees are built on the MSER detected regions. For the experiments we have used two different configurations of CTrees. The first configuration is to link all the co-occurrences right from the lowest resolution. Thus the height of the tree will be $\log(N)$ if the size of the nodeset is N. One other experiment that has been performed is when the initial nodeset is clustered using a K-means clustering algorithm with the clustersize set such that approximately a preset number of nodes are clustered into each bin. Now the whole cluster can be used as a patch in situations where there is a bottleneck in terms of computational time owing to the very large number of nodes. In such cases the height of the tree will be $log(C)$ where C is the number of clusters that are formed by K-means. In the plot shown in Fig 5 the time taken for different sizes of the nodeset are given. The two curves show the times with and without clustering the initial node configurations. Table 2 shows the comparison between the time taken for the matching with different edge densities for the NN graphs and the two cases where the initial set of the CTree is linked with and without initial clustering. Observe that matching a pair of CTrees take less than a second for 500 nodes where it takes close to 160 seconds for matching NN-Graphs

C. Indexing

The indexing method described in Section III and as depicted in Fig.III has been employed to index the CTrees constructed from the dataset. The total number of MSER interest regions and and so the SIFT-descriptors in the dataset are around 26450. These descriptors are clustered using kmeans into 200 bins, which are used as visual words. Thus while indexing the total number of visual words i.e., 200 will be the number of entries in the first level of the index. The second level of the index contained about 800 entries which depict the total set of collocations of the words(first order). The second level contained about 1300 entries. The hierarchical index was constructed till 2 levels and has been stopped there since about 80% of the entries contained 1 or 2 collocations indexed. Proceeding any further would result in only one collocation indexed with every entry for almost all of the entries. The time taken to retrieve the set of results for the first level in the index was on the order of 0.2 seconds which is essentially the computational time taken for the classical bag of words based retrieval. The total time taken to traverse all the levels in the index is approximately 2.5 secs. This is compared with the naive matching based retrieval where the query CTree is compared with all the CTrees in the dataset and the results are ranked according to the matching scores. This process took an average of 13 secs. However, it is to be noted that as the dataset size increases, retrieval by traversing a hierarchical index would yield much faster results as the number of operations will not increase directly with the number of Ctrees to compare, which is true in the naive matching based retrieval.

D. Analysis

The experiments have been implemented in Matlab and run on a 1.8GHz machine on 64-Bit platform. For retrieval performance, the top-5 images have been retrieved and the precision at 5 images is calculated. The various settings at which this is tested involved changing the node-set sizes and also the τ value which regulates the sparseness. For lower τ values, edges will be placed between very few pairs. Thus the geometry wont be encoded properly, so is the case for larger τ values. For CTrees at large node-set sizes, the precision is higher when there is initial clustering. For smaller node-set sizes initial clustering reduces the precision as the level of detail at lower resolutions is lost. For every nodeset size, CTrees show similar or better precision than NN-graphs.This is also reflected in the search results shown in the figure. The computational complexities of matching the structures are shown in Fig 5 and Table 2

For NN-graphs as noted earlier, when size increases above 200 nodes the time goes above 5 seconds to match two graphs (see fig 5). At 500 nodes it takes about 2.5 minutes. This presents a scalability issue. However in most cases with large nodesets, most of the nodes are generated by background clut-

	$\tau = 5$	$\tau = 20$	$\tau = 50$	$\tau = 80$	$\tau = 150$	CTree	CTree+K-means
$NS = 40$	$< 10^{-4}$	0.0001	0.002	0.15	0.9	0.0049	0.009
$NS = 80$	$< 10^{-4}$	0.02	0.09	1.12	4.14	0.0139	0.0022
$NS = 150$	0.0001	0.15	0.53	3.6	8.2	0.056	0.0071
$NS = 200$	0.0015	1.263	3.29	9.256	26.5	0.171	0.0117
$NS = 300$	0.0027	3.231	20.736	46.29	102	0.356	0.0246
$NS = 400$	0.0059	11.29	65.53	180	416	0.476	0.0429
$NS = 500$	0.009	25	160	420	1209	0.772	0.0546

TABLE II
TABLE SHOWING TIMES IN SECONDS TAKEN FOR MATCHING NN-GRAPHS AND CTREES AT DIFFERENT SETTINGS OF τ AND NODESET SIZES NS

ter. MSER is a very robust detector and unlike other detectors would return very repeatable results. In most practical cases the node-set size would be at a maximum of 200 nodes. Thus NN-Graph approach is not very impractical. When there is very high background clutter like fine detailed patterns, the detector outputs most regions in that area and such areas are mostly useless background clutter. Thus if a method were there to discount these areas beforehand then NN-graph approach can be used to good advantage

The second plot in figure 5 shows the time taken for various node-sets using the CTree approach. The two curves show the times without and with initial clustering by K-means. Initial clustering implies that the nodeset is clustered by K-means to form small patches with some nodes. The number of clusters were one-tenths the size of the nodeset or ten nodes whichever is high. This constraint of a maximum of 10 nodes is laid since for very large nodesets due to clutter, clustering into one tenth clusters will prove useless and affect precision of match. When the initial clustersize is 10 approximately the bottom two levels of the tree are removed thus resulting in a faster access. Even without initial clustering, the time taken is less than 1 second even for 500 nodes. This makes the CTree method better suitable for large-scale operations for large node-sets.

VI. CONCLUSIONS AND FUTURE WORK

In this work we have discussed a representation-scheme for efficiently modelling parts based representations and matching them. Though the matching process will be sub-optimal it can be used to large-scale applications without concern to the bottleneck of computational time. For future work, the next logical extension is to index a large database of graphs and employ this technique in large-scale image retrieval.

REFERENCES

[1] F. Chevalier, J. P. Domenger, J. Benois-Pineau, and M. Delest, "Retrieval of objects in video by similarity based on graph matching," *Pattern Recogn. Lett.*, vol. 28, no. 8, pp. 939–949, 2007.

[2] A. Hlaoui and S. Wang, "A new algorithm for graph matching with application to content-based image retrieval," in *SSPR/SPR*, 2002, pp. 291–300.

[3] R. A. Baeza-Yates and G. Valiente, "An image similarity measure based on graph matching," in *SPIRE*, 2000, pp. 28–38.

[4] J. Yuan, Y. Wu, and M. Yang, "Discovery of collocation patterns: from visual words to visual phrases," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1–8.

[5] H. Bunke, P. Foggia, C. Guidobaldi, C. Sansone, and M. Vento, "A comparison of algorithms for maximum common subgraph on randomly connected graphs," in *Proceedings of the Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition*. London, UK: Springer-Verlag, 2002, pp. 123–132.

[6] M. Neuhaus and H. Bunke, "A probabilistic approach to learning costs for graph edit distance," in *ICPR '04: Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 3*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 389–393.

[7] Pierre-Antoine and C. Christine, "Measuring the similarity of labeled graphs," in *Proc. of International Conference on Case-Based Reasoning*, 2003, pp. 80–95.

[8] J. Sivic and A. Zisserman., "Video google: A text retrieval approach to object matching in videos," in *Proc. of the International Conference on Computer Vision*, October 2003, pp. II:1470–1477.

[9] 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(10), pp. 1367–1372, 2004.

[10] S. Gold and A. Rangarajan, "A graduated assignment algorithm for graph matching," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 18, no. 4, pp. 377–388, 1996.

[11] T. Bozkaya and M. Ozsoyoglu, "Indexing large metric spaces for similarity search queries," *ACM Trans. Database Syst.*, vol. 24, no. 3, pp. 361–404, 1999.

[12] S. Berretti, G. D'Amico, and A. D. Bimbo, "Shape representation by spatial partitioning for content based retrieval applications," in *ICME*, 2004, pp. 791–794.

[13] K. Mikolajczyk and C. Schmid, "Scale and Affine Invariant Interest Point Detectors," *International Journal of Computer Vision*, vol. 60, no. 1, pp. 63–86, 2004.

[14] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proc. of the International Conference on Computer Vision ICCV, Corfu*, 1999, pp. 1150–1157.

[15] D. Conte, P. Foggia, C. Sansone, and M. Vento, "Thirty years of graph matching in pattern recognition," in *Proc. of the Intl. Jnl. of Pattern Reconition and Artificial Intelligence IJPRAI*, 2004, pp. 265–298.

[16] R. C. Wilson, E. R. Hancock, and B. Luo, "Pattern Vectors from Algebraic Graph Theory," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 27(7), pp. 1112–1124, 2005.

[17] J. Sivic, F. Schaffalitzky, and A. Zisserman., "Efficient object retrieval from videos," in *Proc. of the 12th European Signal Processing Conference EUSIPCO 04, Vienna, Austria,*, September 2004.