Image Mining Using Inexact Maximal Common Subgraph of Multiple ARGs

Hui Jiang and Chong-Wah Ngo
Department of Computer Science, City University of Hong Kong
E-mail: {jiangh, cwngo}@cs.cityu.edu.hk

Abstract

In this paper we present a novel approach for mining inexact spatial patterns from multiple images. We propose an inexact maximal common subgraph algorithm for discovering similar common subgraph from multiple graphs. We employ attributed relational graph (ARG) for data representation. The subgraph found by our algorithm is supposed to represent the common pattern of the input images. Unlike previous approaches, our algorithm can work effectively and efficiently for more than two input images.

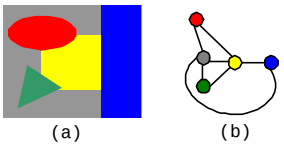

Figure 1. A segmented image and the corresponding ARG representation. (a) A segmented image consists of several image segments in different colors. (b) The ARG representation of (a), each node of the ARG represents a segment of the image, and the attribute of that node represents the color of that segment. There's an edge connecting two nodes if and only if the corresponding two segments are adjacent to each other in the image.

1. Introduction

For years, people want to mine useful information from visual data such as images and videos. Pattern discovery is one of the most important techniques for doing this. In practice, patterns often contain structured information, i.e., a pattern often consists of several basic primitives which exist interdependently. Extracting this kind of structured information from real samples is an interesting and challenging problem. In this paper, we present our work on automatic inexact spatial pattern discovery from multiple images.

A spatial pattern consists of pattern primitives and pattern relations. Pattern primitives represent non-spatial properties, such as color, size, etc. Pattern relations represent spatial properties, such as relative locations, etc.

We choose attributed relational graph (ARG) for data representation [4]. It is a general graphic representation. It can encode all kinds of structured information. An ARG is a graph with attributes (also called labels or weights in the graph theory) on its nodes and/or edges. With the help of the inherent powerful ability of graphs for describing data, ARG can encode structured patterns with both non-spatial and spatial information simultaneously. The attributes on its nodes can encode non-spatial properties of the patterns, while the attributes on its edges can encode spatial properties of the patterns. An example of ARG is in Figure 1.

We can convert the raw images to ARG representation by performing a segmentation algorithm on them, denoting each segment with a node of the ARG, and using the edges of the ARG to reflect the adjacent relations among these segments. After doing this, we will get several ARGs corresponding to the raw sample images. The most important issue is that the similar common patterns in these images are converted to similar common subgraphs in these ARGs. So if we can find the maximal common subgraph (MCS) of these ARGs, actually we have discovered the most often appeared pattern of the sample images. The whole process is illustrated in Figure 4 - 8 in the later part of this paper.

There are many approaches for mining patterns from multiple samples [2, 5, 6]. Some of these approaches deal with deterministic properties of patterns. Some of them deal with only non-spatial patterns or with limited structural information of spatial patterns. The approach most close to us is, Hong and Huang [2] use Expectation Maximization (EM) algorithm to mining spatial patterns from multiple ARGs. Comparing to the EM algorithm, our approach of using MCS is more practical and efficient.

In the rest of the paper, we will first present the methodology for automatically extracting inexact spatial patterns in Section 2. We then present the algorithm to extract the inexact maximal common subgraph in Section 3. The time and space complexities of our algorithm are analyzed in Section

4. Section 5 presents experimental results. The paper is concluded with summary in Section 6.

2. Overview of Our Method

Given several images, first we use attributed relational graph (ARG) to represent them. In this paper, we employ an image segmentation algorithm [3] on the input sample images and encode each image segment with a node of the ARG, just as in Figure 1.

After we use ARGs to represent the input images, the next step is to discover the common patterns embedded in these ARGs. We propose an inexact maximal common subgraph (IMCS) algorithm to achieve this goal. This algorithm has two differences with algorithms to the traditional maximal common subgraph (MCS) problem [1, 7–9]: (a) this algorithm is taking on multiple graphs (maybe more than two), whereas the traditional problem is for just two graphs; (b) this algorithm seeks for inexact maximal common subgraph, whereas the traditional problem is for exact maximal common subgraph. The reason why we employ inexact maximal common subgraph algorithm is obvious, in real applications, inputs always contain errors due to distortion, inaccurate observations, and so on.

Finally, when we get the subgraph, we turn back to the raw images and dig out the image segments which are encoded by the nodes of the subgraph. Those image segments are supposed to be the common spatial patterns we want.

3. Inexact MCS of Multiple ARGs

Before introduce our algorithm, we review some previous works on Maximal Common Subgraph (MCS) problem.

3.1. Previous Works on MCS

Maximal Common Subgraph (MCS) problem is a well-studied problem. Given several graphs $G_1, G_2, \ldots, G_n$, the MCS of them is a graph G^* with maximum number of nodes satisfies: for any graph $G_i (1 \leq i \leq n)$, there's a subgraph of G_i which is isomorphic to G^*.

MCS problem is hard, even with the case of $n = 2$, it is NP-Hard [1]. But since many problems can be modelled as MCS problem, there are many proposed methods for solving MCS [7–9]. Because the NP-Hardness of MCS, these methods are either sub-optimal or exponential time. Most of them are for exact MCS, and all of them are for two graphs.

When dealing with more than two graphs, MCS problem will be much harder. For example, McGregor [8] use a backtrack search algorithms for the MCS problem, and its time complexity is $O(n!)$, where n is the size of the graph. But when it deals with k graphs, the time complexity will be $O((n!)^k)$, which is unacceptable. Another example, Messmer [9] constructs the association graph of the two given graphs first and then detects the maximum clique (MC) of the latter graph. But when this method is generalized to k graphs, the constructed association graph will have n^k nodes, which is too large for ordinary value of n (about 100) and k (about 10).

3.2. Our Algorithm of IMCS

Our algorithm is for solving the Inexact Maximal Common Subgraph (IMCS) Problem. Informally speaking, each node of the input graphs has attributes (also called weights or labels in graph terminology), and what we should do is to find out the maximal common subgraph of these graphs, while if two nodes are mapped to each other in graph isomorphism, their attributes should be close.

To tell which node pairs have close attribute and which have not, we must calculate the "distance" of two nodes. In this paper, the attribute of the nodes is the RGB vector of the mean RGB values of the corresponding image segment, and the distance of two nodes is calculated as the distance of the two RGB vectors in 3D RGB space. We use $D(u, v)$ to denote the distance between nodes u and v.

We choose a distance threshold δ (in our experiments, we fix it at 0.3, where RGB value are real numbers range from 0 to 1), and we think two nodes with distance greater than δ can not be mapped to each other in graph isomorphism.

Given several ARGs, first we calculate all the distances $D(u, v)$, where u is a node of G_i, v is a node of G_j, and $i \neq j$. Then for each node u, we build a node list $L(u)$ for it, which contains all the nodes v satisfies $D(u, v) < \delta$, where u and v belong to different graphs.

Then we employ a backtrack depth first search algorithm to find out the inexact maximal common subgraph. The main flow of this algorithm is in Figure 2, details will be explained later.

In the search, the step for updating the subgraph list $L(G_k)$ for all the graphs G_k except G_i is the most crucial. Suppose in the search algorithm, current graph is G_i, current subgraph of G_i is G^*, and the next step is to add a new node u of G_i into G^*. Suppose for a graph G_k other than G_i, the subgraph list for G_k is $L(G_k)$, which contains all the possible subgraphs of G_k similar to G^*. The updating algorithm is illustrated in Figure 3.

4. Time and Space Analysis

In the worst case, the time and space complexity of our algorithm all exponent functions with respect to the maximal number of nodes of the ARGs, which is common for an NP-Hard problem.

1: **for** all G_i, $(G_i = G_1, G_2, \ldots, G_n)$ **do**
2: **for** all nodes u_{ij} of G_i, $(u_{ij} = u_{i0}, u_{i1}, \ldots, u_{i|G_i|})$ **do**
3: Set current subgraph G_* to a graph with just one node u_{ij}
4: **for** all other graphs G_k, $(k = 1..n, k \neq i)$ **do**
5: Build a subgraph list $L(G_k)$ for G_k, which contains all the possible subgraphs of G_k that similar to G_*. Initialize $L(G_k)$ with empty list.
6: **for** each node v of G_k in $L(u_{ij})$ **do**
7: Add a graph with just one node v to $L(G_k)$
8: **end for**
9: **end for**
10: Carry out a backtrack depth first search, which checks for all the subgraphs of G_i contains u_{ij}, and takes u_{ij} as its smallest label node. During the search, updating the subgraph list $L(G_k)$ for all the graphs G_k other than G_i. Whenever for some graph G_k, $L(G_k)$ is empty, stop deeper search and backtrack. During the search, record the maximal subgraph found so far.
11: **end for**
12: **end for**

Figure 2. Backtrack search for the inexact maximum common subgraph of the ARGs.

1: Create a new subgraph list $L^*(G_k)$ for G_k. Initialize $L^*(G_k)$ with empty list.
2: **for** all nodes v of G_k in $L(u)$ **do**
3: **for** all subgraphs G in $L(G_k)$ **do**
4: **if** v is not in G and adding u to G^* and adding v to G simultaneous doesn't destroy the structure isomorphism between G^* and G **then**
5: Add the graph $G \cap v$ to $L^*(G_k)$
6: **end if**
7: **end for**
8: **end for**
9: Replace $L(G_k)$ with $L^*(G_k)$

Figure 3. Update subgraph list $L(G_k)$ for graph G_k.

But in real applications, worst case analysis does not make much sense with average case. Making an exact time and space complexity analysis for the average case is too difficult. We just make some estimates, from which we can see why our algorithm works efficiently.

Suppose the mean color value of image segments obey to uniform distribution. The probability of two RGB vector with the distance between them less than δ (0.3 in our experiments) is about 0.078. So if we have 10 graphs and each graph has 100 nodes (k=10, n=100, a very common input size for real applications), a node u of graph G_i will has about 100*0.078=7.8 buddy nodes in another graph G_j which is within a distance less than δ of u. In the search algorithm, if size of the current subgraph G_* is s, suppose adding a new node u to G_* will cause adding x subgraphs to another graph G_j. Since the new generated subgraph must obey the structure isomorphism constraint, the value of x is about $\frac{7.8}{2^s}$, which is less than 1 for $s >= 3$. That is, when the current found subgraph has more than 3 nodes, the probability that adding new nodes to the current found subgraphs will cause more new subgraph adding to the possible subgraph list is small. So we can draw out the conclusion that in average case the time and space requirements of our algorithm are fairly small. Experimental results also justified this.

5. Experimental Results

We take real photos for experiment. An example is shown in Figure 4 - 8. The raw images are the 7 "NO STOP" traffic signs in various backgrounds in Figure 4. Those images are segmented (see Figure 5) and represented as ARGs (see Figure 6). The extracted maximal common subgraph are shown in Figure 7. The original image segments corresponding to the nodes in the maximal common subgraph are shown in Figure 8.

In this experiment, the seven ARGs have different number of nodes. The smallest number of nodes is 7, while the biggest number of nodes is 126. In fact, the number of nodes in the ARGs and the number of ARGS do not affect the result and the running time greatly. When we run our algorithm on 20 ARGs with each has greater than 100 nodes, the running time is about 2.5 minutes[1], which is acceptable.

From the figures we can find that, the different occurrences of the pattern are of different shape and different color. Take an example of the "x" shape in the middle of the "NO STOP" traffic sign. The mean RGB values of the "x" in the first three images shown in Figure 4 are (91, 24, 34), (68, 26, 36) and (93, 24, 34) respectively. But our algorithm can deal with these differences well because it search for the inexact maximal common subgraph.

The running time of this experiment is about 20 seconds.

Another example is shown in Figure 9. The input ARGs have about 30 to 100 nodes, the found subgraph has 5 nodes. The running time is about 15 seconds.

6. Summary and Conclusions

We present a method for mining inexact spatial patterns from multiple images. We propose an inexact maximal common subgraph algorithm for extracting the inexact spatial pattern. Differ from the traditional algorithms for MCS problem, our algorithm works for multiple graphs. Without lost of effectiveness and efficiency, it works in an inexact way. Experimental results and time and space complexity analysis are given.

[1] All the experiments in this paper were done with a PC which has a Pentium III 1G CPU and 256M RAM.

Figure 4. Raw images.

Figure 5. The segmentation results of the images in Figure 4.

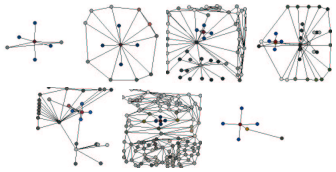

Figure 6. The ARGs of the images in Figure 5.

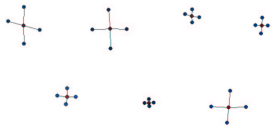

Figure 7. The MCSs found from the ARGs in Figure 6.

Further works will concentrate on introducing more inexactness to our MCS algorithm, such as structure inexactness: to allow the two graphs have a little difference in their structure when they are matching to each other.

Acknowledgments

The work described in this paper was fully supported by RGC Grant CityU 1072/02E (Project No. 9040693).

References

[1] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, New York, 1979.

[2] P. Hong and T. S. Huang, "Inexact Spatial Pattern Mining," *Workshop on Discrete Mathematics and Data Mining*, 2002.

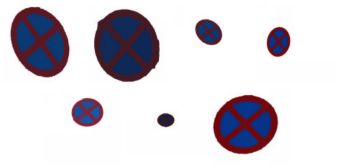

Figure 8. The image segments corresponding to the nodes of the maximal common subgraphs.

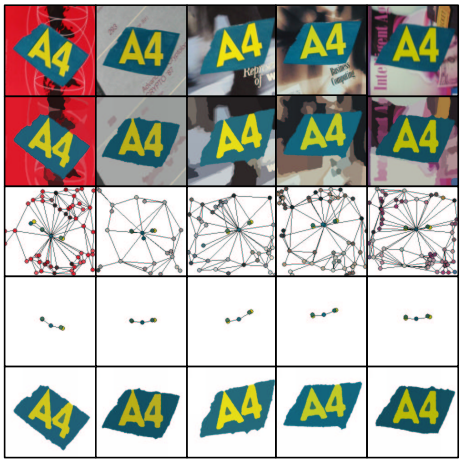

Figure 9. Another experiment result on 5 images of "A4" in various backgrounds.

[3] P. F. Felzenszwalb and D. O. Huttenlocher, "Image Segmentation Using Local Variation," *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 98-104, 1998.

[4] W. H. Tsai, and K. S. Fu, "Error-Correcting Isomorphism of Attributed Relational Graphs for Pattern Analysis," *IEEE Trans. Sys., Man and Cyb.* Vol. 9, 757-768, 1979.

[5] O. Maron and T. Lozano-Prez, "A Framework for Multiple-Instance Learning," *Neural Information Processing Systems* Vol. 10, 1998.

[6] A. L. Ratan, *et al.*, "A Framework for Learning Query Concepts in Image Classification," In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition*, 423-429, 1999.

[7] H. Bunke, P. Foggia, *et al.*, "A Comparison of Algorithms for Maximum Common Subgraph on Randomly Connected Graphs," In *Proc. IAPR Workshop on Structural and Syntactic Pattern Recognition*, 2002.

[8] J. J. McGregor, "Backtrack Search Algorithms and the Maximal Common Subgraph Problem", *Software Practice and Experience*, Vol. 12, 23-34, 1982.

[9] B. T. Messmer, "Efficient Graph Matching Algorithms for Preprocessed Model Graphs," Ph.D. Thesis, Inst. of Comp. Science and Appl. Mathematics, University of Bern, 1996.