UCRL-PROC-204929

LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

Data Structures and Algorithms for Graph Based Remote Sensed Image Content Storage and Retieval

C. W. Grant

June 25, 2004

IEEE International Geoscience and Remote Sensing Symposium, Anchorage, Alaska, September 20-24, 2004

Data Structures and Algorithms for Graph Based Remote Sensed Image Content Storage and Retrieval

Charles W. Grant
Lawrence Livermore National Laboratory
7000 East Ave.
Livermore, CA 94551

***Abstract*- The Image Content Engine (ICE) project at Lawrence Livermore National Laboratory (LLNL) extracts, stores and allows queries of image content on multiple levels. ICE is designed for multiple application domains. The domain explored in this work is aerial and satellite surveillance imagery. The highest level of semantic information used in ICE is graph based. After objects are detected and classified, they are grouped based in their interrelations. The graph representing a locally related set of objects is called a "graphlet". Graphlets are interconnected into a larger graph which covers an entire set of images. Queries based on graph properties are notoriously difficult due the inherent complexity of the graph isomorphism and sub-graph isomorphism problems. ICE exploits limitations in graph and query structure and uses a set of auxiliary data structures to quickly process a useful set of graph based queries. These queries could not be processed using semantically lower level (tile and object based) queries.**

I. Introduction

The Image Content Engine (ICE) is an ongoing project at Lawrence Livermore National Laboratory (LLNL) to produce systems for extracting domain specific semantic information (at multiple levels of abstraction) from large volumes of imagery and evaluating queries using this semantic information. The initial application domain is aerial and satellite surveillance imagery of broad areas. The system will be able to combine imagery information from various observation modes, including optical, hyper-spectral, and synthetic aperture radar.

ICE extracts image information at multiple semantic levels. At the lowest level, tile-based, information is mainly spectral, statistics about pixels in local rectangular regions [1,2]. The next highest level, the region level, involves segmenting the image into sets of contiguous pixels with similar properties [3]. The third level of information is the object level. It is at the object level where tile and region features are used to detect domain specific entities or the image can be directly matched against models of known objects. It is here at the object level where most low level semantic information is introduced. For example a rectangular region classified as a "building" or a “railcar.”

The ICE system is structured as a configurable processing pipeline [4]. Tiles, regions and objects are input to the "semantic representation" stage or since our semantic representation scheme is graph based, it is also called the "graph building" stage.

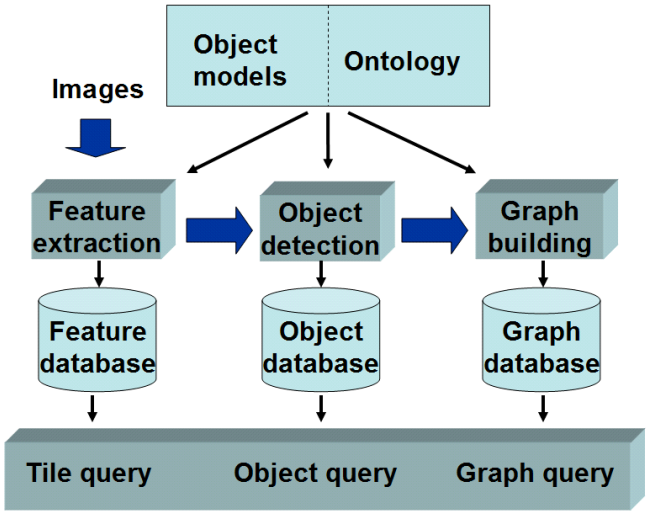

Figure 1, The ICE Pipeline Architecture.

II. Semantic Graph Representation of Content

It was decided early in the project to use a graph based representation to hold the semantic information, a "semantic graph." A graph is simply a data structure consisting of “nodes” and “arcs” or “links” which connect the nodes. In our representation, all objects or collections of objects are represented by nodes. Relations between nodes are represented by links between the corresponding nodes. Most links are directed.

Nodes contain unique identifiers. Nodes are typed, i.e. classified as one of a small number of possibilities such as “building” or “storage tank.” Nodes contain multiple values and the collection of values is called a feature vector. This can be the same as the feature vector generated by the feature extraction and object detection pipelines [4]. Nodes contain spatial information about the object, spectral information and possibly some other known or derived information. Having the nodes be rich in common descriptive information improves the performance of many graph algorithms.

In the current implementation, links are typed and have unique identifiers, but do not carry values. Although one can imagine cases where link values might be useful, this functionality can be simulated by replacing the link with two links and an intermediate node. The intermediate node can be used to hold the “link value”. If the use of this construct is widespread then, for efficiency, the link data

structure can and should be redesigned to incorporate the link values directly into the link data structure.

The graph representation provides an extreme amount of expressive power in describing collections of objects and relations between them. Ignoring performance issues, storing all the information in one graph, formulating queries as graphs, and evaluating the queries by determining all the sub-graphs which are isomorphic to the query is nearly ideal. Unfortunately, the sub-graph isomorphism problem is NP-complete [5]. This means that for some sets of graphs and queries, the sub-graph isomorphism problem can not be solved any faster than exponential time (the time required is an exponential function of the size of the query). So in the general case of arbitrary graphs and arbitrary queries, only trivially small queries can be guaranteed to be executable in reasonable times.

III. Performance Requirements

The restrictions for using very large graphs are severe. Preprocessing the graph is essential but, since the graphs for some applications are going to very large, say millions of nodes and larger, it is clear that there is not much time available to preprocess each element in the graph. We also need to be able to add more nodes and links to the graph at a later time without redoing the preprocessing for the original nodes and links. We also may delete some parts of the graph without having to redo the preprocessing of the remaining parts of the graph. Clearly a preprocessing scheme of quadratic time complexity or worse is not be possible. Something like O(n log n), where n is the size of the graph, is probably about the slowest algorithm that could be used for preprocessing the entire graph. But a linear time algorithm would be preferable.

Executing a query also has very strict performance requirements. Examining every node in the graph, even once, is too slow. All query algorithms must be sub-linear with respect to graph size (this would not be possible without preprocessing). Time complexity for queries should be about O(log n) with respect to graph size or faster and time complexity should be nearly linear, or faster, with respect to query size and the number of matches. So the overall time complexity for queries should be something like O(qr(log q)(log r)(log g)) or faster for reasonably acceptable performance on very large graphs, where q is the query size, r is the number of results and g is the size of the graph. But low degree polynomials in r and q might be acceptable (especially for small r and q).

IV. Graph Restrictions

Because of the performance requirement of very large graphs, it is natural to look for ways to restrict sets of graphs and queries in some way so that reasonable performance can be guaranteed but so the graph and query still have a sufficient amount of expressive power to describe the semantics required for the problem domain.

Some aspects of the surveillance problem domain, and probably imagery in general, are very useful in restricting the types of graph structures to those that are more easily queried. Objects in the images tend to be related only to objects which are spatially very local. This limits the degrees of the nodes in the graph. Groups of objects also tend to be separated from other groups in easily detectable ways, and low level objects of interest are mostly parts of higher level objects which are structured hierarchically. Thus we can expect the graphs to be fairly isolated sub-graphs and the sub-graphs will have mostly planar interrelations which correspond to nearby spatial relations. We call these isolated sub-graphs "graphlets".

Both the graph isomorphism problem and the sub-graph isomorphism problem are much easier if nodes and links are very distinctive [6,7]. The worst case performance occurs when all nodes and links “look alike” so they must be repeatedly examined in all combinations. The design decision to use information rich nodes with lots of values distinguishing information values within each node is a significant performance help by both distinguishing nodes and by reducing the total number of nodes and links in the graph. An alternative representation is to use, for each value in each node, a separate link (“has area of” for example) and a separate node to hold the value. This alternative “information poor” representation scheme is simple, but wasteful of space and much more difficult to search.

V. Graphlets

Within each graphlet we have a hierarchical structure with the lowest level of objects (those detected in previous pipeline stages) at the bottom of the hierarchy. These low level objects will be connected to other low level objects (at the same level in the hierarchy) based on their spatial relations or other known relations. This level can be quite complex.

The next level up in the hierarchy, the "group" level, groups the lower level objects into overlapping sets which share some mutual relation, for example, a set of mutually parallel objects, or a set of connected (touching) objects. The graph node representing the set of mutually related objects contains summary information about the set which is used to satisfy queries without examining the lower level nodes whenever possible. This summary information includes the number of objects interrelated, the total area of the interrelated objects, the area of the largest interrelated object, the maximum length of any of the interrelated objects, etc.

The next highest level node is the graphlet node. The graphlet node contains summary information about the group nodes (number of parallel groups, total number of parallel objects, maximum number of objects in a parallel group, etc.) as well as combined summary information about all the low level object nodes. If the number of types of groups is small and the number of types of low level objects is small, then summary information about each type of group and each type of object can be stored with the graphlet node. Initially we will use the group types connected, near, sequential, linear, parallel and unrelated and the object types: building (rectangle), storage tank (ellipse) and road.

As the object detection pipeline stage generates objects they are input to the graph building pipeline stage (Fig. 1). Objects are first sorted spatially and nearby pairs of

objects are examined for known spatial relations. There are straightforward geometric tests for connectedness, closeness, parallelism, co-linearity, containment, and sequential placement on a mildly curving path, but all require that one or more adjustable tolerance parameters be reasonably set.

Sets of mutually similarly related objects are linked to group nodes (mutually parallel objects, for example). Objects are considered "mutually related" if a path exists between the objects consisting of only links of the same type. A direct link between each pair of objects in the set is not required. Objects can belong to multiple groups by way of different types of relations (near some objects and parallel to a subset of them, for example). Objects which do not belong to any other groups, but which are still close enough spatially to be considered part of the graphlet, are placed in a default "unrelated" group.

The group nodes are then linked to a single graphlet node and the summary information in each group node is merged into the graphlet node. Any links between objects in different graphlets (because of non-spatial relations) are "promoted" up to the graphlet level by creating corresponding links between the corresponding graphlet nodes.

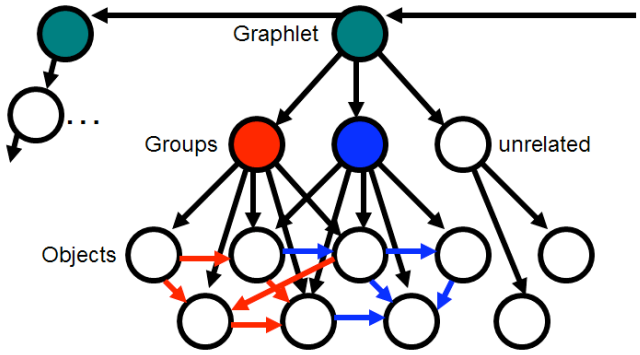

Figure 2. A Simple Graphlet.

The overall structure of a simplified graphlet is shown in figure 2. Graphlet nodes are linked into a global structure at the top level. The graphlet nodes, group nodes and object nodes and links between graphlet nodes and group nodes and between group nodes and object nodes form the directed acyclic graph, DAG, part of the graphlet. At the lowest level in the graphlet the object nodes and their basic interrelations are represented in a nearly planar and usually small sub-graph.

The key to preprocessing for sub-linear performance is then to index the graphlet nodes, based on the graphlet summary information, using conventional database indexing techniques [8]. This allows us to cull most graphlets without examining any of their nodes. In addition the summary information available at each group node is saved and is used to further cull searches within the graphlet. Ideally one would like to have an index pre-calculated for every combination of feature values that could appear in a query, but this is not practical. The graphlets can be indexed using a single multi-dimensional data structure, such as a k-d tree or BSP tree [9]. The advantage of these data structures is that they are flexible enough to allow queries using any features, but they are usually not as efficient as having an index pre-calculated for the specific set of features used in the query. A high performance system would likely include several indices pre-calculated for what are anticipated to be the most likely combinations of features in queries and a general multi-dimensional index to handle everything else.

VI. Querying the Graph

A graph query is essentially a sub-graph isomorphism problem. The "query template" represents the sub-graph. The matching process is more complex than a pure sub-graph isomorphism search because the query template can describe a more complex node matching than a simple type or value match. Matching nodes with a range values is a simple example of more complex search. The query template can also contain items which are not allowed to be present in graphlets that match. For example, more complex query might be a storage tank which is not near any building with an area greater than some specified value. The search could be more properly described as a generalized sub-graph isomorphism algorithm.

Each query template will contain at least one graphlet node template that specifies what ranges of summary values constitute a potential match. The query process will first retrieve all graphlets which match the graphlet template's specified ranges. This uses a pre-computed index of the graphlet nodes and involves only accessing a very small fraction of the total number of graphlets.

Then for each graphlet the other nodes of the graphlet will be examined to see if they match the remaining part of the graph template. First the group nodes will be matched to the range information in the group node templates of the query template. In this way, most non-matching graphlets will be filtered out by using the summary information well before any low level nodes and their potentially complex interrelations are examined. The group nodes are examined in a pre-calculated order which is intended to yield the quickest culling of non-matching nodes. If the group nodes satisfy the range requirements, then it is very likely that this is a match. At this point an explicit search is performed to find a suitable isomorphism.

Depending on the particular query specification, the search within that graphlet can terminate after finding the first match, or it can continue and all isomorphisms, including overlapping ones, will be returned. This is important because the search within a single graphlet can take time that is exponential in the size of the query. The search within a single graphlet can also potentially find an exponential number of matches if overlapping matches are allowed. These limitations do not entirely avoid the worst case exponential behavior of sub-graph isomorphism. It is still possible to spend an amount of time that is exponential in the query size while searching in a single graphlet before finding a match or eventually failing to find a match.

VII. Scalability

These data structures and algorithms are designed for very large data sets. It is expected that we will use the

techniques at different scales, from single processors to large clusters depending on the size of the problems (mainly number of images). The only significant bottlenecks identified for parallel implementations are the same bottlenecks all parallel data base implementations face. We expect to be able to avoid these scalability bottlenecks by using an existing parallel data base system, or using the same techniques used in existing parallel data base systems in a custom system.

Most of the processing in the graph generation pipeline stage can be classified as "embarrassingly parallel." Individual images can be processed completely independently, except for indexing and storage of the final graphlets. We expect very good parallel performance for graph generation.

Parallel query execution can also be classified as "embarrassingly parallel." Since each entire graphlet will be stored on a single machine, the per-graphlet processing will be able to proceed completely independently on each computer.

VIII. Status and Conclusions

A system for the efficient generation, storage and querying of graph based image content has been presented. The system has been implemented to work entirely in-RAM as a first step. An out-of-RAM implementation is in progress. As of this writing the system has not been tested on large numbers of large images. Such testing is scheduled before the presentation of this paper, so an oral progress report will be made at that time.

Acknowledgment

This work was performed under the auspices of the U.S. Department of Energy by University of California, Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.

References

[1] S. Sengupta, A. Lopez, J. Brase, D. Paglieroni, "Tile-based railroad detection in multi-source images", Proceedings of IGARSS 2004, September 2004.

[2] S. Sengupta, A. Lopez, D. Poland, "Class-label statistics: a basis for fusing information from multi-spectral imagery with an application to unsupervised detection of human settlement", Proceedings of IGARSS 2004, September 2004.

[3] D. Paglieroni, "Convergent coarseness regulation for segmented images", Proceedings of IGARSS 2004, September 2004.

[4] G. Weinert, J. Brase, D. Paglieroni, "Computer-aided content-based cueing of remotely sensed images with the image content engine", Proceedins of IGARSS 2004, September 2004.

[5] M. R. Garey, D. S. Johnson, *Computers and intractability a guide to the theory of NP-completeness*, W. H. Freeman and Company, 1979, p. 202

[6] B. D. McKay, Practical graph isomorphism, 10th. Manitoba Conference on Numerical Mathematics and Computing (Winnipeg, 1980); Congressus Numerantium, 30 (1981) 45-87.

[7] B. D. McKay, nauty User's Guide (version 1.5), Tech. Rpt. TR-CS-90-02, Dept. Computer Science, Austral. Nat. Univ. (1990).

[8] C. J. Date, *An introduction to database systems*, 7th ed, Addison Wesley Longman, 1999

[9] F. P. Preparata, M. I. Shamos, *Computational geometry: an introduction (texts and monographs in computer science),* Springer Verlag, 1991