Pattern Recognition Letters 2 (1983) 83–88
North-Holland December 1983

A Hough transform technique for subgraph isomorphism

Simon KASIF
Les KITCHEN
Azriel ROSENFELD
Center for Automation Research, University of Maryland, College Park, MD 20742, USA

Received 24 May 1983

Abstract: The Generalized Hough Transform is an established technique for geometric shape matching. An analogous technique is presented here for approximate solution of subgraph isomorphism problems. Its chief advantages are that it can also be used to find partial isomorphisms and that it can be readily implemented in parallel on a network of simple processors.

Key words: Hough transforms, matching, subgraph isomorphism.

1. Introduction

A crucial problem in pattern recognition is the detection of a specified pattern in input data. Many formalisms for this problem exist; for a survey relevant to computer vision see Ballard and Brown (1982). One quite general formulation is as a subgraph isomorphism problem, for which a number of solution methods have been proposed. A more specialized pattern detection problem is that of geometric shape detection in images. Again, there are numerous methods for solving this problem; one in particular is Ballard's Generalized Hough Transform (Ballard, 1981), which is in many respects an improvement over simple template matching.

We present here a novel approach to subgraph isomorphism inspired by the Generalized Hough Transform. It is of interest because it illustrates the close relationship between the shape matching and subgraph isomorphism problems, and because it is amenable to parallel implementation on a suitable network of rather simple processing elements.

The support of the National Science Foundation under Grant MCS-79-23422 is gratefully acknowledged.

2. Subgraph isomorphism for labelled graphs

We regard a labelled graph as a quadruple (N, A, λ, ϱ), where N is a finite set of *nodes*, $A \subsetneq N \times N$ is a set of ordered pairs giving the directed *arcs* of the graph, λ is a mapping that associates a *node label* with every node in N, and similarly ϱ is a mapping that associates an *arc label with every arc in A*. A *subgraph isomorphism*, or *monomorphism*, from one graph

$$G_1 = (N_1, A_1, \lambda_1, \varrho_1)$$

to another graph

$$G_2 = (N_2, A_2, \lambda_2, \varrho_2)$$

is an injective mapping $f : N_1 \to N_2$ such that

$$\lambda_2(f(x)) = \lambda_1(x)$$

for all $x \in N_1$ and

$$\varrho_2(f(x), f(y)) = \varrho_1(x, y)$$

for all $(x, y) \in A_1$.

There are many techniques for finding such monomorphisms; most are based on some sort of backtrack search, but with important refinements that cut down on the combinatorics of the matching.

3. The Generalized Hough Transform

There are many variations possible on the Generalized Hough Transform; for illustrative purposes we describe how it can be used for detecting a shape (represented as a discrete collection of boundary points, each with an associated boundary direction) in an edge image (obtained by applying some edge detector to a digital image). We assume that the orientation and size of the shape are fixed; only its location is unknown.

We arbitrarily choose some point to be the origin of a coordinate frame for the shape. This origin is called the *reference point* for the shape. It need not be one of the boundary points, nor an interior point of the shape, but for reasons of computational accuracy it should be close to the shape's centroid. For every boundary point of the shape we compute the vector displacement from that point to the reference point, and store this displacement in a table, indexed by the local boundary direction at the boundary point. Ballard calls this table the *R-table*. Having built the R-table, we set up a two-dimensional array of accumulators with the same coordinates as the given edge image. For every point in the image we consult the R-table entries for its edge direction. These entries give displacements to possible positions of the reference point of the sought shape. We increment the accumulators for these possible positions. If there is indeed an instance of the shape in the image, then the accumulator corresponding to its position will eventually have been incremented once for every boundary point. Other accumulators may have been incremented spuriously, but not to the same extent.

In practice, the situation will not be quite as clear as this, because of noise, digitization errors, occlusion, and other factors. But instances of the shape can generally be found by locating significant maxima in the array of accumulators.

4. The Relational Hough Transform

In the abstract, the Generalized Hough Transform for geometric shape matching has the following feaures: For the shape to be matched, an arbitrary reference point is chosen. For every point on the shape boundary its relationship to the reference point is computed, and stored in a table indexed by intrinsic properties of the boundary point. For every edge point in the target image we consult this table and increment the accumulators for all points that stand in the proper relationship to the edge point.

This procedure has a ready analogue for finding a monomorphism between two labelled graphs

$$G_1 = (N_1, A_1, \lambda_1, \varrho_1)$$

and

$$G_2 = (N_2, A_2, \lambda_2, \varrho_2).$$

We choose a node $r \in N_1$ to be the reference point in G_1. For every other node $x \in N_1$ we compute some of its relationships to r. For our purposes, a relationship between x and r is captured by the sequence of arc and node labels that appears on some (undirected) path from x to r. We call such a sequence a *relational sequence* from x to r. In a relational sequence the arc labels need to be specially marked to indicate their sense. For example, consider the graph in Figure 1, with nodes 1, 2, 3 and 4, node labels L and M, and arc labels R and S. Relational sequences from node 4 to node 1 are MSL and $\mathrm{M\bar{R}L\bar{S}L}$, with the overbars indicating when the path runs against the direction of an arc.

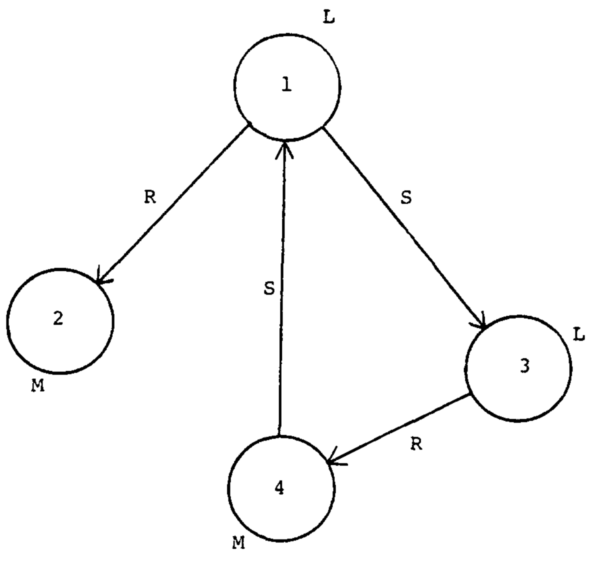

Fig. 1. A graph with arc and node labels.

The relational sequences to the reference point from every other node are stored in a table. (In our current implementation only a single relational sequence is computed and stored for each node.) This table thus contains a description of the structure of G_1 relative to the reference point. The description may possibly not specify G_1 completely, but can capture much of its structure.

Having constructed this table of relational sequences, we associate an accumulator with every node of G_2. Then for every node y in G_2 we select from the table those relational sequences that begin with the same label as y bears. For each of these relational sequences we follow all those paths from y that match the sequence, label by label, respecting the directions of arcs. Any node in G_2 that lies at the end of such a path stands in a proper relationship to y to be the image of the reference point under some monomorphism, so its accumulator is incremented.

Clearly, if there is a monomorphism from G_1 to G_2, then after this process has finished the image of the reference point will have had its accumulator incremented once for every relational sequence in the table. As in the shape matching case, other accumulators may have been incremented spuriously; if the two graphs have sufficient confusing symmetries, these other accumulators may equal that of the image of the reference point. So any node with a large enough sum associated might possibly be the image of the reference point under some monomorphism. Our experiments below with some typical subgraph isomorphism problems indicate that this procedure will usually yield only one or two such possibilities.

5. An example

We illustrate this technique with a graph based on the map of the U.S.A. The nodes of the graph correspond to the 48 contiguous states; the arcs represent the adjacency relations between pairs of

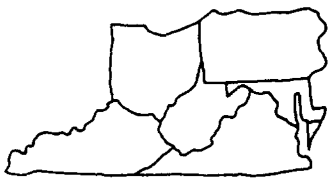
Fig. 2. Map of states in subgraph.

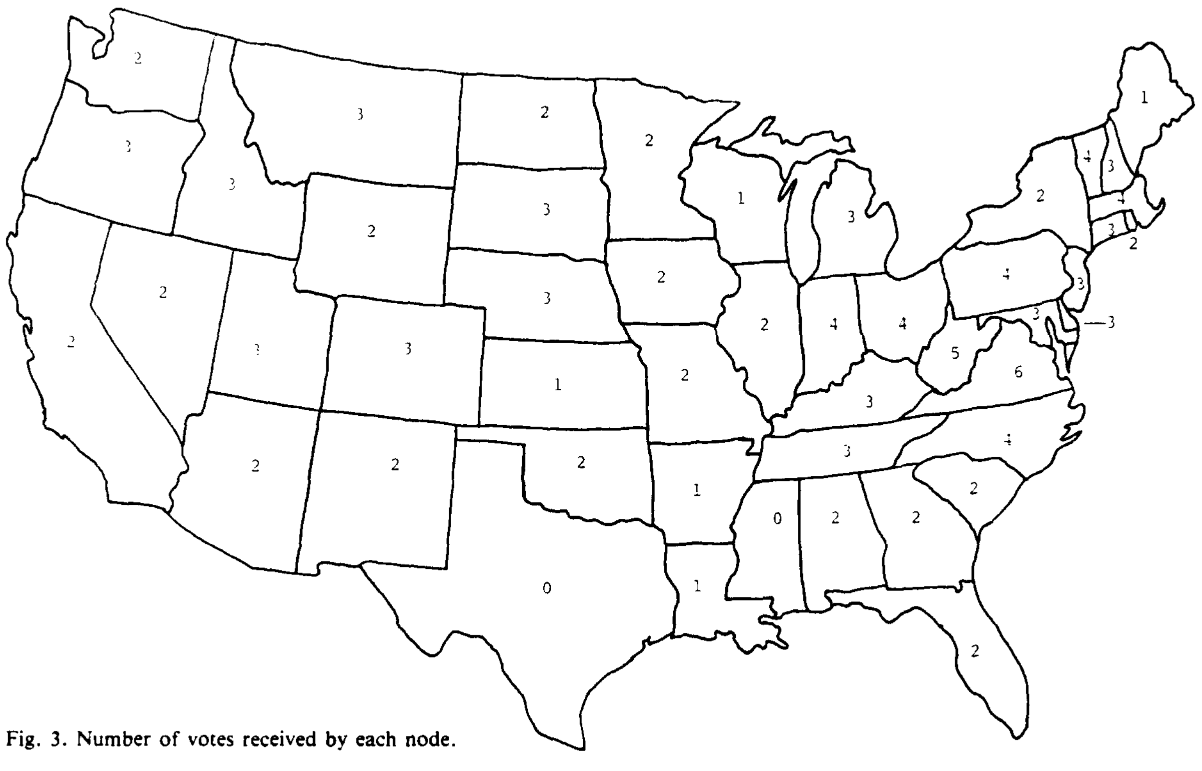

Fig. 3. Number of votes received by each node.

states with each adjacency stored twice, once in each sense. We took the subgraph composed on the six states Kentucky, Maryland, Ohio, Pennsylvania, Virginia and West Virginia, with Virginia as reference point. Notice that these particular graphs are unlabelled, but can be treated as labelled graphs for which all labels are the same. This is a worst case for the Relational Hough Transform, since the only information in a relational sequence is its path length.

Figure 2 shows the map of the states in the subgraph, and Figure 3 shows the number of votes received by each node of the full U.S.A. graph after applying the Relational Hough Transform. As can be seen, only the correct image of the reference point receives the full complement of votes, namely 6. Some other nodes have totals almost as high; examination reveals that they are reasonable partial matches.

6. Parallel network implementation

The incrementing of possible reference-point images can readily be performed on a marker-passing network machine similar to Fahlman's NETL machine (Fahlman, 1979). The machine's processing elements are connected so as to represent the arcs and nodes of the graph G_2, with flags set to represent the various arc and node labels. The control processor broadcasts the initial node label of a relational sequence; all nodes carrying this label are marked. Then the controller broadcasts the consecutive directed arc labels and node labels of the sequence; the markers are successively propagated to adjacent elements bearing the appropriate labels. After the entire sequence has been broadcast, the marked node elements increment their counters. The computation of the relational sequences from the graph G_1 can also be done in parallel in a network machine, but requires that the processing elements be capable of passing and storing descriptions of relational sequences.

Both of these computations could be readily performed on a parallel machine like ZMOB (Rieger et al., 1981), which permits arbitrary intercommunication amongst a collection of several quite substantial processing elements (each a Zilog Z80A microprocessor with 64K bytes of memory). On such a machine it should also be possible to run the relational-sequence computation and the incrementing processes simultaneously. The details of the synchronization of these two processes and of the interprocessor communication are yet to be worked out.

Graph-structured automata have been used before for solving graph-theoretic problems. In particular, Rosenfeld and Wu (Rosenfeld, 1975; Wu and Rosenfeld, 1979) give graph-automaton algorithms for subgraph isomorphism, where (using our notation) G_1 is fixed, the graph automaton has the structure of G_2 (having bounded degree), with an encoding of G_1 stored in the state transition table of each node. Our technique is different from theirs in two respects. First, ours is intended only as an approximate technique; being based on a necessary condition for subgraph isomorphism, it is useful for eliminating impossible correspondences, and in practice seems to be able to eliminate almost all of them, leaving only correct correspondences. However, this weakness is offset by the simplicity of the processing required on the nodes and arcs of G_2. Second, our construction of the network machine for G_2 is independent of G_1. Once this machine is configured, it can be used for matching against many graphs G_1, simply by feeding it the appropriate collection of relational sequences.

7. Discussion

The shape-matching Hough transform can usually find a match to a shape in an image that may be incomplete because of occlusion, low boundary contrast, or other factors. It is only necessary that enough of the shape be present to give an appreciable peak in the accumulator space. The Relational Hough Transform has similar behavior: If G_2 contains a subgraph isomorphic not to G_1 itself but to some significant piece of G_1, then the corresponding accumulator in G_2 will not be fully incremented, but will likely be incremented sufficiently to give some clue to the existence of a restricted monomorphism from G_1 to G_2. So partial matches to G_1 can be found by searching G_2 for accumulators with sums sufficiently high.

As mentioned earlier, our present implementation computes only a single relational sequence from any node in G_1 to the reference point. Using more relational sequences (if there are any) requires more computation, but will reduce the likelihood of spurious matches and also improve the ability of the technique to tolerate errors. There is also a certain redundancy in our collection of relational sequences: If a node x in G_1 lies on a path to the reference point from some other node y, then the relational sequence from x will be subsumed by that from y - any correct voting done by x's sequence will also be done by y's. However, using such redundant relational sequences will improve error tolerance: if distant parts of G_1 are missing in G_2, then the nearer parts will still make their votes for the correct image of the reference point.

Some thought should be given to the criteria to be used in building up a collection of relational sequences to describe G_1. First, what point in G_1 will serve best as reference point? Second, given that the reference point has been determined, from what other points should relational sequences be computed? And third, if there are multiple paths possible from a particular node, which of them should be used for computing relational paths? We have not pursued these issues, except to note that the use of shortest paths, with or without redundancy as remarked above, seems to offer some advantages, and that in practice it may be beneficial to prefer relational sequences whose labels have low a priori probability.

This technique, as presented above, will find possible image points in G_2 for a single reference point chosen in G_1. In order to construct a monomorphism from G_1 to G_2, this process can be repeated, taking every node of G_1, in turn, as a reference point. This way we can find possible images for every node of G_1. An obvious refinement is to take into account the location of previous reference points. If a path in G_1 passes through a node n already used as a reference point, then the corresponding paths through G_2 should be followed only if at the appropriate point they pass through a node in G_2 previously determined to be a possible image of n. This can be done by suitably annotating the node labels in G_2, and in the relational sequences. Even if this technique fails to find unique images for every node in G_1, usually it will have greatly reduced the number of possibilities, so that little work need be done by a subsequent exhaustive combinatorial search.

The Relational Hough Transform, in its general approach, may be compared to relaxation methods as applied to subgraph isomorphism and relational-structure matching (Ullmann, 1976; Kitchen and Rosenfeld, 1979; Faugeras and Price, 1981). Both attempt to solve a difficult, large combinatorial problem by decomposing the problem into pieces. In relaxation, the matching is performed using local neighborhoods of nodes immediately related; the results of these separate local matches are globally integrated. In the Relational Hough Transform, paths through the graphs are used as the basis of matching. This may well be a better approach, since the extended paths capture more global structure than do local neighborhoods. On top of this, path matching is easier to perform than neighborhood matching, because of the simpler, linear structure of paths as compared with neighborhoods.

The use of paths as a basis for matching is related to the notion of *path consistency* as used by Mackworth (1977), and (from another perspective) to some of the criteria used by Unger in his graph isomorphism program (Unger, 1964).

8. Experiments

To demonstrate the effectiveness of the Relational Hough Transform, we implemented the technique and applied it to some typical subgraph isomorphism problems. We used two target graphs. One was derived from the map of the U.S.A., as described in Section 5, but with each node labelled with the first letter of the name of the capital city of the corresponding state (16 distinct labels). The second graph was similar, based on the 44 countries of Africa (with 17 distinct node labels derived from the capitals of the countries). The average degree of nodes in either graph was about 5. These graphs are the same as those used by Kitchen and Rosenfeld (1979).

From each of these graphs we generated nine random subgraphs for every subgraph size from 3 up to 9. These subgraphs were generated as follows. A node of the full graph is chosen at random to be the reference point for the subgraph. Then the following procedure is applied to the growing subgraph until it reaches the specified size. From among the nodes not in the subgraph that are immediately connected to the subgraph by an arc (in either direction) a single node is chosen at random and added to the subgraph, along with any arcs that join it to nodes in the subgraph. This method generates connected subgraphs that tend to be fairly compact.

Every subgraph thus generated was matched against its parent graph using the Relational Hough Transform, and it was noted whether the known correct image of the reference point was unambiguously determined (that is, whether the correct image received strictly more votes than any other node in the parent graph). The number of unambiguous determinations for each subgraph size (out of the nine subgraphs considered for each size) is shown in Table 1. The results for the Africa map are shown in the second column, those for U.S.A. are shown in the third, and the fourth gives the results for matching against the U.S.A. map, but without the node labels. (In this last case, the only information in a relational sequence is its path length.) We see that this method almost always finds the correct image of the reference point unambiguously. For the larger subgraphs the results tend to be somewhat better because there is less likelihood of spurious matches.

Table 1
Number of unambiguous determinations of reference point out of 9 trials

Subgraph size	Africa with labels	USA with labels	USA without labels
3	8	5	5
4	5	8	6
5	7	9	8
6	7	8	8
7	7	9	8
8	8	9	9
9	9	8	7

9. Conclusions

We have presented a novel method for subgraph isomorphism inspired by a technique for geometric shape matching, namely the Generalized Hough Transform. This method is usually effective for finding the images of particular nodes; it can be extended, or combined with other methods, to find complete subgraph isomorphisms. It can be used to find partial matches when arcs and nodes are missing. In addition, this method is suitable for parallel implementation on a network of simple processing elements.

References

Ballard, D.H. (1981). Generalizing the Hough transform to detect arbitrary shapes. *Pattern Recognition* 13 (2), 111-122.

Ballard, D.H. and C.M. Brown (1982). *Computer Vision.* Prentice-Hall, Englewood Cliffs, NJ, Chapter 11.

Fahlman, S.E. (1979). *NETL: A System for Representing and Using Real-World Knowledge.* MIT Press, Cambridge, MA.

Faugeras, O.D. and K.E. Price (1981). Semantic description of aerial images using stochastic labelling. *IEEE Trans. Pattern. Anal. Mach. Intell.* 3(6), 633-642.

Kitchen, L.J. and A. Rosenfeld (1979). Discrete relaxation for matching relational structures. *IEEE Trans. Systems Man Cybernet.* 9(12), 869-874.

Mackworth, A.K. (1977). Consistency in networks of relations. *Artificial Intelligence* 8(1), 99-118.

Rieger, C., R. Trigg and R. Bane (1981). ZMOB: a new computing engine for AI. In: *Proceedings 7th IJCAI,* Vancouver, B.C., August 1981, pp. 955-960.

Rosenfeld, A. (1975). Networks of automata: some applications. *IEEE Trans. Systems Man Cybernet.* 5(3), 380-383.

Ullmann, J.R. (1976). An algorithm for subgraph isomorphism. *J. Assoc. Comput. Mach.* 23(1) 31-42.

Unger, S.H. (1964). GIT - a heuristic program for testing pairs of line graphs for isomorphism. *Comm. Assoc. Comput. Mach.* 7(1).

Wu, A.Y. and A. Rosenfeld (1979). Cellular graph automata. II. Graph and subgraph isomorphism, graph structure recognition. *Inform. Control* 42(3), 305-353.