2-D Scene Analysis Using Split-Level Relaxation

Tom Henderson and Ashok Samal

Computer Science Department
University of Utah
Salt Lake City, Utah 84112 USA

Abstract

We present a new method for applying multiple semantic constraints based on discrete relaxation. A separate graph is maintained for each constraint relation and is used in parallel to achieve a consistent labeling. This permits both local and global analysis without recourse to complete graphs. Here 'local' means with respect to a particular constraint graph, and thus actually includes global spatial relations on the features; e.g., parallel edges on an object will be neighbors in the parallel constraint graph even though they are far apart in Euclidean space. Another major result is a technique for handling occlusion by incorporating the use of spatially local feature sets in the relaxation-type updating method.

1. Introduction

One of the problems in computer vision is to identify the set of objects present in a given image which essentially is the *Scene Labeling Problem*. The problem can be mapped into what has been variously called the *Consistent Labelling Problem* [4], the *Satisfying Assignment Problem* [3], the *Constraint Satisfaction Problem* [7], *Waltz Filtering* [9], etc. We will refer to it as the Consistent Labeling Problem (**CLP**).

The approach used here is based on discrete relaxation [5]. However, there are some major differences. We distinguish between different types of constraints during the process of relaxation. Also, the model proposed here uses both local and global constraints. We also give a formalism to apply this technique in a parallel processing framework. For an example of the use of multiple semantic constraints with stochastic relaxation, see Faugeras and Price [2].

2. The Consistent Labeling Problem (CLP)

The basic idea is to assign labels to a set of items or units, **U** = $\{u_1, u_2, \ldots, u_n\}$. The labels can take any value from a discrete domain, **D**, which forms the set of all acceptable labels for the units. Usually, however, there are restrictions on the labels a set of units can have simultaneously in order to be consistent. These constraints are expressed by a *constraint relation* **R**. Thus the labeling problem is to find a complete consistent labeling, given a set of units **U**, the domain **D**, and the constraint relation **R**.

<CLP> ::= (**U**, **D**, **R**).

There are several ways to solve the problem: *generate and test*, *standard backtracking*, etc. It can be shown that **CLP** is NP-complete, which means that there is no efficient solution procedure. So, some approaches use a preprocessing step to reduce the computation. Mackworth [7] gives three consistency tests, *Node Consistency*, *Arc Consistency*, and *Path Consistency*, which prevent the *thrashing* behavior of the backtrack algorithms. Recently Mohr and Henderson have given an optimal algorithm for arc consistency and an improved algorithm for path consistency [8].

Now we give an informal formulation of scene analysis (SA) as a consistent labeling problem (See [6] for a formal treatment). The set of *units* which needs to be labeled is the set of feature instances found in the image. The domain of the units is the union of the features of all possible objects which can be in the scene. The constraints which control the relaxation process now are not only the constraints in the models but also the constraints in the image.

3. Relaxation Process

The graph/network model which is the underlying basis for our algorithm is similar to the model in [7]. The nodes represent the units to be labeled and the constraints are represented by the arcs in the graph. However, what we have here is conceptually closer to a family of graphs rather than a single graph. For the rest of the paper we assume that we are looking for a particular object in the image. It can be easily extended for searching multiple objects in the same scene.

The model has a set of graphs (model constraint graphs) corresponding to the different constraint types, e.g., *parallel*, *neighbor*, etc. Thus each relation has its own graph. Similarly the image has a set of graphs (image constraint graphs) corresponding to its constraint types.

Now we summarize how the relaxation process works and how it fits in a parallel processing framework. The first step is to build all the graphs, i.e., the model and the image constraint graphs and associate the labels with the nodes of the image graphs. Then the *node*, *arc*, and *path* consistencies are enforced. This is where the system lends itself to parallel execution. The graphs are independent since they represent different types of constraints and hence can be processed in parallel. After the consistencies are enforced, we find the solutions using standard backtracking. It should be pointed out that, although we treated the graphs separately, they need not be really disjoint in actual implementation. Also the structure of the graphs does not change during relaxation.

It can be shown that the above procedure is both correct and complete (see [6]). However, the performance of the algorithm is very poor. There are several ways to make it better. One simple and obvious improvement is to do a *type-checking* while assigning labels to a feature. Also if a feature in the image is hypothesized to belong to a particular object then the other features of the same object should be close by. So it is useless to consider the the portion of scene which could not have any features of the object under consideration.

This work was supported in part by NSF Grants ECS-8307483, MCS-82-21750, DCR-8506393 and DMC-8502115.

4. Extension for Occluded Scenes

Although the relaxation process works fine for non-occluded scenes, it doesn't work well if the scene is occluded. The basic reason is that the constraints don't have the same discriminating power now. If a constraint is missing in the image, it doesn't mean that the constraint actually doesn't hold. It may just mean that some or all of its associated units are occluded.

One way to get around this problem, is to use only local features like holes, corners, etc. and constraints between them (à la Local Feature Focus [1]). Instead of using constraints between the sides (or boundary edges), we use constraints between the vectors between the locations of features. We refer to these vectors as *inter-feature vectors* (or iv's). The advantage of using these iv's is that, unlike the sides, they are either present or absent; i.e., they cannot be partially missing or broken up into different parts because of occlusion. If both the features constituting an iv are present in the image, then the iv is defined, otherwise it is not. We can use the same constraints as before, but now they are between these iv's instead of the sides.

Still the constraints don't have the discriminating power to drive the relaxation process, since the constraint set in the image is incomplete. In order to drive the relaxation process we need to *seed* the label set of some units to start with. The idea is to get some positive information from the scene and then propagate it.

So we start by seeding the label sets of nodes of the graphs. Before the actual seeding is done, the iv's are constructed for the image. The details of how the iv's are constructed and how they are used in the seeding process are given in [6]. Then the control is passed on to the relaxation process, and finally to the backtracking operator. We now give the structure of the modified version of the relaxation process.

We have used two different approaches to overcome the problems described above. In the first scheme we divide the nodes (primitives) into two groups. The nodes whose labels are fixed during the seeding process, are called the *strong* nodes and others are called *weak* nodes (thus, the name "split-level" relaxation). The *strong* nodes signify positive information. During relaxation, only a strong node can affect the label set of another node. This prevents the nodes which are not really a part of the model from affecting the label sets of other nodes. Those units whose label sets are not empty at the end of the first iteration, are then changed to *strong* nodes. Nodes which do not survive the first iteration are ignored from then on and are considered not to be part of the model. After that, the relaxation process works as usual.

Another way to solve the problem is to allow a label at a node to remain, if there is at least one support for it from any other node. (In standard discrete relaxation, a label remains iff it has support from all neighboring nodes.) This is motivated by the fact that some of the constraints may be unsatisfied due to occlusion. However there has to be support for the label in each of the graph types.

5. Implementation and Results

Most of the above ideas were implemented in PSL (Portable Standard Lisp). The compiled code was run on a VAX-8600. The actual runtimes can be vastly improved if implemented on a lisp machine like the Symbolics 3600.

The system works in two phases. In the first phase (also called training phase) information about the model is computed and stored. The input to the system consists of a list of local features, their locations and the constraints between them. The different types of constraints used are *parallel*, *perpendicular*, *adjacent*, *longer-than*, etc.

In the second phase, an image description is given along with the set of constraints associated with the local features in it. We also give a model to be searched for in the image. We only check for node and arc consistency. The AC-3 algorithm as defined in [7] is used for enforcing the consistency. The output is a listing of features in the image and the corresponding features in the model. If the label set of an unit is empty, it is labelled as *unknown*.

We now present some results obtained using the algorithms described. First, we give the run-times for recognizing unoccluded objects. Figure 4 summarizes the results obtained by using the algorithms on objects in Figures 1,2, and 3. It also, gives the number of different types of constraints and total number of constraints used. Here only the boundary edges are used for recognition.

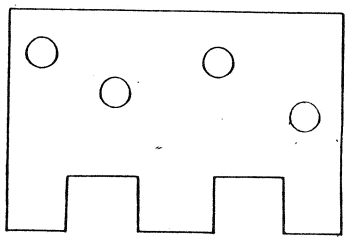

Figure 1. Object No 1

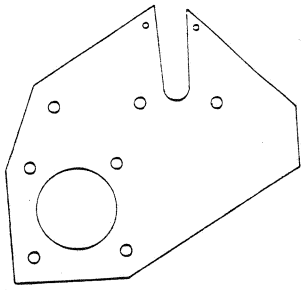

Figure 2. Object No 2

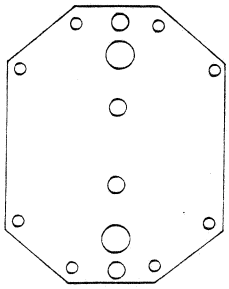

Figure 3. Object No 3

Object No	*RunTime (msec)*	*Constraint types*	*Total Number of constraints*
1	357	5	43
2	476	6	33
3	85	4	15

Figure 4. *Results for unoccluded scenes*

Next we present some results for occluded scenes. Figures 5, 6, and 7 show three scenes where some parts are occluded. These scenes were searched for the occurrence of objects 1, 2 and 3, respectively. Figure 8 gives the time taken to recognize these parts in the corresponding scene. There are two run times in the table, corresponding to the two schemes described in the previous section.

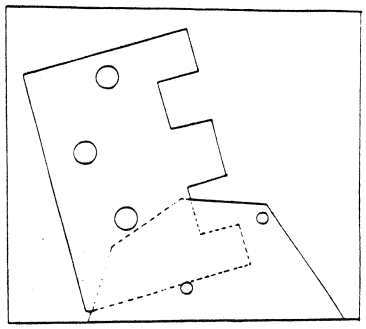

Figure 5. Occluded Scene No 1

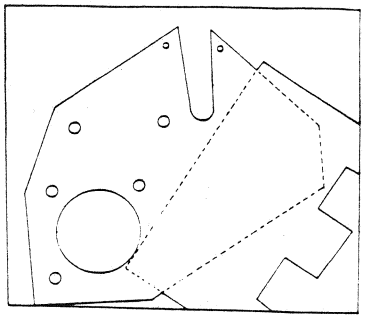

Figure 6. Occluded Scene No 2

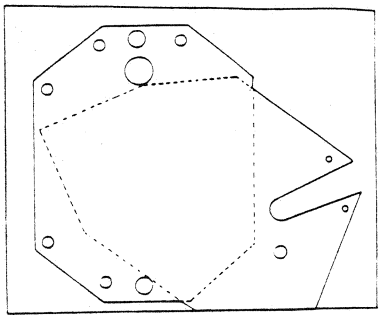

Figure 7. Occluded Scene No 3

Scene No	*RunTime(1) (msec)*	*RunTime(2) (msec)*
1	1360	6307
2	1581	2074
3	3791	26452

Figure 8. *Results for occluded scenes*

6. Conclusions and Extensions

In this paper we have formulated the scene analysis (SA) problem as a consistent labeling problem (CLP). We also gave a solution to the problem within the framework of discrete relaxation methods. However, the approach is based on a different perspective in order to exploit the inherent parallelism in the problem and to account for occlusion. We have also recommended several variations to improve the efficiency of the computation.

Although the approach sounds attractive, particularly if a multi-processor is accessible, it by no means solves all the problems. First, we are still dealing with 2-D models. Extending it to 3-D has its obvious challenges. Another problem we have not considered is the optimality of the constraint set. We have used the constraints which were obvious in the model and the image. However, this leads to redundancy and hence should be done systematically.

References

[1] Robert C. Bolles and Ronald A. Cain.
Recognizing and Locating Partially Visible Objects : The Local-Feature-Focus Method.
The International Journal of Robotics Research 1(3):57-82, Fall, 1982.

[2] Faugeras, Olivier and Keith Price.
Semantic Description of Aerial Images Using Stochastic labeling.
IEEE Transactions on Pattern Analysis and Machine Intelligence :633-642, November, 1981.

[3] John Gaschnig.
Performance Measurement and Analysis of Certain Search Algorithms.
PhD thesis, Carnegie-Mellon University, May, 1979.

[4] Robert M. Haralick and Linda G. Shapiro.
The Consistent Labelling Problem: Part I.
IEEE Transactions On Pattern Analysis And Machine Intelligence PAMI-1(2):173-184, April, 1979.

[5] Thomas C. Henderson.
A Note on Discrete Relaxation.
Computer Vision, Graphics And Image Processing 28:384-388, 1984.

[6] Tom Henderson and Ashok Samal.
Multi-constraint Shape Analysis.
Image and Vision Computing :to appear, May, 1986.

[7] Alan K. Mackworth.
Consistency in Network of Relations.
Artificial Intelligence 8:99-118, 1977.

[8] Roger Mohr and Thomas C. Henderson.
Arc and Path Consistency Revisited.
Artificial Intelligence 28(2):225-233, March, 1986.

[9] David Waltz.
Understanding Line Drawings of Scenes with Shadows.
In Patrick Henry Winston (editor), *The Psychology of Computer Vision*, pages 19-92. McGraw-Hill Book Company, 1975.

IAPR

afcet

EIGHTH INTERNATIONAL CONFERENCE ON PATTERN RECOGNITION

PARIS, FRANCE OCTOBER 27-31, 1986

PROCEEDINGS

Sponsor : IAPR, International Association for Pattern Recognition

Organizer : AFCET, Association Française pour la Cybernétique Economique et Technique