COMPUTER VISION, GRAPHICS, AND IMAGE PROCESSING **28**, 384–388 (1984)

NOTE

A Note on Discrete Relaxation

THOMAS C. HENDERSON

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

Received October 10, 1983; revised May 31, 1984

Classical relaxation techniques have been studied for many years, but only recently have researchers realized the potential for such methods in solving machine vision problems. The earliest applications of symbolic relaxation methods involved a "discrete" relaxation which is shown to be a restriction of the general relaxation process to systems of Boolean inequalities which take values over the two element set $\{0, 1\}$. © 1984 Academic Press, Inc.

1. INTRODUCTION

Classical relaxation techniques were introduced by Southwell over fifty years ago as a practical aid to the engineer in computing stresses in braced frameworks [7]. As investigation proceeded, it was shown that the method had a much wider range of application than the original domain for which it was designed. Likewise, the symbolic (as opposed to numeric) version of relaxation introduced by Waltz [9] as a method for labeling edges in line drawings has been shown to be a powerful technique of much wider significance. Our main goal is to present a unified view of relaxation, both as a numerical and as a symbolic technique. In doing so, we will try to demonstrate the range and power of this approach. For a review of the numerous applications of symbolic relaxation processes in computer vision, see Davis and Rosenfeld [1].

The classical relaxation problem typically involves a spatial arrangement of grid elements (here a square grid) such as that shown in Fig. 1. The goal is to determine a simultaneous assignment of values to the u_i's such that some constraint function, e.g., Laplace's equation, is satisfied. Such a function is transformed from a global expression, e.g., $\delta^2x/\delta x^2 + \delta^2y/\delta y^2 = 0$ to a set of local finite difference equations, e.g., $u_i - (u_{k1} + u_{k2} + u_{k3} + u_{k4})/4 = 0$, where element u_i is expressed in terms of its neighbors defined by the ki's. Thus, each of the finite difference equations can be expressed as one row of a coefficient matrix which represents the relation of one particular unknown. That is, for u_i,

$$m_{i,i}u_i + \sum_{j \neq i} m_{i,j}u_j = v_i. \tag{1}$$

(If the value of u_i is known, then $m_{i,j} = 0$ for j not equal i, $m_{i,i} = 1$, and v_i is the known value.) Thus, for n unknowns, an n by n matrix of coefficients is generated, and the ith row has nonzero elements at the positions of unknowns related to the ith unknown. The set of simultaneous linear difference equations can be written

$$Mu = v, \tag{2}$$

$$\begin{array}{ccc}
u_1 & u_2 \cdots & u_k \\
u_{k+1} & u_{k+2} \cdots & u_{2k} \\
 & \vdots & \\
u_{n-k+1} & u_{n-k+2} \cdots & u_n
\end{array}$$

FIG. 1. Typical grid layout for classical relaxation.

where M is the matrix of coefficients of the equations, u is the column vector of unknowns, and v is the column vector of known constants. The matrix M is assumed nonsingular and the diagonal entries nonzero. In the classical case, various methods exist for solving such a system, usually involving an iterative scheme when inversion of M is too expensive or impractical (see Varga [8]).

2. BOOLEAN FORMULATION OF DISCRETE RELAXATION

Suppose now that a real number solution at each unknown is no longer sought, but rather that the solution to be found involves the assignment of a set of labels at each unknown such that some constraint relation among the labels is satisfied by neighboring unknowns. Whereas the unknowns in (2) take on real number values, the unknowns in a labeling problem take on a Boolean vector value with each element in the vector corresponding to a possible label. Boolean vector operations are denoted by $'$, $\times$, t, $*$, $+$, and $\cdot$ which represent complementation, vector multiplication, transpose, Boolean "and," Boolean "or," and Boolean vector dot product, respectively. Following Rosenfeld *et al.* [5], let

$U = \{u_1, \ldots, u_n\}$ be the set of unknowns,

$\Lambda = \{\lambda_1, \ldots, \lambda_m\}$ be the set of possible labels,

$\Lambda_i = (l_1, \ldots, l_m)^t$ be the column vector describing the set of labels (i.e., zero or one) possible for u_i, where $l_j = 1$ if λ_j is compatible with u_i; 0 otherwise,

C be an m by m compatibility matrix for label pairs, where $C(i, j) = 1$ if λ_i is compatible with λ_j; 0 otherwise,

$\Lambda_{ij} = (\Lambda_i \times \Lambda_j^t) * ((\mathrm{Nei}(i, j)'E) + C)$ be an m by m compatibility matrix for u_i and u_j, where E is the m by m matrix of all 1's, and $\mathrm{Nei}(i, j) = 1$ if u_i neighbors u_j; 0 otherwise.

$\Lambda_{ij}(k)$ denotes the kth row of Λ_{ij}.

A *labeling* is a vector $L = (L_1, \ldots, L_n)^t$, where $L_i = (l_{i1}, \ldots, l_{im})^t$ in Λ_i is a Boolean vector with $l_{ij} = 1$ if label λ_j is a possible label for object u_i; 0 otherwise. A labeling is *consistent* if for every i and k,

$$l_{ik} \leq \prod_{j=1}^{n} \left[\sum_{p=1}^{m} \left(l_{ik} * l_{jp} * \Lambda_{ij}(k, p) \right) \right]. \tag{3}$$

Once a formal definition of local consistency such as (3) has been given, it is easy to see that a situation very similar to that which led to (2) now holds. However, instead of having to manipulate (3) into a form amenable to iterative solution, we merely note that (3) can be rewritten

$$l_{ik} \le l_{ik} * \prod_{j=1}^{n} \left[\sum_{p=1}^{m} \left(l_{jp} * \Lambda_{ij}(k,p) \right) \right] \qquad \text{for every } i \text{ and } k, \tag{4}$$

since the l_{ik} on the right-hand side is independent of j and p. It is clear that if (4) does not hold it can be made to hold by setting l_{ik} equal to the value on the right-hand side. This is, in fact, equivalent to discarding label k for u_i if at some neighbor u_j there does not exist a compatible label. If the l_{ik}'s, $k = 1, m$ are now gathered together in vector form:

$$\begin{bmatrix} l_{i1} \\ l_{i2} \\ \vdots \\ l_{im} \end{bmatrix} \le \begin{bmatrix} l_{i1} \\ l_{i2} \\ \vdots \\ l_{im} \end{bmatrix} * \begin{bmatrix} \sum_{p=1}^{m} \left(l_{1p} * \Lambda_{i1}(1,p) \right) \\ \sum_{p=1}^{m} \left(l_{1p} * \Lambda_{i1}(2,p) \right) \\ \vdots \\ \sum_{p=1}^{m} \left(l_{1p} * \Lambda_{i1}(m,p) \right) \end{bmatrix} * \dots * \begin{bmatrix} \sum_{p=1}^{m} \left(l_{np} * \Lambda_{in}(1,p) \right) \\ \sum_{p=1}^{m} \left(l_{np} * \Lambda_{in}(2,p) \right) \\ \vdots \\ \sum_{p=1}^{m} \left(l_{np} * \Lambda_{in}(m,p) \right) \end{bmatrix}$$

or

$$\begin{bmatrix} l_{i1} \\ l_{i2} \\ \vdots \\ l_{im} \end{bmatrix} \le \begin{bmatrix} l_{i1} \\ l_{i2} \\ \vdots \\ l_{im} \end{bmatrix} * \begin{bmatrix} L_1 * \Lambda_{i1}(1)^t \\ L_1 * \Lambda_{i1}(2)^t \\ \vdots \\ L_1 * \Lambda_{i1}(m)^t \end{bmatrix} * \dots * \begin{bmatrix} L_n * \Lambda_{1n}(1)^t \\ L_n * \Lambda_{1n}(2)^t \\ \vdots \\ L_n * \Lambda_{1n}(m)^t \end{bmatrix}$$

or

$$L_i \le L_i * \prod_{j=1}^{n} \left(\left\{ [L_j] \times \left[\Lambda_{ij}(1)^t \cdots \Lambda_{ij}(m)^t \right] \right\}^t \right). \tag{5}$$

Let

$$P = \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ \vdots \\ P_n \end{bmatrix},$$

where the column vector $P_i = \prod_{j=1}^{n}(\{[L_j] \times [\Lambda_{ij}(1) \quad \Lambda_{ij}(m)]\}^t)$ in (5). Gathering together the L_i's, $i = 1, n$, we have

$$L \leq L * P.$$

This formulation emphasizes the relation to classical relaxation.

3. DISCUSSION

The parallel iterative discrete relaxation algorithm proposed in [5] is achieved by repeating

$$L := L * P \tag{7}$$

until L does not change value. The form of Eq. (7) brings out clearly:

> The parallel iterative algorithm corresponds to the classical point Jacobi method of relaxation; the Gauss-Seidel method can also be used by taking advantage of already updated unknowns, i.e., asynchronous updating is possible.

This formulation of discrete relaxation permits greater insight into the nature and use of discrete relaxation. Namely, one important insight gained by studying this formulation is that a single underlying architecture can be used to perform both classical and Boolean relaxation where the only essential difference is at the lowest level of the particular operations performed, that is, arithmetic or logical. Given that many low-level vision operations can be performed in terms of classical relaxation, and that certain symbolic processes can be set up as a symbolic relaxation process, this could provide a possible underlying mechanism for both forms of processing.

In comparing classical relaxation to Boolean relaxation, we see that in classical relaxation local constraints along with the boundary conditions are sufficient to determine a unique global fixed point for the system. The initial vector affects the speed of convergence, but not the final answer. On the other hand, in the Boolean relaxation scheme, once a label is zero, it will stay zero. Moreover, the binary constraints do not necessarily constrain the system to a unique fixed point. In fact, the fixed point achieved by the algorithm is that consistent labeling which requires the elimination of the fewest labels from the initial set, and even if nonempty, does not in fact guarantee an unambiguous labeling. In general, the process does however, significantly reduce the search space.

Since most of the classical theory developed by Varga is for real-valued solutions, it is too general for direct application when the matrices under consideration are Boolean valued. For a more appropriate theory for the solution of Boolean-valued simultaneous equations, see Rudeanu [6].

It should be pointed out that Hummel and Zucker [3] have shown that the more general stochastic relaxation labeling process can be nicely formulated in the framework of variational calculus. They use the notion of consistency to tie together discrete and stochastic relaxation. The definition they give counts the number of supporting neighbors rather than simply determining in a yes/no manner whether sufficient support exists as described here.

The generalization of discrete relaxation to encompass n-ary relations has been studied by several authors [2, 4]. In the context of Eq. (3), this merely involves anding all n labels and using an n-dimensional matrix for the consistency relation.

REFERENCES

1. L. Davis and A. Rosenfeld, Cooperating processes for low level vision; A survey, *Artif. Intell.* **17**, 1981, pp. 245–63.
2. E. C. Freuder, *Synthesizing Constraint Expressions*, Technical Report A1 Memo No. 310, MIT, July 1976.
3. R. A. Hummel, and S. W. Zucker, On the foundations of relaxation labeling processes, *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-5**, No. 3, 1983, 267–286.
4. A. K. Mackworth, Consistency in networks of relations, *Artif. Intell.* **8**, 1977, 99–118.
5. A. Rosenfeld, R. Hummel, and S. Zucker, Scene labeling by relaxation operations, *IEEE Trans. Systems, Man, Cybernet.* **SMC-6**, No. 6, 1976, 420–433.
6. S. Rudeanu, *Boolean Functions and Equations*, North–Holland, Dordrecht, 1974.
7. R. V. Southwell, *Relaxation Methods in Engineering Science*, Oxford Univ. Press, London, 1940.
8. R. S. Varga, *Series in Automatic Computation*: *Matrix Iterative Analysis*, Prentice-Hall, Englewood Cliffs, N.J., 1962.
9. D. Waltz, Understanding line drawings of scenes with shadows, in *The Psychology of Computer Vision* (P. H. Winston, Ed.) pp. 19–91, McGraw-Hill, New York, 1975.