On the Foundations of Relaxation Labeling Processes

ROBERT A. HUMMEL, MEMBER, IEEE, AND STEVEN W. ZUCKER, MEMBER, IEEE

Abstract—A large class of problems can be formulated in terms of the assignment of labels to objects. Frequently, processes are needed which reduce ambiguity and noise, and select the best label among several possible choices. Relaxation labeling processes are just such a class of algorithms. They are based on the parallel use of local constraints between labels. This paper develops a theory to characterize the goal of relaxation labeling. The theory is founded on a definition of consistency in labelings, extending the notion of constraint satisfaction. In certain restricted circumstances, an explicit functional exists that can be maximized to guide the search for consistent labelings. This functional is used to derive a new relaxation labeling operator. When the restrictions are not satisfied, the theory relies on variational calculus. It is shown that the problem of finding consistent labelings is equivalent to solving a variational inequality. A procedure nearly identical to the relaxation operator derived under restricted circumstances serves in the more general setting. Further, a local convergence result is established for this operator. The standard relaxation labeling formulas are shown to approximate our new operator, which leads us to conjecture that successful applications of the standard methods are explainable by the theory developed here. Observations about convergence and generalizations to higher order compatibility relations are described.

Index Terms—Consistency, constraint satisfaction, cooperative processes, labeling, probabilistic relaxation, relaxation labeling.

Manuscript received June 1, 1981; revised July 19, 1982. This work was supported by ARO Grant DAAG29-81-K-0043 and NSERC Grant A4470.

R. A. Hummel is with the Courant Institute of Mathematical Sciences, New York University, New York, NY 10012.

S. W. Zucker is with the Computer Vision and Graphics Laboratory, Department of Electrical Engineering, McGill University, Montreal, P.Q., Canada H3A 2A7.

MOTIVATION

RELAXATION labeling processes are a class of mechanisms that were originally developed to deal with ambiguity and noise in vision systems. The general framework, however, has far broader potential applications and implications. The

0162-8828/83/0500-0267$01.00

structure of relaxation labeling is motivated by two basic concerns: 1) the decomposition of a complex computation into a network of simple "myopic," or local, computations; and 2) the requisite use of context in resolving ambiguities. Given impetus by successes in constraint propagation, particularly in "blocks-world" vision [16], a broad class of algorithms was introduced and subsequently explored under the generic heading of relaxation labeling operations [12].

Although many technical details in the design and implementation of relaxation labeling algorithms have been subject to ad hoc, or heuristic choices, the general structure has attracted substantial interest. Our goal in this paper is to provide a formal foundation. The algorithms are conceptually parallel, allowing each process to make use of the context to assist in a labeling decision. Further, there is an appealing qualitative agreement between the parallel structure of relaxation algorithms and the distributed appearance of the neural machinery in the early stages of the human visual system. There has been a lot of hope and some evidence that relaxation labeling operations are robust—that the resulting processes are relatively insensitive to large changes in design parameters. The lure of relaxation labeling is largely based on a desire to achieve a globally consistent interpretation by using a fixed simple control structure together with some common sense appraisals of local constraints. Since it can be applied to any problem which can be posed as a labeling problem, interest in the algorithms has inspired applications in domains very different from computer vision [4].

When relaxation operations are used to solve systems of linear equations, or equivalently, discretized partial differential equations, many of the parameters and choices about the graph structure are dictated by the problem domain and the underlying equations. However, sophisticated relaxation applications left a number of choices to the "computors," or project leaders [13]. For example, the use of overrelaxation and block relaxation methods are "labor-saving devices" whose utility is demonstrated by practical experience.

Relaxation labeling is a natural extension of relaxation operations to the class of problems whose solutions involve symbols rather than functions. Constraints between neighboring labels replace finite difference equations used to represent the local behavior imposed by differential equations. The relaxation of labels occurs by manipulating assignment weights attached to separate labels, as opposed to adjusting the estimated function value up or down. Indeed, the main difference between relaxation labeling and relaxation is that the labels do not necessarily have a natural ordering. One might be literally comparing apples and oranges, and relaxation labeling might point to an adjustment from the label of apples to the label of oranges. When using classical relaxation to find solution functions, the adjustments of the function values are always either up or down.

Most applications of relaxation labeling, however, have not been guided by an underlying differential equation or system of linear equations. As a result, most of the parameters and much of the control structure had to be justified by empirical support. As long as applications are successful, researchers generally feel little motivation to further justify the approach with an analysis of underlying equations. When applications are less than perfectly successful, it is impossible to ascribe failure. Is the problem with the application of the algorithm, or is the problem with the algorithm? Usually, one documents the pattern of failure, and understands the behavior in light of the particular design parameters. Without a reasonable, abstract characterization of what the algorithm is doing, it is impossible to attribute the cause of failure to an inadequate theory. In order to assist in the development of relaxation labeling applications with a predictable domain of success, a complete characterization of the computation underlying the algorithm, independent of the application, is necessary. This paper is intended as a contribution toward such a characterization. Using this foundation, the design of relaxation processes need no longer be based entirely on ad hoc principles and heuristic choices.

In order to accomplish this characterization, the treatment is necessarily abstract. We also need to cover a lot of material: to relate discrete relaxation to a description of the usual relaxation labeling schemes, to develop a theory of consistency, and to formalize its relationship to optimization. Several mathematical results also follow from the formulation.

Other frameworks for the foundations of relaxation labeling have been attempted. We briefly compare our approach to some of these alternative viewpoints in Section II. Our framework is derived from variational calculus, i.e., from a generalization of standard optimization techniques, and central to this framework is an explicit notion of consistency. It is this notion of consistency that provides the link between discrete and continuous relaxation labeling, and serves as a foundation from which relaxation updating formulas can be derived. This central notion is developed in Section III. We begin by introducing the domain of problems to which relaxation labeling is applicable.

I. Introduction to Labeling Problems

In a labeling problem, one is given:

1) a set of objects;
2) a set of labels for each object;
3) a neighbor relation over the objects; and
4) a constraint relation over labels at pairs (or n-tuples) of neighboring objects.

Generally speaking, a solution to a labeling problem is an assignment of labels to each object in a manner which is consistent with respect to the constraint relation 4) above. Noticing that 1) and 3) above define a graph, the problem can also be described as one of assigning labels to nodes in a graph. With the graph structure in mind, we will sometimes refer to objects as nodes.

To make these terms more precise, it is useful to consider the historically important example of labeling edges in an ideal image of polyhedral solids [16]. The objects in this example represent the line segments of a line drawing of the polyhedra. Each object is a single line segment, and arises from an edge or portion of an edge of a polyhedron. Two objects, or nodes, are considered neighbors, and therefore joined by an arc in the graph structure, if the two line segments represented by the objects meet at a junction or vertex. The labels indicate

the interpretation of the physical configuration giving rise to the edge: edges can be formed by the convex joining of two surfaces, concave joining, or as an occlusion boundary with the object to the right or left of the directed edge. Constraints are provided by lists of physically realizable edge vertices. For example, three lines representing convex edges can meet, but three occlusion edges cannot meet at a single junction. The goal of the labeling process is to associate with each node a label that describes the actual correct interpretation of each corresponding edge. In cases when the line drawing is ambiguous and could conceivably be used to represent two or more different polyhedral solids, each node should be labeled with the set of all labels which can arise from a correct interpretation.

The determination of which physically realizable scenes are described by a given line drawing can be viewed as a combinatorial search. However, by requiring that labels be consistent locally, i.e., by demanding that the edges form legitimate vertices, it is usually possible to prune the space that must be searched. Discrete relaxation labeling is a process which performs this pruning in a parallel, local, fashion [6].

To abstract and formalize the situation, consider a graph with a set of labels attached to each node. We shall denote the nodes by the variable i, which can take on integer values between 1 and n (the number of nodes), the set of labels attached to node i by Λ_i, and the individual label (elements of Λ_i) by the variable λ.[1] For simplicity, we will assume that the number of labels at each node is m, independent of i, so that the variable λ takes on integer values from 1 to m. The constraint Λ_{ij} is the set of all pairs (λ, λ') such that label λ at object i is compatible with label λ' at object j. Label pairs in $\Lambda_i \times \Lambda_j$ which are not in Λ_{ij} represent pairs of incompatible labels at the corresponding objects i and j. Constraints are only defined over neighboring nodes.

Discrete relaxation is accomplished by means of the *label discarding rule:* discard a label λ at a node i if there exists a neighbor j of i such that every label λ' currently assigned to j is incompatible with λ at i, i.e., $(\lambda, \lambda') \notin \Lambda_{ij}$ for all λ' assigned to j. The discrete relaxation labeling process is defined by the iterative application of the label discarding rule, applied in parallel at each node, until limiting label sets are obtained. Note that the label discarding rule prescribes that a label is retained if at every neighboring node there exists at least one compatible label, and that this property will hold for all labels in the limit sets.

A numerical formulation can be used to give an alternative characterization of labels in the limit sets. Define variables $p_i(\lambda)$ that indicate whether label λ is associated with node i, according to

$$p_i(\lambda) = \begin{cases} 1 & \text{if } \lambda \text{ is associated with object } i \\ 0 & \text{if } \lambda \text{ is not associated with object } i. \end{cases}$$

Let the variables $R_{ij}(\lambda, \lambda')$ represent the constraints by

$$R_{ij}(\lambda, \lambda') = \begin{cases} 1 & \text{if } (\lambda, \lambda') \in \Lambda_{ij} \\ 0 & \text{if } (\lambda, \lambda') \notin \Lambda_{ij}. \end{cases}$$

[1] A glossary of symbols is included in Appendix B.

We can count the number of neighbors of an object i which has labels compatible to a given label λ at i by the support function

$$S_i(\lambda) = \sum_{j \text{ neighboring } i} \max_{\lambda' \in \Lambda_j} \{R_{ij}(\lambda, \lambda') p_j(\lambda')\}.$$

In the limit sets, a label λ which is associated with a node i has support from all neighbors, and thus

$$S_i(\lambda) \geq S_i(\lambda'), \quad \text{for all } \lambda' \in \Lambda_i.$$

That is, labels that have been discarded have support which is strictly less than $S_i(\lambda)$, because at least one of the terms in the sum obtained from a max over $\lambda' \in \Lambda_j$ is zero.

We can define the map which assigns sets of labels to each object in a manner which is invarient with respect to the label discarding rule (i.e., a limit set) to be a *consistent labeling* with respect to the constraints. We see that if support is suitably defined, labels in a consistent labeling have maximal support at each object. This is the notion which serves as the basis for our subsequent studies of continuous relaxation labeling processes.

II. Continuous Relaxation Labeling Processes

The constraints used in the labeling problem described in the previous section do not allow for labels to express a preference or relative dislike for other labels at neighboring nodes. Instead, pairs of labels are either compatible or completely incompatible. Continuous relaxation labeling attempts to allow greater flexibility in the constraints by replacing these logical assertions about compatibilities with weighted values representing relative preferences. That is, the constraints are generalized to real-valued compatibility functions $r_{ij}(\lambda, \lambda')$ signifying the relative support for label λ at object i that arises from label λ' at object j. This support can be either positive or negative. (Some formulations require that the compatibility values satisfy $-1 \leq r_{ij}(\lambda, \lambda') \leq 1$, but we will make no such restriction.) Generally, positive values indicate that labels form a locally consistent pair, whereas a negative value indicates an implied inconsistency. The magnitude of $r_{ij}(\lambda, \lambda')$ is proportional to the strength of the constraint. When there is no interaction between labels, or when i and j are not neighbors, the compatibility $r_{ij}(\lambda, \lambda')$ is zero.

Having given the compatibilities weights, continuous relaxation also uses weights for label assignments. We denote the weight with which label λ is assigned to node i by $p_i(\lambda)$, and will require that

$$0 \leq p_i(\lambda) \leq 1, \quad \text{all } i, \lambda$$

and

$$\sum_{\lambda=1}^{m} p_i(\lambda) = 1, \quad \text{all } i = 1, \cdots, n.$$

As an example of these ideas, consider the problem of detecting and labeling lines in digital imagery [17], [18]. Initial assertions about the presence of lines can be established at every pixel location on the basis of the response of a local line

detector. However, the responses are usually ambiguous to some degree: there may be gaps, weak responses, and multiple responses in different orientations at some pixels. In a relaxation labeling approach, each pixel would represent a distinct object, and the label sets are formed from assertions about the existence of a line at that pixel in a given orientation, or the nonexistence of a line at that pixel. Initial assignments of the values $p_i(\lambda)$ are accomplished by inspecting the local line detector responses. Constraints can be obtained from a local model for good continuation of line elements. Thus, a horizontal line label supports a horizontal line label positioned (horizontally) next to it. The relaxation process iteratively updates the weighted label assignments to be more consistent with neighboring labels, ideally so that the weights designate a unique label at each node.

A conceptual difficulty remains in attempting to abstract this situation to the general problem of formulating a consistent labeling. We have not defined the precise meaning of consistency to replace the notion used for discrete relaxation labeling. Generally speaking, a consistent labeling is one in which the constraints are satisfied. But since we have replaced logical constraints by weighted assertions, a new foundation is required to describe the structural framework and the precise meaning of the goal of consistency.

Many such structural frameworks have been attempted. Some have defined consistency as the stopping points of a standard relaxation labeling algorithm. This approach is circular, however, and gives no clue as to the weaknesses in the standard algorithm.

Many researchers have regarded the label weights as probabilities, and attempted to describe the relaxation labeling process in terms of a Bayesian analysis [11]. The constraints are generally interpreted as statistical quantities, related to conditional probabilities or correlations between, e.g., pairs of labels. From our perspective, analysis of relaxation labeling operations within the probabilistic framework has been unsuccessful. Various independence assumptions are required, and the analysis at best leads to an approximate understanding of one and only one iteration of the process. Our approach is very different, and at no time do we interpret the $p_i(\lambda)$'s as probabilities.

An alternate development, based on optimization theory, has used as a definition of consistency the norm of the difference between a vector composed of the current label weights and an evidence vector obtained from a computation involving each label's neighborhood weights [2], [5]. This measure can be minimized by standard methods (see also [1], [14]). We believe that this approach is fundamentally the more appropriate one, although the goal of finding the best functional to minimize is limiting. In this paper we extend the optimization viewpoint to treat relaxation from foundations that are more basic than the presupposition of such functionals.

A related approach develops relaxation labeling entirely within the constructs of linear programming [7]. In this theory, the constraints are obtained from arithmetical equivalents to logical constraints, and preferences can be incorporated only by adding new labels. Consistent labelings form a convex subset of assignment space. This viewpoint is different from our theory, but interesting and not incompatible with our development.

Each of these different structural frameworks for explaining the goal of relaxation labeling leads to a variant algorithm. In the next section, we begin with a formulation of continuous labelings and compatibilities, and then define consistency. These notions eventually yield (Section VIII) yet another relaxation labeling algorithm. However, all of these variants share similarities with the original algorithm defined in [12] (see Section XI), which was justified purely by heuristic arguments. This prototype algorithm is an iterative, parallel procedure analogous to the label discarding rule used in discrete relaxation. Specifically, for each object and each label, one computes

$$q_i(\lambda) = \sum_{j=1}^{n} \sum_{\lambda'=1}^{m} r_{ij}(\lambda, \lambda')\, p_j(\lambda')$$

using the current assignment values $p_i(\lambda)$. Then new assignment values are defined to replace the current values according to the ad hoc formula

$$p_i(\lambda) := \frac{p_i(\lambda)\,[1 + q_i(\lambda)]}{\sum_{l=1}^{m} p_i(l)\,[1 + q_i(l)]}.$$

In light of the foundations developed here, new formulas will be offered that have important advantages over these original ones.

III. Consistency

The principal contribution of this paper is a new definition of a consistent labeling. Rather than defining consistency in terms of a set of logical constraints, we will require a system of inequalities. In a sense, then, we are permitting the logical constraints to be ordered, or weighted, with respect to their relative importance. This leads to much more flexibility in defining and analyzing consistent labelings than earlier treatments. Furthermore, our approach allows an analytic, rather than logical or symbolic, study. For example, in Section IX, we use techniques from the theory of ordinary differential equations to prove a local convergence theorem.

Our definition of consistency is considerably different from more familiar treatments of consistency in labeling problems [6]. However, we will attempt to show that our formulation is a natural one, embodying an intuitive notion of consistency, and leading to a rich mathematical theory. The term will be defined for both unambiguous labelings and for weighted labeling assignments. We begin by giving a formal definition of these two spaces of labelings.

An *unambiguous labeling assignment* is a mapping from the set of objects into the set of all labels, so that each object is associated with exactly one label. The mapping can be represented (inefficiently) by a collection of binary digits, $p_i(\lambda)$, indicating whether label λ is assigned to object i. We set

$$p_i(\lambda) = \begin{cases} 1 & \text{if object } i \text{ maps to label } \lambda \\ 0 & \text{if object } i \text{ does not map to } \lambda. \end{cases}$$

Note that for each object i,

$$\sum_{\lambda=1}^{m} p_i(\lambda) = 1.$$

The variables $p_i(1), \cdots, p_i(m)$ can be viewed as composing an m-vector $\bar{p}_i$, and the concatenation of the vectors $\bar{p}_1, \bar{p}_2, \cdots, \bar{p}_n$ can be viewed as forming an assignment vector $\bar{p} \in \mathbb{R}^{nm}$. The space of unambiguous labelings is defined by

$$\mathbb{K}^* = \Big\{ \bar{p} \in \mathbb{R}^{nm} : \bar{p} = (\bar{p}_1, \cdots, \bar{p}_n); \quad \bar{p}_i = (p_i(1), \cdots, p_i(m)) \in \mathbb{R}^m; \quad p_i(\lambda) = 0 \text{ or } 1, \quad \text{all } i, \lambda; \quad \sum_{\lambda=1}^{m} p_i(\lambda) = 1 \quad \text{for } i = 1, \cdots, n \Big\}.$$

Given a vector $\bar{p}$ in $\mathbb{K}^*$, the corresponding unambiguous assignment is determined exactly. That is, the set of vectors in $\mathbb{K}^*$ is in one-to-one correspondence with the set of mappings from objects to labels.

The extension to *weighted labeling assignments* can be done by replacing the condition $p_i(\lambda) = 0$ or 1 by the condition $0 \leqslant p_i(\lambda) \leqslant 1$ for all i and λ. This extension yields the space of weighted labeling assignments

$$\mathbb{K} = \Big\{ \bar{p} \in \mathbb{R}^{nm} : \bar{p} = (\bar{p}_1, \cdots, \bar{p}_n); \quad \bar{p}_i = (p_i(1), \cdots, p_i(m)) \in \mathbb{R}^m; \quad 0 \leqslant p_i(\lambda) \leqslant 1 \quad \text{for all } i, \lambda; \quad \sum_{\lambda=1}^{m} p_i(\lambda) = 1 \quad \text{for } i = 1, \cdots, n \Big\}.$$

We claim that $\mathbb{K}$ is simply the convex hull of $\mathbb{K}^*$. To see this, let $\bar{e}_k$ denote the standard unit m-vector with a 1 in the kth component. Then any labeling assignment $\bar{p}$ in $\mathbb{K}$ can be expressed by

$$\bar{p} = \sum_{l_1=1}^{m} \sum_{l_2=1}^{m} \cdots \sum_{l_n=1}^{m} p_1(l_1)\, p_2(l_2) \cdots p_n(l_n) \cdot (\bar{e}_{l_1}, \bar{e}_{l_2}, \cdots, \bar{e}_{l_n}).$$

Since each nm-vector $(\bar{e}_{l_1}, \cdots, \bar{e}_{l_n})$ is in $\mathbb{K}^*$, the above sum can be interpreted as a convex combination of the elements of $\mathbb{K}^*$. Note that the sum over all of the coefficients is 1.

The notion of consistency depends upon constraints between label assignments. These will be represented by a matrix of real numbers—*the compatibility matrix*—the elements of which can indicate both positive and negative constraints. Compatibility between pairs, for example, is denoted by $r_{ij}(\lambda, \lambda')$, representing how label λ' at object j influences label λ at object i. If object j having label λ' lends high support to object i having label λ, then $r_{ij}(\lambda, \lambda')$ should be large and positive. If the constraint is such that object j having label λ' means that label λ at object i is highly unlikely, then $r_{ij}(\lambda, \lambda')$ should be negative. No restrictions are placed on the magnitudes of the constraints. If there is no interaction between (i, λ) and (j, λ'), then $r_{ij}(\lambda, \lambda') = 0$.

In Section I, we defined the support $S_i(\lambda)$ of label at node i by $\Sigma \max_\lambda \{r_{ij}(\lambda, \lambda')\, p_j(\lambda')\}$. A maximum operation was used because more than one label with assignment value $p_j(\lambda') = 1$ could occur at each node j. Since our labeling spaces now require $\Sigma_\lambda\, p_i(\lambda) = 1$, we replace the max operator with a sum.

Definition 3.1: Let a matrix of compatibilities and a labeling assignment (unambiguous or not) be given. We define the *support* for label λ at object i by the assignment $\bar{p}$ by

$$s_i(\lambda) = s_i(\lambda; \bar{p}) = \sum_{j=1}^{n} \sum_{\lambda'=1}^{m} r_{ij}(\lambda, \lambda')\, p_j(\lambda'). \qquad \square$$

Note that the support value is a linear function of the components of the labeling $\bar{p}$. In this case, the support values depend on the coefficients $\{r_{ij}(\lambda, \lambda')\}$ comprising the matrix of compatibilities, which we assume given along with the objects and labels as part of the problem definition. In fact, the support for label λ at object i is a weighted average of the compatibilities of currently assigned labels at neighboring objects with the label λ at node i.

The exact functional dependence of the support values on the assignment $\bar{p}$ is in fact unimportant to the theory which follows. It only matters that support values $s_i(\lambda; \bar{p})$ depending on $\bar{p}$ are given. For example, it is possible to define the support function in terms of higher order combinations of object labels. Suppose that $r_{ijk}(\lambda, \lambda', \lambda'')$ represents the compatibility of object i having label λ with the configuration of label λ' at object j and label λ'' at object k. Then, given the multidimensional matrix of compatibilities $\{r_{ijk}(\lambda, \lambda', \lambda'')\}$, we can define the support at object i for label λ by the assignment $\bar{p} \in \mathbb{K}$ as

$$s_i(\lambda) = \sum_{j, \lambda'} \sum_{k, \lambda''} r_{ijk}(\lambda, \lambda', \lambda'')\, p_j(\lambda')\, p_k(\lambda'').$$

More generally, the support values $s_i(\lambda)$ combine to give a support vector $\bar{s}$ that is a function of the total $\bar{s} = \bar{s}(\bar{p})$. The form of the vector valued function serves as a representation for the (*a priori*) knowledge of the relative compatibility of all configurations of labelings. In the following, $s_i(\lambda)$ are the components of any one of these support functions, calculated for the specified assignment $\bar{p}$. Whenever a formula for $s_i(\lambda)$ is needed, we find it convenient to use the linear dependence defined in Definition 3.1. Historically, and because of its simplicity, the case when $\bar{s}$ is linear is of special interest, but the entire theory presented in this paper can be extended to general nonlinear support functions.

We can now define consistency for unambiguous labelings.

Definition 3.2: Let $\bar{p} \in \mathbb{K}^*$ be an unambiguous labeling. Suppose that $\lambda_1, \cdots, \lambda_n$ are the labels which are assigned to objects $1, \cdots, n$, respectively, by the labeling $\bar{p}$. That is, $\bar{p} = (\bar{e}_{\lambda_1}, \bar{e}_{\lambda_2}, \cdots, \bar{e}_{\lambda_n})$. The unambiguous labeling $\bar{p}$ is *consistent* (in $\mathbb{K}^*$) providing

$$s_i(\lambda_i) \geqslant s_i(\lambda), \quad \text{all } \lambda, \quad \text{for } i = 1, \cdots, n. \qquad \square$$

It is important to realize that consistency in $\mathbb{K}^*$ corresponds to satisfying a system of inequalities:

$$s_1(\lambda_1; \bar{p}) \geqslant s_1(\lambda; \bar{p}), \quad 1 \leqslant \lambda \leqslant m$$

$$s_2(\lambda_2; \bar{p}) \geqslant s_2(\lambda; \bar{p}), \quad 1 \leqslant \lambda \leqslant m$$

$$s_n(\lambda_n; \bar{p}) \geqslant s_n(\lambda; \bar{p}), \quad 1 \leqslant \lambda \leqslant m$$

The consistency condition is a reasonable one. At a consistent unambiguous labeling, the support, at each object, for the assigned label is the maximum support at that object. However, also note that consistency is *not* a pure maximization problem (although see Section V for a partial retraction). If we change the assignment of labels, it may happen that some or all of the maximum supports at individual objects may increase. Such a labeling is not judged to be better; indeed, if the maximum support is no longer attained by the instantiated label at one or more of the nodes, then consistency is spoiled.

Given a set of objects, labels, and support functions, there may be many consistent labelings. Our notion of consistency does not place an objective ranking on labelings. The system of inequalities is simply a criterion for determining whether a given unambiguous labeling (one of the n^m elements in $\mathbb{K}^*$) is consistent. There is an analogy with mathematical programing, in that a consistent labeling is a feasible point in $\mathbb{K}^*$. However, there is no objective function, $\mathbb{K}^*$ is discrete, and the constraints are nonlinear. We can view consistency as a "locking-in" property. That is, since the support for a given label at a given node depends upon the assigned labels at all other nodes, these assigned labels dictate, through the compatibilities and support functions, values for the support which are consistent with the current assignment. Further, this agreement between the assigned label and the ordering of the support values among the label set must hold at every object.

The condition for consistency in $\mathbb{K}^*$ can be restated as follows:

$$\sum_{\lambda=1}^{m} p_i(\lambda)\, s_i(\lambda; \bar{p}) \geqslant \sum_{\lambda=1}^{m} v_i(\lambda)\, s_i(\lambda; \bar{p}), \quad i = 1, \cdots, n,$$

for all unambiguous labelings $\bar{v} \in \mathbb{K}^*$.

Note that the condition consists of n inequalities which must hold simultaneously for every competing labeling $\bar{v} \in \mathbb{K}^*$. We emphasize that in this formulation, and in the next definition, the supports are calculated for the given assignment $\bar{p}$, and do not change as the vector $\bar{v}$ varies. Accordingly, if $\bar{p}$ assigns label λ_i to object i, then the left side of this inequality evaluates to $s_i(\lambda_i)$, whereas the right-hand side is simply $s_i(\lambda)$ for some λ, depending on which label $\bar{v}$ designates to label object i.

Consistency for weighted labeling assignments is defined analogously.

Definition 3.3: Let $\bar{p} \in \mathbb{K}$ be a weighted labeling assignment. Then $\bar{p}$ is *consistent* (in $\mathbb{K}$) providing

$$\sum_{\lambda=1}^{m} p_i(\lambda)\, s_i(\lambda; \bar{p}) \geqslant \sum_{\lambda=1}^{m} v_i(\lambda)\, s_i(\lambda; p), \quad i = 1, \cdots, n,$$

for all labelings $\bar{v} \in \mathbb{K}$.

Once again, consistency in $\mathbb{K}$ is not a maximization problem, since n quantities, each depending on $\bar{p}$, simultaneously satisfy maxima problems: $\bar{p}_i \cdot \bar{s}_i = \max \bar{v}_i \cdot \bar{s}_i$, for $i = 1, \cdots, n$. Changing $\bar{p}$ might increase some of the n quantities on the left-hand side, but it might decrease some of the others. One could maximize the sum or product of the n quantities, but this does not (generally) guarantee that each quantity is individually maximized.

In Section IX, we will find it useful to consider labelings which are strictly consistent.

Definition 3.4: Let $\bar{p} \in \mathbb{K}$. Then $\bar{p}$ is *strictly consistent* providing

$$\sum_{\lambda=1}^{m} p_i(\lambda)\, s_i(\lambda; \bar{p}) > \sum_{\lambda=1}^{m} v_i(\lambda)\, s_i(\lambda; \bar{p}), \quad i = 1, \cdots, n,$$

for all labeling assignments $\bar{v} \in \mathbb{K}$, $\bar{v} \neq \bar{p}$. □

In Section IX we will see that a strictly consistent labeling $\bar{p}$ is necessarily unambiguous, i.e., $\bar{p} \in \mathbb{K}^*$.

The intuitive meaning of consistency is easier to consider at a strictly consistent labeling. Suppose one were to discard knowledge of the label assigned to a given object, but retain such knowledge at all neighboring objects. Then by computing the support values, the original labeling at the object could be deduced by maxima selection among the support values. We see that the support values and the labeling are in complete agreement. This is a very special property, since the instantiated label contributes to the calculation of the support values of neighboring object-labels, which must independently satisfy the same agreement principle.

Note that an unambiguous assignment that is consistent in $\mathbb{K}$ will also be consistent in $\mathbb{K}^*$, since $\mathbb{K}^* \subseteq \mathbb{K}$. The converse is also true.

Proposition 3.5: An unambiguous labeling which is consistent in $\mathbb{K}^*$ is also consistent in $\mathbb{K}$.

Proof: This follows because any weighted assignment $\bar{v} \in \mathbb{K}$ can be written as

$$\bar{v}_i = \sum_{l=1}^{m} v_i(l)\, \bar{e}_l$$

where $\bar{e}_k$ is the standard unit m-vector with a 1 in the kth coordinate. Since

$$\sum_{\lambda=1}^{m} p_i(\lambda)\, s_i(\lambda) \geqslant \sum_{\lambda=1}^{m} e_l(\lambda)\, s_i(\lambda), \quad l = 1, \cdots, m, \quad i = 1, \cdots, n,$$

the same inequality holds true for any convex combination:

$$\sum_{\lambda=1}^{m} p_i(\lambda)\, s_i(\lambda) \geqslant \sum_{l=1}^{m} v_i(l) \sum_{\lambda=1}^{m} e_l(\lambda)\, s_i(\lambda)$$

$$= \sum_{\lambda=1}^{m} v_i(\lambda)\, s_i(\lambda), \quad i = 1, \cdots, n$$

which proves that $\bar{p}$ is consistent in $\mathbb{K}$. □

However, there may be consistent assignments in $\mathbb{K}$ which are ambiguous, and thus not consistent in $\mathbb{K}^*$.

IV. Overview of Results

Having established the notion of consistency, we wish to develop algorithms for converting a given labeling into a consistent one. We will pursue two approaches, one based on

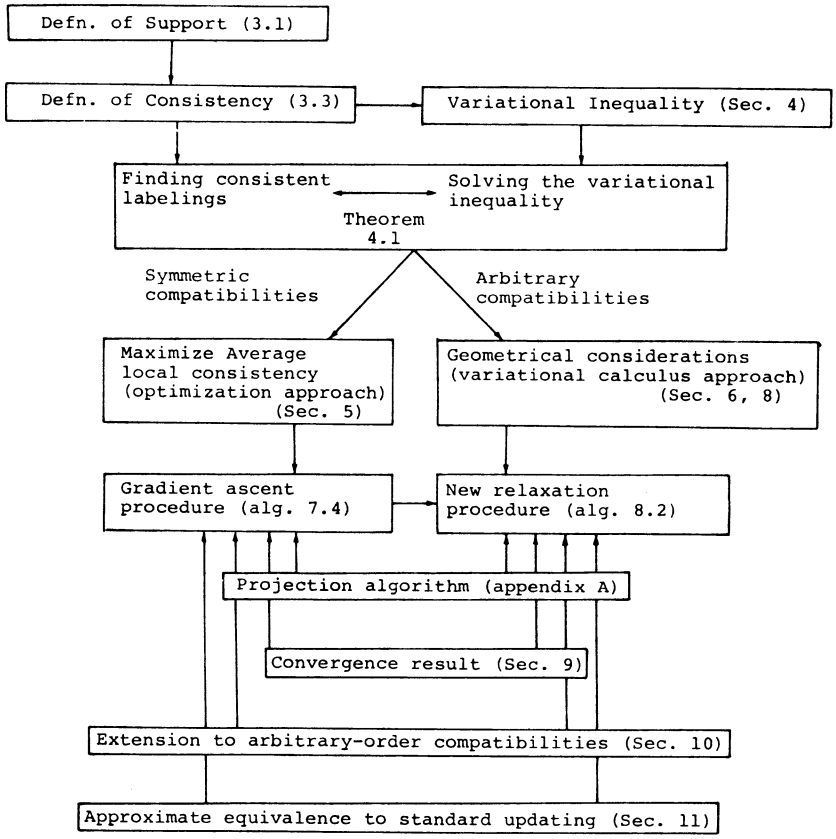


Fig. 1. Organization of the paper.

optimization theory, and the other founded in finite variational calculus. Surprisingly, the two methods will lead to the same algorithm, although the optimization approach applies only to a restricted class of compatibility matrices.

Fundamental to the analytic study of consistency is Theorem 4.1, which shows that achieving consistency is equivalent to solving a variational inequality. This inequality is important because it suggests an algorithm for finding solutions. Moreover, it will be shown that, in some cases, all local maxima of a certain functional on $\mathbb{K}$ solve the variational inequality. Thus one can find solutions, and hence consistent labelings, by finding local maxima.

In the following, we assume that consistency is defined (Definition 3.3) in terms of the linear support function $\bar{s}$ (Definition 3.1).

Theorem 4.1: A labeling $\bar{p} \in \mathbb{K}$ is consistent if and only if

$$\bar{p} \in \mathbb{K}: \sum_{i,\lambda,j,\lambda'} r_{ij}(\lambda,\lambda')\, p_j(\lambda')\, [v_i(\lambda) - p_i(\lambda)] \leqslant 0$$

for all $\bar{v} \in \mathbb{K}$.

Proof: If $\bar{p}$ is consistent, then

$$\sum_{\lambda} p_i(\lambda)\, s_i(\lambda) \geqslant \sum_{\lambda} v_i(\lambda)\, s_i(\lambda)$$

for any $\bar{v} \in \mathbb{K}$. But $s_i(\lambda) = \Sigma_{j,\lambda'} r_{ij}(\lambda,\lambda')\, p_j(\lambda')$. Substituting, and summing over i, one obtains the above variational inequality.

Conversely, if the variational inequality is solved by $\bar{p}$, then, for any $\bar{v} \in \mathbb{K}$, fix k, $1 \leqslant k \leqslant n$, and set

$$\bar{v}' = (\bar{p}_1, \cdots, \bar{v}_k, \cdots, \bar{p}_n).$$

Then $\bar{v}' \in \mathbb{K}$, and so

$$\sum_{i,\lambda,j,\lambda'} r_{ij}(\lambda,\lambda')\, p_j(\lambda')\, [v_i'(\lambda) - p_i(\lambda)]$$
$$= \sum_{i,\lambda} s_i(\lambda)\, [v_i'(\lambda) - p_i(\lambda)]$$
$$= \sum_{\lambda} s_k(\lambda)\, [v_k(\lambda) - p_k(\lambda)] \leqslant 0.$$

Since k is arbitrary, the above expression implies that $\bar{p}$ is consistent. □

Later, in Section VIII, we will derive another equivalent formulation of consistency. This final formulation is in terms of the stopping criteria for algorithms that solve the variational inequality in Theorem 4.1.

The study of consistency, and the derivation of algorithms for achieving it, will be pursued along two paths. These paths, as well as the organization of the rest of the paper, are illustrated in Fig. 1. One path involves showing how the variational inequality can be solved by an iterative procedure, as indicated by the right path in the diagram. Since this is a rather formal procedure, we shall postpone it to consider the case of symmetric compatibility matrices first (the left path through Fig. 1). This assumption allows us to structure the derivation more intuitively, in that it allows us to prove that maximizing a natural measure of the consistency of a labeling also yields solutions to the variational inequality. This maximization,

which is posed as a gradient ascent algorithm, leads to a form of relaxation labeling with a modified updating formula (see Section XI for a comparison of this new iterative procedure with the more standard ones). Finally, in Section X, we show how higher (and lower) order support functions modify the theory.

V. Average Local Consistency

The notion of consistency defined in Section III suggests a measure useful for guiding the updating of a nearly consistent labeling into a consistent one. Specifically, the n values $\{\Sigma_\lambda\, p_i(\lambda)\, s_i(\lambda)\}$ should each be large. Thus,

$$A(\bar{p}) = \sum_{i=1}^{n} \sum_{\lambda} p_i(\lambda)\, s_i(\lambda)$$

should also be large. We will refer to $A(\bar{p})$ as the *average local consistency*, because each of the terms in the sum represents the local consistency of $\bar{p}$ from the viewpoint of an object/label, weighted by the labeling weight. In the case when $s_i(\lambda)$ is given by a linear sum of assignment values as in Definition 3.1, we note that

$$A(\bar{p}) = \sum_{i,\lambda} \sum_{j,\lambda'} r_{ij}(\lambda, \lambda')\, p_i(\lambda)\, p_j(\lambda').$$

In this case, $A(\bar{p})$ is proportional to the average of the local consistency of the labeling $\bar{p}$ based on the pair of object labels (i, λ) and (j, λ'), averaged over all such pairs.

Since the average local consistency $A(\bar{p})$ should be large, it would seem natural to attempt to maximize it. Two problems deserve immediate comment. First, maximizing a sum does not necessarily maximize each of the individual terms. Secondly, the individual components $s_i(\lambda)$ depend on $\bar{p}$ (which varies during the maximization process), whereas consistency occurs when $\Sigma\, v_i(\lambda)\, s_i(\lambda; \bar{p})$ is maximized by $\bar{v} = \bar{p}$. That is, the $s_i(\lambda)$ should be fixed during the maximization. In summary, maximizing $A(\bar{p})$ is the same as maximizing

$$\sum_i \sum_\lambda p_i(\lambda)\, s_i(\lambda; \bar{p})$$

which is not the same as maximizing the n quantities

$$\sum_\lambda p_i(\lambda)\, s_i(\lambda; \bar{p}), \qquad i = 1, \cdots, n.$$

In fact, since the n quantities are not independent, there is no such thing as the problem of finding a single $\bar{p}$ which maximizes these n values simultaneously. However, finding consistent labelings $\bar{p}$ is a real problem, corresponding to the statement that each of the n values

$$\sum_\lambda v_i(\lambda)\, s_i(\lambda; \bar{p}), \qquad i = 1, \cdots, n.$$

are independently maximized among $\bar{v} \in \mathbb{K}$ when $\bar{v} = \bar{p}$.

Despite these caveats, we still find it interesting to study average local consistency. This is because of the independent interest of average local consistency, and because of the following theorem.

Theorem 5.1: Suppose that the matrix of compatibilities $\{r_{ij}(\lambda, \lambda')\}$ is symmetric, i.e.,

$$r_{ij}(\lambda, \lambda') = r_{ji}(\lambda', \lambda) \qquad \text{for all } i, j, \lambda, \lambda'.$$

If $A(\bar{p})$ attains a local (relative) maximum at $\bar{p} \in \mathbb{K}$, then $\bar{p}$ is a consistent labeling.

Proof: Let $\bar{v} \in \mathbb{K}$. Since $\mathbb{K}$ is convex, $t\bar{v} + (1 - t)\bar{p} \in \mathbb{K}$ for $0 \leqslant t \leqslant 1$. Since $A(\bar{p})$ does not increase as one moves away from $\bar{p}$

$$\left.\frac{d}{dt}\right|_{t=0} A(t\bar{v} + (1 - t)\bar{p}) \leqslant 0.$$

Thus grad $A(\bar{p}) \cdot (\bar{v} - \bar{p}) \leqslant 0$, by the chain rule. Using

$$A(\bar{p}) = \sum \sum r_{ij}(\lambda, \lambda')\, p_i(\lambda)\, p_j(\lambda'),$$

a calculation which is given later in this section shows that the (i, λ) component, $q_i(l)$, of grad $A(\bar{p})$ is given by

$$q_i(\lambda) = \sum_{j,\lambda'} [r_{ij}(\lambda, \lambda') + r_{ji}(\lambda', \lambda)]\, p_j(\lambda').$$

Since the $r_{ij}(\lambda, \lambda')$'s are symmetric,

$$q_i(\lambda) = 2 \sum_{j,\lambda'} r_{ij}(\lambda, \lambda')\, p_j(\lambda').$$

Thus grad $A(\bar{p}) \cdot (\bar{v} - \bar{p}) \leqslant 0$ implies that

$$2 \sum_{i,\lambda} \sum_{j,\lambda'} r_{ij}(\lambda, \lambda')\, p_j(\lambda')\, (v_i(\lambda) - p_i(\lambda)) \leqslant 0.$$

This holds true for all $\bar{v} \in \mathbb{K}$, so $\bar{p}$ satisfies the variational inequality of Theorem 4.1. According to Theorem 4.1, $\bar{p}$ is therefore a consistent labeling. □

Theorem 5.1 is surprising in light of all of the caveats about maximizing average local consistency, and its apparent dissimilarity with finding consistent labelings. Even so, in the special case when the compatibility matrix is symmetric, maximizing $A(\bar{p})$ leads to consistent labeling assignments!

In the general case when the compatibility matrix is not symmetric, local maxima of $A(\bar{p})$ still exist, and may be of limited interest, but will not be consistent labeling assignments in the sense of Definition 3.3. In Section VIII, we present an algorithm which generally leads to consistent labelings regardless of whether the compatibilities are symmetric or not. However, if the compatibilities happen to be symmetric, the algorithm in Section VIII is equivalent to finding local maxima of average local consistency, which, according to our previous theorem, are consistent labelings.

The proof of Theorem 5.1 makes clear the need for the symmetry condition. In general, local maxima of $A(\bar{p})$ will satisfy a variational inequality (as in Theorem 4.1) in which the $r_{ij}(\lambda, \lambda')$ terms are replaced by symmetrical terms $(r_{ij}(\lambda, \lambda') + r_{ji}(\lambda', \lambda))$:

$$\sum_{i,\lambda} \sum_{j,\lambda'} (r_{ij}(\lambda, \lambda') + r_{ji}(\lambda', \lambda))\, p_j(\lambda') \cdot (v_i(\lambda) - p_i(\lambda)) \leqslant 0$$

for all $\bar{v} \in \mathbb{K}$. That is, a local maximum of $A(\bar{p})$ will be a consistent labeling with respect to a symmetrized matrix of compatibilities. If one begins with a nonsymmetric matrix of compatibilities, and then locally maximizes $A(\bar{p})$, the result

is the same as if the original matrix had been symmetrized. Indeed,

$$A(\bar{p}) = \sum r_{ij}(\lambda, \lambda')\, p_i(\lambda)\, p_j(\lambda')$$
$$= \sum \tfrac{1}{2}\left(r_{ij}(\lambda, \lambda') + r_{ji}(\lambda', \lambda)\right) p_i(\lambda)\, p_j(\lambda').$$

Clearly, the calculation of $A(\bar{p})$ discards any information contained in the nonsymmetry of the compatibility matrix. We emphasize, however, that in Section VIII we present a technique for finding consistent labelings that preserves the information in the nonsymmetric (skew) part of the compatibility matrix.

A classical approach to finding local maxima of a smooth functional $A(\bar{p})$ is by means of gradient ascent [9]. The method prescribes that we should successively move the current position $\bar{p}$ by a small step to a new position $\bar{p}'$, so that the functional $A(\bar{p})$ is increased as much as possible. The amount of increase in $A(\bar{p})$ is related to the directional derivative of A in the direction of the step, which in turn is related to the gradient of A at $\bar{p}$. As promised in the proof of Theorem 5.1, we now compute grad $A(\bar{p})$. Since $\bar{p} \in \mathbb{K} \subseteq \mathbb{R}^{nm}$, the gradient, which we will denote by $\bar{q}$, contains nm components. The (i, λ) component is given by

$$q_i(\lambda) = \frac{\partial}{\partial p_i(\lambda)}\left[\sum_{\alpha,l}\sum_{\beta,l'} r_{\alpha\beta}(l, l')\, p_\alpha(l)\, p_\beta(l')\right]$$
$$= \sum_{\alpha,l}\sum_{\beta,l'} \left[r_{\alpha\beta}(l, l')\, \delta_{i\alpha}\delta_{\lambda l}\, p_\beta(l') + r_{\alpha\beta}(l, l')\, p_\alpha(l)\, \delta_{i\beta}\delta_{\lambda l'}\right]$$
$$= \sum_{\beta,l'} r_{i\beta}(\lambda, l')\, p_\beta(l') + \sum_{\alpha,l} r_{\alpha i}(l, \lambda)\, p_\alpha(l)$$
$$= \sum_{j,\lambda'} r_{ij}(\lambda, \lambda')\, p_j(\lambda') + \sum_{j,\lambda'} r_{ji}(\lambda', \lambda)\, p_j(\lambda')$$
$$= \sum_{j,\lambda'} \left(r_{ij}(\lambda, \lambda') + r_{ji}(\lambda', \lambda)\right) p_j(\lambda').$$

As noted before, when the compatibilities are symmetric, this expression simplifies to

$$q_i(\lambda) = 2 \sum r_{ij}(\lambda, \lambda')\, p_j(\lambda').$$

The right-hand side of this formula is a familiar expression. Comparing with Definition 3.1, we see that $q_i(\lambda) = 2\, s_i(\lambda)$. Also, the $q_i(\lambda)$ components are essentially the same values that were used as an intermediate updating "direction" in the earlier treatments of relaxation labeling [12].

Our goal is now to find a consistent labeling nearby a given initial weighted labeling assignment. By Theorem 5.1, when the compatibilities are symmetric, this can be accomplished by locally maximizing $A(\bar{p})$. Ordinarily, gradient ascent proceeds by moving a small step in the direction of the gradient. In the case of the problem at hand, the labeling assignment $\bar{p}$ must be constrained to lie in $\mathbb{K}$, whereas grad $A(\bar{p})$ may "point out of the surface." Instead, the assignment $\bar{p}$ should be updated by moving a small step in the direction which maximizes the directional derivative. In Section VII, we consider the problem of maximizing the directional derivative, and describe the complete gradient ascent algorithm. Despite its restricted applicability to the case of symmetric compatibilities, the algorithm leads naturally to the more general case considered in Section VIII. Before presenting the gradient ascent algorithm, however, we address in Section VI the geometric configuration and terminology to be used when describing updating assignment labelings in $\mathbb{K}$.

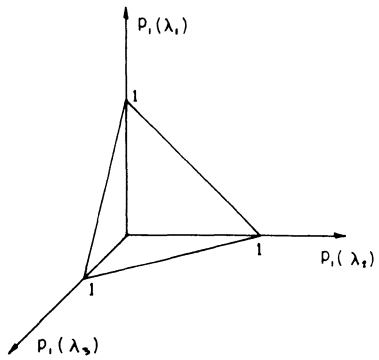


Fig. 2. The space of possible weighted labeling assignments for a single object with three labels.

VI. Geometric Structure of Assignment Space

To discuss gradient ascent on $\mathbb{K}$, and to visualize the more general updating algorithms, it is useful to consider the geometric structure of the weighted labeling space. The labeling space $\mathbb{K}$ was introduced in Section III as the convex hull of of the unambiguous labeling assignment space $\mathbb{K}^*$. In order to visualize the picture of the space $\mathbb{K}$, let us consider a simple example. Suppose that there are two objects, with three possible labels for each object. A labeling assignment then consists of six nonnegative numbers:

$$\bar{p} = (\bar{p}_1, \bar{p}_2) = (p_1(1), p_1(2), p_1(3), p_2(1), p_2(2), p_2(3))$$

satisfying

$$\sum_{\lambda=1}^{3} p_i(\lambda) = 1, \qquad i = 1, 2.$$

If we plot the locus of possible subvectors $\bar{p}_1$ in $\mathbb{R}^3$, the result is shown in Fig. 2. It consists of the portion of an affine subspace in the positive quadrant, has the shape of a simplex (a 2-simplex), and might be called "a probability space." We hasten to add, however, that the latter association with probabilities is particularly unhelpful. In any case, the vector $\bar{p} = (\bar{p}_1, \bar{p}_2)$ can be regarded as two points, each lying in a copy of the space shown in Fig. 2. Thus $\mathbb{K}$, for the particular example under consideration, can be identified with the set of all pairs of points in two copies of the triangular space in Fig. 2.

In more general situations, say with n objects each with m labels, $\mathbb{K}$ is more complicated. Now, the space consists of n copies of an $(m-1)$-simplex each formed from the positive quadrant portion of a flat $(m-1)$-dimensional affine subspace lying in $\mathbb{R}^m$. Then $\mathbb{K}$ can be identified as the set of all n-tuples of points, each point lying in a copy of the $(m-1)$-dimensional surface. A weighted labeling assignment is a point in the assignment space $\mathbb{K}$, and $\mathbb{K}$ is in turn the convex hull of the set of unambiguous labeling assignments $\mathbb{K}^*$. An unambiguous labeling also lies in $\mathbb{K}$, and can be thought of as

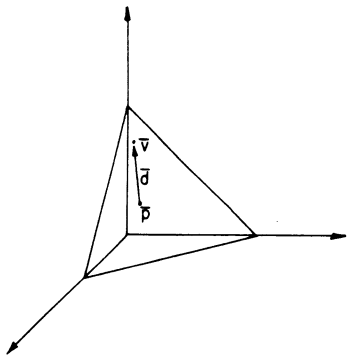


Fig. 3. A tangent vector $\bar{d}$ at a point $\bar{p} \in \mathbb{K}$.

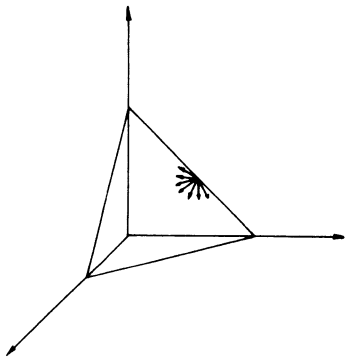


Fig. 4. The set of tangent directions at a point $\bar{p} \in \mathbb{K}$ that lies on a boundary.

one of the "corners," or extreme points, of the set. As an n-tuple of points, each lying in an $(m-1)$-simplex, an unambiguous assignment is composed of points which lie at vertices of their respective surfaces. Of course, any combination of n vertices gives rise to an unambiguous labeling—it is not necessary that each point represent the same vertex. In fact, each simplex has m corners, corresponding to the m possible labels for that particular object.

In differential geometry, the tangent space to a point of a multidimensional surface has a well defined meaning in terms of the set of all directions, in a limiting sense, along which a curve can move away from the given point. The tangent space is a surface, which when placed at the given point, lies "tangent" to the entire surface. For the assignment space $\mathbb{K}$, the initial surface $\mathbb{K}$ and tangent space at any interior point are flat, and so coincide when the tangent space is placed at its base point. The tangent space to a point in the interior of a surface is in fact a vector space. However, at a point of the boundary of a surface, the set of possible directions is restricted by the boundary, and one is forced to speak of the tangent set, which is simply a convex subset of a vector space.

More precisely, suppose that $\bar{p}$ is a labeling assignment in $\mathbb{K}$, and that $\bar{v}$ is any other assignment in $\mathbb{K}$. The difference vector $\bar{d} = \bar{v} - \bar{p}$, when placed at $\bar{p}$, points toward $\bar{v}$ (see Fig. 3). Thus $\bar{d}$ indicates a direction at $\bar{p}$ which points toward $\bar{v}$. Moreover, $\bar{d}$ and all positive scalar multiples of $\bar{d}$ are tangent vectors to $\mathbb{K}$ at $\bar{p}$. As $\bar{v}$ roams around $\mathbb{K}$, the set of all possible tangent directions at $\bar{p}$ is swept out. The set of all tangent vectors at $\bar{p}$ is therefore given by

$$T_{\bar{p}} = \{\bar{d}\colon\ \bar{d} = \alpha(\bar{v} - \bar{p}),\ \bar{v} \in \mathbb{K},\ \alpha \geqslant 0\}.$$

Note that any tangent vector is composed of n subvectors $\bar{d}_i$, so that $\bar{d} = (\bar{d}_1, \cdots, \bar{d}_n)$, and

$$\sum_{\lambda=1}^{m} d_i(\lambda) = \sum_{\lambda=1}^{m} \alpha(v_i(\lambda) - p_i(\lambda))$$

$$= \alpha \cdot (1 - 1) = 0.$$

When $\bar{p}$ is a point in the interior of $\mathbb{K}$ (i.e., no components are zero), the vector $\bar{v}$ may be chosen from a neighborhood that completely surrounds $\bar{p}$ in the n copies of the affine subspace. The result is that the set of tangent vectors at the interior point $\bar{p}$ consists of an entire subspace, which is given by

$$T_{\bar{p}} = \left\{\bar{d} = (\bar{d}_1, \cdots, \bar{d}_n)\colon\ \bar{d}_i \in \mathbb{R}^m,\ \sum_{\lambda=1}^{m} d_i(\lambda) = 0\right\}$$

($\bar{p}$ interior to $\mathbb{K}$).

Observe that $T_{\bar{p}}$ and $\mathbb{K}$ are parallel flat surfaces.

When $\bar{p}$ lies on a boundary of $\mathbb{K}$, the tangent set is a proper subset of the above space $T_{\bar{p}_o}$, for any interior point $\bar{p}_o$. That is, when the assignment $\bar{p}$ has some zero components, the set of vectors of the form $\alpha \cdot (\bar{v} - \bar{p})$ is restricted to

$$T_{\bar{p}} = \left\{\bar{d} = (\bar{d}_1, \cdots, \bar{d}_n)\colon\ \bar{d}_i \in \mathbb{R}^m,\ \sum_{\lambda=1}^{m} d_i(\lambda) = 0,\right.$$

$$\left.\text{and}\ \ d_i(\lambda) \geqslant 0\ \ \text{if}\ \ p_i(\lambda) = 0\right\}.$$

Fig. 4 shows a set of tangent directions that can arise from a boundary point in a single object, three label, assignment space. The tangent set $T_{\bar{p}}$ changes as $\bar{p}$ moves from one boundary edge to another. However, the tangent space $T_{\bar{p}}$ is always the same when $\bar{p}$ is an interior point.

VII. Maximizing Average Local Consistency

Now we can return to the problem of finding relative maxima of average local consistency using gradient ascent. We reiterate that maximizing $A(\bar{p})$ corresponds to finding a consistent labeling when the constraints are symmetric (see Section V). A different analysis must be applied when the constraints are not symmetric, but leads to essentially the same algorithm (Section VIII).

The increase in $A(\bar{p})$ due to a small step of length α in the direction $\bar{u}$ is approximately the directional derivative:

$$A(\bar{p} + \alpha\bar{u}) - A(\bar{p}) \approx \left.\frac{d}{dt}\right|_{t=0} A(\bar{p} + t\alpha\bar{u}) = \text{grad}\, A(\bar{p}) \cdot \alpha\bar{u}$$

where $\|u\| = 1$. In general, the greatest increase in $A(\bar{p})$ can be expected if a step is taken in the tangent direction $\bar{u}$ which maximizes the directional derivative. However, if the directional derivative is negative or zero for all nonzero tangent directions, then $A(\bar{p})$ is a local maximum and no step should be taken.

Accordingly, to find a direction of steepest ascent, grad $A(\bar{p}) \cdot \bar{u}$ should be maximized among the set of tangent

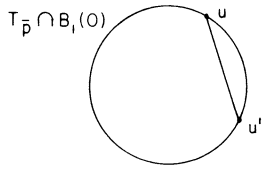


Fig. 5. Convex combinations of two distinct solutions $\bar{u}$ and $\bar{u}'$ lie on the straight line joining them.

vectors. However, it suffices to consider only those tangent vectors with Euclidean norm $\|\bar{u}\| = 1$ together with $\bar{u} = 0$. If unrestricted vectors are included in the maximization, then the problem has no solution since the vector dot product grad $A(\bar{p}) \cdot \bar{u}$ is scaled by $\|\bar{u}\|$. The problem is equivalent to maximizing $\bar{q} \cdot \bar{u}$ among tangent vectors with $\|\bar{u}\| \leqslant 1$, where $\bar{q} = \frac{1}{2}$ grad $A(\bar{p})$. Thus the direction of steepest ascent can be found by solving the following.

Problem 7.1: Find $\bar{u} \in T_{\bar{p}} \cap B_1(0)$ such that

$$\bar{u} \cdot \bar{q} \geqslant \bar{v} \cdot \bar{q} \qquad \text{for all } \bar{v} \in T_{\bar{p}} \cap B_1(0).$$

Here $B_1(0) = \{\bar{v} \in \mathbb{R}^{nm} : \|\bar{v}\| \leqslant 1\}$. □

A solution to Problem 7.1 always exists because the problem requires one to maximize a continuous functional over a compact set. In many cases, the solution will be unique.

Observation 7.2: If $\bar{u}$ solves Problem 7.1, and if $\bar{u} \cdot \bar{q} > 0$, then $\bar{u}$ is the unique solution.

Proof: First, note that $\bar{u} \neq 0$, since $\bar{u} \cdot \bar{q} \neq 0$. Thus $\|u\| = 1$, since otherwise $\bar{w} = \bar{u}/\|\bar{u}\|$ is a vector in $T_{\bar{p}} \cap B_1(0)$ satisfying $\bar{u} \cdot \bar{q} < \bar{w} \cdot \bar{q}$, in contradiction to $\bar{u}$ being a solution. Suppose $\bar{u}'$ is any other solution. Then $\bar{u}' \cdot \bar{q} = \bar{u} \cdot \bar{q} \neq 0$, and so $\|\bar{u}'\| = 1$ for the same reason that $\|u\| = 1$. We will show that $\bar{u}' = \bar{u}$.

Suppose that $\bar{u}$ and $\bar{u}'$ are two *distinct* solutions. Then $\bar{u} \cdot \bar{q} = \bar{u}' \cdot \bar{q} = (t\bar{u} + (1 - t)\bar{u}') \cdot \bar{q}$ for all t, $0 \leqslant t \leqslant 1$ (see Fig. 5). Since $T_{\bar{p}} \cap B_1(0)$ is convex, all the vectors on the straight line from $\bar{u}$ to $\bar{u}'$ are solutions. But by the same argument given in the previous paragraph, all such solutions must lie on the surface of the unit ball $B_1(0)$. Since $\|\bar{u}\| = \|\bar{u}'\| = 1$, and $B_1(0)$ is strictly convex, the line between $\bar{u}$ and $\bar{u}'$ must lie in the interior of the ball, and so cannot contain any solutions. This contradicts the existence of two distinct solutions. □

The zero vector is always in $T_{\bar{p}} \cap B_1(0)$, so the maximum of $\bar{v} \cdot \bar{q}$ is nonnegative. When the maximum $\bar{u} \cdot \bar{q}$ is positive, then $\bar{u}$ is the unique solution. When the maximum $\bar{u} \cdot \bar{q} = 0$, the zero vector is among possibly many solutions. In this case, we will agree that $\bar{u} = 0$ is the best solution for Problem 7.1. In the solution method presented in Appendix A, the zero vector is always the solution returned by the algorithm when more than one solution is possible.

Conceptually, our method for finding relative maxima of average local consistency is very simple. Starting at an initial labeling $\bar{p}$, we compute $\bar{q} = \frac{1}{2}$ grad $A(\bar{p})$, and solve Problem 7.1. If the resulting $\bar{u}$ is nonzero, we take a small step in the direction $\bar{u}$, and repeat the process. The algorithm terminates when $\bar{u} = 0$. This is the algorithm given below (Algorithm 7.4). Clearly, however, we need a way to solve Problem 7.1 given a $\bar{p} \in \mathbb{K}$ and a $\bar{q} \in \mathbb{R}^{nm}$.

A simple, finite algorithm for solving Problem 7.1 is presented in Appendix A. When $\bar{p}$ is in the interior of the assignment space $\mathbb{K}$, solving Problem 7.1 is a triviality, corresponding to projecting $\bar{q}$ onto the tangent space $T_{\bar{p}}$, and then normalizing. Lemma 7.3, which follows, proves that this procedure works. When $\bar{p}$ is on a boundary of $\mathbb{K}$, the situation is considerably more complicated. Fortunately, the algorithm given in Appendix A handles all cases.

Lemma 7.3: If $\bar{p}$ lies in the interior of $\mathbb{K}$, then the following algorithm solves Problem 7.1:

1) set $c_i = \dfrac{1}{m} \sum_{l=1}^{m} q_i(l)$, $\qquad i = 1, \cdots, n.$

2) set $w_i(\lambda) = q_i(\lambda) - c_i$, $\qquad$ all i, λ,

3) set $u_i(\lambda) = w_i(\lambda)/\|\bar{w}\|$, $\qquad$ all i, λ.

Here

$$\|\bar{w}\| = \left[\sum_{i,\lambda} w_i(\lambda)^2\right]^{1/2}.$$

Proof: First observe that $\bar{u} \in T_{\bar{p}} \cap B_1(0)$, since $\|\bar{u}\| = 1$ and

$$\sum_{\lambda=1}^{m} u_i(\lambda) = \sum_{\lambda=1}^{m} (q_i(\lambda) - c_i)/\|\bar{w}\|$$

$$= \left[\sum_{\lambda=1}^{m} q_i(\lambda) - mc_i\right] \Big/ \|\bar{w}\| = 0, \qquad \text{for all } i.$$

Since $\bar{p}$ is in the interior of $\mathbb{K}$, membership in $T_{\bar{p}}$ requires no further conditions.

Next observe that $\bar{w}$ is the projection of $\bar{q}$ onto $T_{\bar{p}}$, i.e., $(\bar{q} - \bar{w}) \cdot \bar{v} = 0$ for all $\bar{v} \in T_{\bar{p}}$. To see this, we simply calculate

$$\sum_i \sum_\lambda (q_i(\lambda) - w_i(\lambda)) \cdot v_i(\lambda) = \sum_i c_i \sum_\lambda v_i(\lambda) = 0$$

(since $\bar{v} \in T_{\bar{p}}$). Thus $\bar{v} \cdot \bar{q} = \bar{v} \cdot \bar{w}$ for all $\bar{v} \in T_{\bar{p}}$. Since $\bar{u} \in T_{\bar{p}}$,

$$\bar{u} \cdot \bar{q} = \bar{u} \cdot \bar{w} = \frac{\bar{w}}{\|w\|} \cdot \bar{w} = \|\bar{w}\| \geqslant \|\bar{w}\| \cdot \|\bar{v}\|$$

for any $\bar{v} \in T_{\bar{p}} \cap B_1(0)$ (note that $\|\bar{v}\| \leqslant 1$). By the Cauchy-Schwarz inequality, $\bar{v} \cdot \bar{w} \leqslant \|\bar{v}\|\ \|\bar{w}\|$, so we have

$$\bar{u} \cdot \bar{q} \geqslant \bar{v} \cdot \bar{w} = \bar{v} \cdot \bar{q} \qquad \text{for all } \bar{v} \in T_{\bar{p}} \cap B_1(0).$$

That is, $\bar{u}$ solves Problem 7.1. □

The algorithm in Lemma 7.3 may fail when $\bar{p}$ is a boundary point of $\mathbb{K}$, since there is no guarantee that $w_i(\lambda) \geqslant 0$ when $p_i(\lambda) = 0$. It works out that Problem 7.1 is still solved by performing a projection of $\bar{q}$ followed by length normalization, but when $\bar{p}$ is a boundary point the projection onto $T_{\bar{p}}$ is a projection onto a convex set, and not onto a subspace (see the discussion of tangent sets in Section VI). The algorithm in Appendix A gives a method for computing this projection. However, the theory which shows that this method solves Problem 7.1, even when $\bar{p}$ is a boundary point, is rather involved, and is the topic of a companion paper [10].

Combining these results, we obtain the following algorithm for finding local maxima of $A(\bar{p})$.

Algorithm 7.4:

Initialize:

1) Start with an initial labeling assignment $\bar{p}^0 \in \mathbb{K}$. Set $k = 0$.

Loop until a stop is executed:

2) Compute $\bar{q}^k = \frac{1}{2}$ grad $A(\bar{p}^k)$.
3) Use the algorithm in Appendix A, with $\bar{p} = \bar{p}^k$, $\bar{q} = \bar{q}^k$, to find the solution $\bar{u}^k$ to Problem 7.1.
4) If $\bar{u}^k = 0$, stop.
5) Set $\bar{p}^{k+1} = \bar{p}^k + h\bar{u}^k$, where $0 < h \leq \alpha_k$ is determined so that $\bar{p}^{k+1} \in \mathbb{K}$. The maximum step size α_k is some predetermined small value, and may decrease as k increases to facilitate convergence.
6) Replace k by $k + 1$.

End loop. □

In summary, successive iterates are obtained by moving a small step in the direction of the projection of the gradient $\bar{q}$ onto the convex set of tangent directions $T_{\bar{p}}$. The algorithm stops when this projection is zero. Note also that the gradient, calculated in Step 2, yields the formula

$$q_i^k(\lambda) = \sum_j \sum_{\lambda'} r_{ij}(\lambda, \lambda') p_j^k(\lambda')$$

according to the calculation in Section V (assuming symmetric compatibilities).

Proposition 7.5: Suppose $\bar{p}$ is a stopping point of Algorithm 7.4. Then if the matrix of compatibilities is symmetric, $\bar{p}$ is consistent.

Proof: At a stopping point, $\bar{u} = 0$ solves Problem 7.1. Thus $\bar{v} \cdot \bar{q} \leq 0$ for all tangent vectors $\bar{v} \in T_{\bar{p}}$. If we choose any $\bar{v} \in \mathbb{K}$, then $\bar{v} - \bar{p}$ is a tangent vector, and so $(\bar{v} - \bar{p}) \cdot \bar{q} \leq 0$. Using the formula for $\bar{q}$,

$$\sum_{i,\lambda} \sum_{j,\lambda'} r_{ij}(\lambda, \lambda') p_j(\lambda') \cdot (v_i(\lambda) - p_i(\lambda)) \leq 0 \quad \text{for all } \bar{v} \in \mathbb{K}.$$

Thus, $\bar{p}$ satisfies the variational inequality of Section IV, and so by Theorem 4.1, $\bar{p}$ is consistent in $\mathbb{K}$. □

We now have a method for finding consistent labelings, given an initial labeling assignment. Whether the resulting consistent labeling is an improvement over the initial assignment depends upon the extent to which it makes sense to increase average local consistency.

The property that $\bar{p}$ yields a local maximum of the average local consistency is actually stronger than is required by the definition of consistency. In particular, while all local maxima of $A(\bar{p})$ are at consistent labelings (Theorem 5.1), the gradient ascent algorithm 7.4 may stop at one of a number of pathological points. This behavior can occur because, for symmetric compatibilities, consistency is a property that depends on the first derivatives of $A(\bar{p})$, whereas a local maximum must include second derivatives in its characterization. The pathologies can include local minima, saddle points, and boundary points with local minima or saddles [see Fig. 6(b) and (c)]. Note that if a relative maximum occurs at a boundary point, the gradient does not necessarily vanish [Fig. 6(d)].

In practice, the gradient ascent algorithm will generally find a local maximum of $A(\bar{p})$. The algorithm will stop at one of the pathological points only if one of the iterates happens to land exactly on such a point. However, all of these points, according to Proposition 7.5, are consistent.

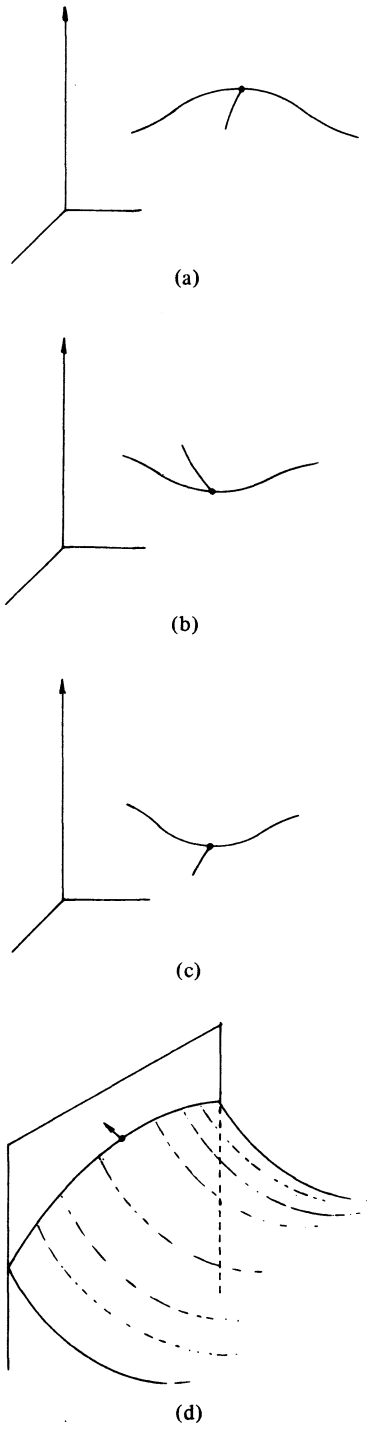


Fig. 6. Examples of stopping points: (a) local max, (b) local min, (c) saddle point, (d) local max at the boundary. Note that the gradient is nonzero at the stopping point in (d).

VIII. The Relaxation Labeling Algorithm

Algorithm 7.4 gives us a method of finding consistent labelings when the matrix of compatibilities is symmetric. Our entire analysis of average local consistency relies on the assumption of symmetric compatibilities. The assumption

is unreasonable. In general, we believe that compatibility coefficients need not be symmetric.

Consider, for example, the constraints between letters comprising words in the English language. The occurrence of a "q" in a word lends strong support to the labeling of the following letter as a "u," whereas the occurrence of a "u" only weakly supports the proposition that the previous letter is a "q."

Fortunately, our theory of consistency does not rely on symmetric compatibilities. The characterization of consistent labelings in Theorem 4.1 is completely general, valid whether or not the compatibilities are symmetric. The purpose of this section is to present an algorithm that finds consistent labelings based on the variational inequality in Theorem 4.1. Although the point of view is now much more general, the resulting algorithm is nearly identical to Algorithm 7.4! Further, both algorithms share some similarities with the ad hoc methods [12], as we discuss in Section XI. The algorithm presented in this section is based on a theory of consistency, and allows one to interpret what the relaxation labeling process accomplishes.

We begin by recalling the variational inequality for consistency:

$$\sum_{i,\lambda}\sum_{j,\lambda'} r_{ij}(\lambda,\lambda')\,p_j(\lambda')\,(v_i(\lambda)-p_i(\lambda)) \leqslant 0 \qquad \text{for all } \bar{v}\in \mathbb{K}$$

or, more generally,

$$\sum_{i,\lambda} s_i(\lambda;\bar{p})\cdot(v_i(\lambda)-p_i(\lambda)) \leqslant 0 \qquad \text{for all } \bar{v}\in \mathbb{K}.$$

In Section VII, we defined $\bar{q}=\frac{1}{2}\operatorname{grad} A(\bar{p})$, so that for symmetric compatibilities, we had

$$q_i(\lambda)=\sum_{j,\lambda'} r_{ij}(\lambda,\lambda')\,p_j(\lambda')=s_i(\lambda;\bar{p}).$$

Hereafter, we define $\bar{q}$ by

$$q_i(\lambda)=\sum_{j,\lambda'} r_{ij}(\lambda,\lambda')\,p_j(\lambda')$$

whether or not the compatibilities are symmetric. That is, we have set $\bar{q}=\bar{s}(\bar{p})$. It is important to realize, however, that $\bar{q}$ is not in general the gradient of any functional on $\mathbb{K}$.

Observation 8.1: With $\bar{q}$ defined as above, the variational inequality is equivalent to the statement

$$\bar{q}\cdot\bar{t}\leqslant 0 \qquad \text{for all } \bar{t}\in T_{\bar{p}}.$$

That is, a labeling $\bar{p}$ is consistent if and only if $\bar{q}$ points away from all tangent directions.

Proof: We have $\bar{q}=\bar{s}$, and any tangent vector $\bar{t}$ at $\bar{p}$ can be written as a positive scalar multiple of $\bar{v}-\bar{p}$, where $\bar{v}\in\mathbb{K}$. The observation follows immediately. □

Observation 8.1 suggests a way of finding a consistent labeling. If, at a labeling $\bar{p}$, the associated vector $\bar{q}$ points in the same direction as some tangent vector, then $\bar{p}$ is not consistent. So $\bar{p}$ should be moved in the direction of that tangent vector. This process may be repeated until $\bar{q}$ evaluated at the current assignment points away from all tangent directions. Then $\bar{p}$ will be a consistent labeling.

Note that $\bar{q}$ varies as $\bar{p}$ moves, but that generally $\bar{q}$ will change smoothly and gradually. Thus if $\bar{q}$ points away from the surface $\mathbb{K}$ at a vertex (and is therefore an unambiguous consistent labeling, then $\bar{q}$ will point generally toward the vertex at nearby assignments in $\mathbb{K}$. Accordingly, if $\bar{p}$ is near the unambiguous consistent labeling, moving $\bar{p}$ in a tangent direction $\bar{u}$ that points in the same direction as $\bar{q}$, should cause $\bar{p}$ to converge to the vertex (see Theorem 9.1). To present these ideas more formally, we begin by defining the algorithm.

If $\bar{q}\cdot\bar{t}>0$ for some tangent direction t, then the current assignment $\bar{p}$ is not consistent, and should be updated. In which direction should we move $\bar{p}$? In analogy with the gradient ascent algorithm it makes sense to move $\bar{p}$ in the direction $\bar{u}$ that maximizes $\bar{q}\cdot\bar{u}$. This is exactly the vector $\bar{u}$ returned by the algorithm of Appendix A as the solution to Problem 7.1. This method of updating $\bar{p}$ is identical to the gradient ascent method (Algorithm 7.4), applicable for symmetric compatibilities, except that $\bar{q}$ has a new conceptual meaning. Instead of setting $\bar{q}=\frac{1}{2}\operatorname{grad} A(\bar{p})$ in Step 2, we set $\bar{q}=\bar{s}(\bar{p})$. If the gradient ascent method is applied only when the compatibilities are symmetric, and providing $\bar{s}(\bar{p})$ is linear, the formulas are identical.

Accordingly, the relaxation labeling algorithm is given by the following.

Algorithm 8.2: Replace Step 2 in Algorithm 7.4 with:

2′) Compute $\bar{q}=\bar{s}(\bar{p})$. That is,

$$q_i(\lambda)=\sum_{j,\lambda'} r_{ij}(\lambda,\lambda')\,p_j(\lambda').$$

All other steps remain the same. □

Compare the following result with Proposition 7.5.

Proposition 8.3: Suppose $\bar{p}$ is a stopping point of Algorithm 8.2. Then $\bar{p}$ is consistent.

Proof: A point $\bar{p}$ is a stopping point of Algorithm 8.2 if and only if $\bar{u}=0$ solves Problem 7.1. If $\bar{u}=0$, then $\bar{t}\cdot\bar{q}\leqslant\bar{0}\cdot\bar{q}=0$ for all tangent vectors $\bar{v}\in T_{\bar{p}}$. On the other hand, if $\bar{t}\cdot\bar{q}\leqslant 0$ for all $\bar{t}\in T_{\bar{p}}$, then $\bar{u}=0$ maximizes $\bar{u}\cdot\bar{q}$ for $\bar{u}\in T_{\bar{p}}\cap B_1(0)$. According to Observation 8.1, $\bar{t}\cdot\bar{q}\leqslant 0$ for all $\bar{t}\in T_{\bar{p}}$ is equivalent to the variational inequality, which is in turn equivalent to $\bar{p}$ being consistent (Theorem 4.1). □

At this point, we have introduced a definition of consistency and presented the relaxation labeling algorithm in such a way that the stopping points of the algorithm are consistent labelings. A number of questions remain to be answered. First, are there any consistent labelings for the relaxation labeling algorithm to find? Second, assuming that such points exist, will the algorithm find them? And finally, even if a relaxation labeling process converges to a consistent labeling, is the final labeling better than the initial assignment?

The first question is answered affirmatively by Proposition 8.4 below. The second question is far more subtle, and is substantially answered by a local convergence result in Section IX. The third question, concerning the significance of the final result, is not really well defined. If the compatibilities are symmetric, then the functional $A(\bar{p})$ provides a quantitative measure of how much the consistent labeling improves the

initial, inconsistent one. In the very special case that $A(\bar{p})$ is convex, then the final labeling will be a global maximum. But if the compatibilities are not symmetric, then no such functional exists. The only statement that can be made is that the labeling is now consistent with respect to the compatibility relations; it may be substantially different from the initial labeling.

In general, the space of consistent labelings will be quite rich. Consider, once again, the labeling of letters comprising words in the English language. If we are given five letters, then every five letter word constitutes a consistent labeling. Thus, if an initial estimate is derived from realistic measurements, there will normally be a consistent labeling nearby to which the process will converge.

Proposition 8.4: The variational inequality of Theorem 4.1 always has at least one solution. Thus consistent labelings always exist, for arbitrary compatibility matrices.

Proof: We invoke a version of the Brouwer fixed point theorem, due to Stampacchia [8]. The result states that if $\mathbb{K} \subseteq \mathbb{R}^N$ is a compact convex set, and if $\bar{F}(\bar{x})$ is a continuous function from $\mathbb{K}$ into $\mathbb{R}^N$, then there exists an $\bar{x} \in \mathbb{K}$ such that $\bar{F}(\bar{x}) \cdot (\bar{y} - \bar{x}) \leqslant 0$ for all $\bar{y} \in \mathbb{K}$. For the variational inequality of Theorem 4.1, we set

$$(F(\bar{p}))_i(\lambda) = -\sum_{j,\lambda'} r_{ij}(\lambda, \lambda')\, p_j(\lambda').$$

Note that F is a continuous (linear, in fact), and that $\mathbb{K}$ is a compact and convex subset of $\mathbb{R}^{nm}$. Thus Stampacchia's result applies, and the proof is complete. □

Usually, more than one solution will exist, however, which is desirable for most relaxation labeling applications. It is only in highly specialized situations, such as when the matrix of compatibilities is negative definite, that the solution is unique.

IX. A Local Convergence Result

As the step size of the relaxation labeling algorithm 7.4 or 8.2 becomes infinitesimal, these discrete algorithms approximate dynamical systems [3]. The iterates $\bar{p}^k$ become a parameterized curve $\bar{p}(t)$, $t \in \mathbb{R}$, lying in $\mathbb{K}$. The tangent to the curve at every point is the updating direction $\bar{u}$, where $\bar{u}$ is the solution to Problem 7.1 at that point. Note that $\bar{u}$ is the normalized projection of $\bar{q}$, which in turn is computed from the matrix of compatibilities and the current labeling assignment $\bar{p}$. Thus the dynamical system obeys a differential equation

$$\frac{d}{dt}\bar{p}(t) = \bar{u}(\bar{p}(t)).$$

Inside the interior of $\mathbb{K}$, $\bar{u}(\bar{p})$ is a linear function of $\bar{p}$. As $\bar{p}$ moves from the interior to a boundary point of $\mathbb{K}$, $\bar{u}(\bar{p})$ may be discontinuous. For this reason, the dynamical system may be very complex.

The main mathematical result about relaxation labeling in this paper concerns the convergence of the above dynamical system. We already know that the dynamical system has a stopping point at $\bar{p}$ if and only if $\bar{p}$ is a consistent labeling (i.e., $\bar{u}(\bar{p}) = 0$). Thus, if the dynamical system, which is approximated by the relaxation labeling algorithm, converges to a stopping point, then we have found a consistent labeling. However, in general, a dynamical system need not converge. Even though $\bar{p}(t)$ always lies in the compact assignment space $\mathbb{K}$, the process might approach an infinite cycle (an orbit), or, it might approach an Ω-limit set that forms a toroidal shape or higher dimensional set. We cannot guarantee that orbits or other examples of nonconvergence will never happen. However, we claim that by using the proper updating rule (Appendix A) and reasonable compatibility values, such behavior is extremely unlikely. Theorem 9.1 argues in justification of this claim.

Recall that a labeling is strictly consistent if

$$\sum_{\lambda} p_i(\lambda)\, s_i(\lambda) > \sum_{\lambda} v_i(\lambda)\, s_i(\lambda), \qquad i = 1, \cdots, n$$

whenever $\bar{v} \neq \bar{p}$, $\bar{v} \in \mathbb{K}$. As a result, the variational inequality can be replaced by the statement

$$\sum_{i,\lambda,j,\lambda'} r_{ij}(\lambda, \lambda')\, p_j(\lambda')\,(v_i(\lambda) - p_i(\lambda)) < 0$$

$$\text{for all } \bar{v} \in \mathbb{K},\ \bar{v} \neq \bar{p}$$

for a strictly consistent labeling. In particular, $\bar{q} \cdot \bar{u} < 0$ for all nonzero tangent directions $\bar{u}$ at a strictly consistent labeling $\bar{p}$. We claim that $\bar{p} \in \mathbb{K}^*$ (i.e., that $\bar{p}$ is an unambiguous labeling). Suppose, for contradiction, that $0 < p_{i_o}(\lambda_o) < 1$ for some (i_o, λ_o). Then for some other λ_o', $0 < p_{i_o}(\lambda_o') < 1$. We consider two tangent directions,

$$u_1(i, \lambda) = \begin{cases} 0 & i \neq i_o \\ (0, \cdots, 0, 1, \cdots, -1, \cdots, 0) & i = i_o, \end{cases}$$

and $\bar{u}_2 = -\bar{u}_1$.

That is, $\bar{u}_1$ has a 1 in the (i_o, λ_o) position and a -1 in the (i_o, λ_o') position, and $\bar{u}_2$ is the other way around. These are valid tangent directions according to the formulation of $T_{\bar{p}}$ in Section VI. However, $\bar{q} \cdot \bar{u}_1 = -\bar{q} \cdot \bar{u}_2$, so they cannot both be negative. Hence, we have shown that a strictly consistent labeling $\bar{p}$ must be unambiguous.

Our main result is the following.

Theorem 9.1: Suppose $\bar{e} \in \mathbb{K}^*$ is strictly consistent. Then there exists a neighborhood of $\bar{e}$ such that if $\bar{p}(t)$ enters the neighborhood, then $\lim_{t \to \infty} \bar{p}(t) = \bar{e}$.

In fact, once $\bar{p}(t)$ enters the neighborhood, $\bar{p}(t) \equiv \bar{e}$ after a finite length of time.

Proof: We will make use of the Euclidean norm

$$\|\bar{v}\| = \left[\sum_{i,\lambda} (v_i(\lambda))^2\right]^{1/2}$$

for vectors in $\mathbb{R}^{nm}$. Our first task is to show that if $\bar{p} \in \mathbb{K}$, $\bar{p} \neq \bar{e}$, then there is an assignment $\bar{p}' \in \mathbb{K}$ such that

$$\bar{p}' - \bar{e} = \frac{\bar{p} - \bar{e}}{\|\bar{p} - \bar{e}\|}.$$

We begin by setting $\alpha = \|\bar{p} - \bar{e}\|$, and define

$$\bar{p}' = \left(1 - \frac{1}{\alpha}\right)\bar{e} + \frac{1}{\alpha}\bar{p}.$$

If $\alpha \geq 1$, then $\bar{p}' \in \mathbb{K}$ since $\mathbb{K}$ is convex. (Here we use the convexity of $\mathbb{K}$ in a crucial way!) If $\alpha \leq 1$, a separate argument is needed. Clearly,

$$\sum_{\lambda} p_i'(\lambda) = 1, \quad i = 1, \cdots, n.$$

It remains to be shown that $0 \leq p_i'(\lambda) \leq 1$ when $\alpha \leq 1$, in order to conclude that $\bar{p}' \in \mathbb{K}$. Suppose that $e_i(\lambda) = 0$. Then since $\alpha = \|\bar{p} - \bar{e}\| \geq |p_i(\lambda) - e_i(\lambda)| = p_i(\lambda)$, we have $p_i(\lambda)/\alpha \leq 1$. But $p_i'(\lambda) = p_i(\lambda)/\alpha$, so $0 \leq p_i'(\lambda) \leq 1$. Next suppose that $e_i(\lambda) = 1$. Then $\alpha \geq 1 - p_i(\lambda)$, and so

$$p_i'(\lambda) = (1 - 1/\alpha) + p_i(\lambda)/\alpha = \frac{\alpha - 1 + p_i(\lambda)}{\alpha}$$

satisfies $p_i'(\lambda) \geq 0$. Further, since $p_i(\lambda) \leq 1$, therefore $\alpha - 1 + p_i(\lambda) \leq \alpha$, and so $p_i'(\lambda) \leq 1$. The conclusion is that $\bar{p}' \in \mathbb{K}$, and satisfies the required equation.

Next, according to the variational inequality,

$$\sum_{i,\lambda} \sum_{j,\lambda'} r_{ij}(\lambda, \lambda')\, e_j(\lambda')\, (p_i(\lambda) - e_i(\lambda)) < 0$$

for any $\bar{p} \in \mathbb{K}$, $\bar{p} \neq \bar{e}$. In particular, the left-hand side has a negative maximum on any compact subset of $\mathbb{K}$ not including $\bar{e}$. Thus there exists a $\beta < 0$ such that

$$\bar{q}(\bar{e}) \cdot (\bar{p}' - \bar{e}) \leq \beta < 0,$$

for all $\bar{p}' \in \mathbb{K}$ such that $\|\bar{p}' - \bar{e}\| = 1$.

Here $\bar{q}(\bar{p}) = \bar{q}$, where

$$q_i(\lambda) = \sum_{j,\lambda'} r_{ij}(\lambda, \lambda')\, p_j(\lambda').$$

Now suppose that $\bar{p} \in \mathbb{K}$, $\bar{p} \neq \bar{e}$, and $\bar{p}'$ is the corresponding assignment in $\mathbb{K}$ satisfying

$$\bar{p}' - \bar{e} = \frac{\bar{p} - \bar{e}}{\|\bar{p} - \bar{e}\|}.$$

Note that $\|\bar{p}' - \bar{e}\| = 1$, so that

$$\bar{q}(\bar{e}) \cdot \frac{(\bar{p} - \bar{e})}{\|\bar{p} - \bar{e}\|} \leq \beta < 0.$$

Since $\bar{q}(\bar{p})$ is a continuous function, we have $\|\bar{q}(\bar{p}) - \bar{q}(\bar{e})\| \leq |\beta|/2$ for $\bar{p}$ in a neighborhood of $\bar{e}$, and thus

$$\bar{q}(\bar{p}) \cdot \frac{(\bar{p} - \bar{e})}{\|\bar{p} - \bar{e}\|} \leq \beta/2 < 0$$

for $\bar{p} \in \mathbb{K}$ in the neighborhood of $\bar{e}$.

If $\bar{p}$ is in the interior of $\mathbb{K}$, then $\bar{u}(\bar{p}) = \bar{w}/\|\bar{w}\|$ where $\bar{w}$ is the projection of $\bar{q}$ onto $T_{\bar{p}}$. By the projection theorem, $(\bar{q} - \bar{w}) \cdot \bar{t} = 0$ for any tangent vector $\bar{t} \in T_{\bar{p}}$. Thus $\bar{q} \cdot (\bar{p} - \bar{e}) = \bar{w} \cdot (\bar{p} - \bar{e})$, and so for $\bar{p}$ in the interior of $\mathbb{K}$ sufficiently near $\bar{e}$, $\bar{p} \neq \bar{e}$,

$$\bar{u}(\bar{p}) \cdot \frac{(\bar{p} - \bar{e})}{\|\bar{p} - \bar{e}\|} = \frac{1}{\|w\|} \cdot \bar{q}(\bar{p}) \cdot \frac{(\bar{p} - \bar{e})}{\|\bar{p} - \bar{e}\|} \leq \frac{\beta}{2\|w\|}.$$

Further, since $\bar{w}$ is the projection of $\bar{q}$, $\|\bar{w}\| \leq \|\bar{q}(\bar{p})\|$. Set

$$\gamma = \max_{\bar{p} \in \mathbb{K}} \|\bar{q}(\bar{p})\|$$

whence $\|w\| \leq \gamma$. Since β is negative,

$$\bar{u}(\bar{p}) \cdot \frac{(\bar{p} - \bar{e})}{\|\bar{p} - \bar{e}\|} \leq \frac{\beta}{2\gamma} < 0.$$

We will next show that the same inequality holds if $\bar{p}$ is on a boundary of $\mathbb{K}$, but sufficiently near $\bar{e}$.

According to Appendix A, the vector $\bar{u}$ at $\bar{p}$ is obtained by normalizing a vector $\bar{w}$, which in turn is obtained from $\bar{q} = \bar{q}(\bar{p})$ by setting some of the components to zero, and subtracting constants from the other components: $w_i(\lambda) = q_i(\lambda) - c_i$. The indexes (i, λ) of the components which are set to zero are a subset of the set for which $p_i(\lambda) = 0$. Denote this subset of indexes by S. Since $\bar{p}$ is near $\bar{e}$, the components of $\bar{e}$ are close to the components of $\bar{p}$, so we must have $e_i(\lambda) = 0$ whenever $p_i(\lambda) = 0$. Hence $p_i(\lambda) - e_i(\lambda) = 0$ for $(i, \lambda) \in S$. Thus,

$$\begin{aligned}
\bar{w} \cdot (\bar{p} - \bar{e}) &= \sum_{(i,\lambda) \notin S} w_i(\lambda) \cdot (p_i(\lambda) - e_i(\lambda)) \\
&= \sum_{(i,\lambda) \notin S} (q_i(\lambda) - c_i)\, (p_i(\lambda) - e_i(\lambda)) \\
&= \sum_{i,\lambda} q_i(\lambda) \cdot (p_i(\lambda) - e_i(\lambda)) \\
&\quad - \sum_{i=1}^{n} c_i \sum_{\lambda=1}^{m} (p_i(\lambda) - e_i(\lambda)) \\
&= \bar{q} \cdot (\bar{p} - \bar{e}) + 0.
\end{aligned}$$

Having shown that $\bar{q} \cdot (\bar{p} - \bar{e}) = \bar{w} \cdot (\bar{p} - \bar{e})$ even when $\bar{p}$ is a boundary point of $\mathbb{K}$ (provided $\bar{p}$ is near $\bar{e}$), the proof of the inequality

$$\bar{u}(\bar{p}) \cdot \frac{(\bar{p} - \bar{e})}{\|\bar{p} - \bar{e}\|} \leq \frac{\beta}{2\gamma}, \quad \bar{p} \text{ near } \bar{e}$$

proceeds exactly as in the above case when $\bar{p}$ is interior to $\mathbb{K}$.

We recognize the left hand side of this inequality as the derivative of $\|\bar{p}(t) - \bar{e}\|$ with respect to t. So

$$\frac{d}{dt} \|\bar{p}(t) - \bar{e}\| \leq \frac{\beta}{2\gamma} < 0$$

providing $\bar{p}(t)$ is sufficiently near $\bar{e}$, but not equal to $\bar{e}$. Once $\bar{p}(t)$ enters this neighborhood of $\bar{e}$, the distance $\|\bar{p}(t) - \bar{e}\|$ must decrease to zero within a period of time equal to $2\gamma/|\beta|$. When the distance drops to zero, we have $\bar{p}(t) = \bar{e}$, which is a stopping point of the dynamical system, since $\bar{e}$ is consistent. This completes the proof. □

In Theorem 9.1, we used the assumption that $\bar{e}$ is strictly consistent in order to prove that it is a local attractor of the relaxation labeling dynamical system. As a bonus, convergence to $\bar{e}$ occurs in finite time. One might object that the rapid convergence is an artifact of the normalization process in the projection operator. That is, since either $\bar{u} = 0$ or $\|\bar{u}(\bar{p})\| = 1$, the dynamical system must always move with unit speed before convergence. In fact, however, with the assumption of strict consistency, Theorem 9.1 would still hold true if the length normalization step were omitted from the projection algorithm.

If $\bar{p}$ is consistent, but not strictly consistent, then $\bar{p}$ may be a local attractor of the dynamical system, or it may be a saddle point, or even an unstable stopping point. We will not pursue these topics here.

We consider a local convergence result like Theorem 9.1 to be preferable to a universal convergence result, because the resulting consistent labeling is related to the initial labeling assignment. In Theorem 9.1, the hypothesis that a labeling is close to the consistent assignment requires that the labeling assignment at every object is close to the designated assignment. It would be desirable to extend Theorem 9.1 to the case when the initial labeling assignment is completely incorrect in a few "noncritical" object labelings.

X. Generalizations to Higher Order Compatibilities

In Section III, we stated that consistency could be defined in terms of support functions which depend on arbitrary orders of compatibilities. For example, for third-order compatibilities, we need a matrix of values $\{r_{ijk}(\lambda, \lambda', \lambda'')\}$, from which the support values $s_i(\lambda)$ are calculated:

$$s_i(\lambda) = \sum_{j,\lambda'} \sum_{k,\lambda''} r_{ijk}(\lambda, \lambda', \lambda'')\, p_j(\lambda')\, p_k(\lambda'').$$

In general, compatibilities of order k can be used to define the support components

$$s_i(\lambda) = \sum_{i_2,\lambda_2} \sum_{i_3,\lambda_3} \cdots \sum_{i_k,\lambda_k} r_{i,i_2,i_3,\cdots,i_k}(\lambda, \lambda_2, \cdots, \lambda_k) \cdot p_{i_2}(\lambda_2) \cdots p_{i_k}(\lambda_k).$$

Of special note is the case of first order compatibilities. In this case, the compatibilities are given by a vector $\{r_i(\lambda)\}$ and the support $s_i(\lambda)$ is independent of $\bar{p}$, and given by,

$$s_i(\lambda) = r_i(\lambda).$$

The analog to the variational inequality, which serves as a characterization of consistent labelings for higher order compatibilities, is given by $\bar{p} \in \mathbb{K}$ such that $\Sigma_{i,\lambda}\, s_i(\lambda)\,(v_i(\lambda) - p_i(\lambda)) \leqslant 0$ for all $\bar{v} \in \mathbb{K}$. Of course, $s_i(\lambda)$ depends on $\bar{p}$ according to the appropriate formula.

For second order compatibilities, we showed that a symmetry condition leads to the existence of a potential $A(\bar{p})$, satisfying $\bar{s} = \bar{q} = 1/2 \operatorname{grad} A(\bar{p})$. For compatibilities of first order, such a function always exists, namely,

$$A(\bar{p}) = \sum_{i,\lambda} r_i(\lambda)\, p_i(\lambda).$$

In the general case of kth order, the appropriate function $A(\bar{p})$ will satisfy the condition $\bar{s} = 1/k \operatorname{grad} A(\bar{p})$ when a certain symmetry condition is satisfied. In this case the average local consistency is given by

$$A(\bar{p}) = \sum_{i_1,\lambda_1} \sum_{i_2,\lambda_2} \cdots \sum_{i_k,\lambda_k} r_{i_1 \cdots i_k}(\lambda_1, \cdots, \lambda_k) \cdot p_{i_1}(\lambda_1) \cdots p_{i_k}(\lambda_k).$$

The symmetry condition, in its most general form, states that

$$\sum_{\sigma \in C_k} r_{i_{\sigma(1)}, i_{\sigma(2)}, \cdots, i_{\sigma(k)}}(\lambda_{\sigma(1)}, \cdots, \lambda_{\sigma(k)}) = k \cdot r_{i_1, \cdots, i_k}(\lambda_1, \cdots, \lambda_k)$$

for all sets of indexes $\{(i_1, \lambda_1), \cdots, (i_k, \lambda_k)\}$, where C_k is the group of cyclic permutations on k objects. For example, for order 3 the symmetry condition is

$$r_{ijk}(\lambda, \lambda', \lambda'') + r_{kij}(\lambda'', \lambda, \lambda') + r_{jki}(\lambda', \lambda'', \lambda) = 3 r_{ijk}(\lambda, \lambda', \lambda'')$$

for all i, λ, j, λ', k, and λ''.

The same algorithms serve to find consistent labelings. For the general nonsymmetric case, Algorithm 8.2 yields consistent labelings by finding solutions to the analog of the variational inequality, except that in Step 2', we set $q_i(\lambda) = s_i(\lambda)$. Here $s_i(\lambda)$ is calculated using $\bar{p}$ and the appropriate formula for the kth order compatibilities.

As mentioned in Section III, more general support functions are possible by combining supports of different orders. In full generality, the support vector $\bar{s}$ is a function $\bar{s}(\bar{p})$, where $s_i(\lambda)$ is computed from a nonlinear function of current assignment values in $\bar{p}$. Presumably, $s_i(\lambda)$ depends on the components of $\bar{p}_j$ for objects j near object i, and is relatively independent of the values of $\bar{p}_j$ for objects distant from i. The extent to which a particular problem yields local dependence of the support value is outside the scope of the abstract analysis given here. Once functions have been formulated from problem-specific considerations, the formal theory could allow one to determine whether a lower order approximation, perhaps in terms of a Taylor series expansion, is approximately sufficient.

When the compatibilities are first-order and the supports $s_i(\lambda)$ are constant and equal to the given coefficients $r_i(\lambda)$, a consistent labeling can be found immediately, i.e., without an iterative procedure. A simple argument shows that the unambiguous labeling $\bar{e} \in \mathbb{K}^*$ defined by

$$e_i(\lambda) = \begin{cases} 1 & \text{if } r_i(\lambda) > r_i(\lambda'), \text{ for all } \lambda', \\ 0 & \text{otherwise} \end{cases}$$

is always consistent in $\mathbb{K}$. If at some object i, there is no single label λ that maximizes $r_i(\lambda)$, then $e_i(\lambda)$ may be set to 1 for exactly one of them, and the result is a consistent labeling. In short, finding consistent labelings for first order compatibilities amounts to local maxima selection (cf. [19]).

It is interesting to note that Ullman's scheme for motion correspondence [15] is a problem of just this (unary) form, but for which the compatibilities are variable as a function of the underlying data. This more general situation requires an algorithm, such as gradient ascent, to obtain (local) maxima.

For compatibilities higher than second order, or nonpolynomial compatibilities, the difficulty becomes one of a combinatorial growth in the number of required computations. At this date, most implementations of conceptually similar relaxation labeling processes have limited the computations to second-order compatibilities.

XI. Comparisons with Standard Relaxation Labeling Updating Schemes

Algorithm 8.2 updates weighted labeling assignments by computing an intermediate vector $\bar{q}$, where

$$q_i(\lambda) = \sum_j \sum_{\lambda'} r_{ij}(\lambda, \lambda') p_j(\lambda')$$

and then updating $\bar{p}$ in the direction defined by the projection of $\bar{q}$ onto $T_{\bar{p}}$. As we shall show, the original updating formula introduced in [12] is an approximation to this new one. The intermediate vector $\bar{q}$ is identical in the two algorithms. The principal difference lies in the manner in which $\bar{q}$ is projected onto a tangent vector. In Algorithm 8.2, the tangent direction was obtained by maximizing $\bar{u} \cdot \bar{q}$ among $\bar{u} \in T_{\bar{p}} \cap B_1(0)$.

Other formulas have been proposed and used for relaxation labeling. We will concentrate on two standard formulas. The first was suggested by Rosenfeld *et al.* [12], and is given by

$$p_i(\lambda) := \frac{p_i(\lambda)\,[1 + q_i(\lambda)]}{\sum_{l=1}^{m} p_i(l)\,[1 + q_i(l)]}.$$

(It is assumed, when using this formula, that the $r_{ij}(\lambda, \lambda')$ values are sufficiently small that one can be sure that $|q_i(\lambda)| < 1$.) To consider the behavior of this standard formula, first assume that $\bar{p}$ is near the center of the assignment space, so that very approximately $p_i(\lambda) \approx 1/m$ for all i, λ. The updating can then be regarded as consisting of two steps. First, the vector $\bar{p}$ is changed into an intermediate $\hat{p}$, where

$$\begin{aligned}\hat{p}_i(\lambda) &= p_i(\lambda)\,[1 + q_i(\lambda)] \\ &\approx p_i(\lambda) + q_i(\lambda)/m.\end{aligned}$$

Next, $\hat{p}$ is normalized using a scalar constant for each object $\hat{p}_i$. When $\bar{p}$ is near the center of $\mathbb{K}$, this rescaling process shifts $\hat{p}$ in a direction essentially perpendicular to $\mathbb{K}$. That is, $\bar{p}$ is reset to approximately the projection of $\hat{p}$ onto $\mathbb{K}$. Denoting the orthogonal projection operator by $\mathcal{O}_K$, we have

$$\bar{p} := \bar{p}' \approx \mathcal{O}_K(\hat{p}) \approx \mathcal{O}_K(\bar{p} + \bar{q}/m)$$

by virtue of the continuity of $\mathcal{O}_K$. Further, assuming that $\bar{p}$ is in the interior of $\mathbb{K}$, and $\bar{q}$ is sufficiently small, then

$$\mathcal{O}_K(\bar{p} + \bar{q}/m) = \bar{p} + \frac{1}{m}\mathcal{O}_T(\bar{q})$$

where $\mathcal{O}_T$ is the orthogonal projection onto the linear subspace $T_{\bar{p}}$. However, by Lemma 7.3, the solution $\bar{u}$ to Problem 7.1 is obtained by normalizing $\mathcal{O}_T(\bar{q})$. Combining, we have that

$$\bar{p} := \bar{p}' \approx \bar{p} + \alpha\bar{u}$$

for some scalar α. Thus, $\bar{p}$ is reset to a vector which is approximately the updated vector that one would obtain by Algorithm 8.2. However, we have assumed that $\bar{p}$ is near the center of the assignment space.

When $\bar{p}$ is close to an edge or corner, the situation is somewhat more complicated. The first step in standard updating can be viewed as an initial operation changing $\bar{q}$, since the components of $\bar{q}$ corresponding to small components of $\bar{p}$ have minimal effect. That is, the operation $p_i(\lambda) + p_i(\lambda)\,q_i(\lambda)$ differs from adding $\bar{q}$ to $\bar{p}$ (or a small multiple of $\bar{q}$ to $\bar{p}$) in that the motions in directions perpendicular to the nearby edges are scaled down. Furthermore, the normalization step is no longer equivalent to a simple projection. Rather, it can be seen that the rescaling is equivalent to modifying the updating direction back toward the center of the assignment space. This further constricts motions perpendicular to nearby edges, when that motion is outward. Qualitatively, this behavior of the standard updating formula near the edges is reminiscent of the projection operator on the same edges. However, because the formula results in attenuation of motion perpendicular to an edge, even when that motion is directed back toward the surface, the standard updating formula will tend to round corners more smoothly and more slowly. Further, a zero component can never become nonzero, even if the evidence supports increasing the value. In summary, there is strong agreement between the projection operator and standard updating in the interior of $\mathbb{K}$, but some differences are possible near the edges. One of the probable effects of these differences is to show convergence when the older updating formula is used.

An alternative updating formula is now common, based on its relationship to a Bayesian analysis of a single iteration of relaxation labeling [11]. In this formulation, the compatibility coefficients are nonnegative, and the updating formula is given by

$$p_i(\lambda) := \frac{p_i(\lambda) \cdot \left(\prod_{j=1}^{n} \sum_{\lambda'=1}^{m} r_{ij}(\lambda, \lambda') p_j(\lambda') \right)}{\sum_l p_i(l) \prod_j \left(\sum_{\lambda'} r_{ij}(\lambda, \lambda') p_j(\lambda') \right)}.$$

Once again, the denominator is a normalization term. The numerator can be rewritten as

$$\begin{aligned}&p_i(\lambda) \prod_{j=1}^{n} \left[1 + \sum_{\lambda'=1}^{m} (r_{ij}(\lambda, \lambda') - 1) p_j(\lambda') \right] \\ &= p_i(\lambda) \left[1 + \sum_j \sum_{\lambda'} (r_{ij}(\lambda, \lambda') - 1) p_j(\lambda') \right. \\ &\qquad + \text{quadratic terms of } \bar{p} \\ &\qquad \left. + \text{higher order terms} \right].\end{aligned}$$

We can view this terms as $p_i(\lambda)\,[1 + q_i(\lambda)]$, where $q_i(\lambda)$ is a complicated nonlinear function of the assignment vector $\bar{p}$. When viewed in this fashion, we see that the "product of sums" updating formula is identical to the earlier updating formula, except that the formulation of the updating direction $\bar{q}$ is more complicated, and disguised. In this sense, the same comments as before apply to the relationship between Algorithm 8.2, which uses the projection operator, and more classical relaxation labeling procedures using other updating formulas.

XII. Summary and Conclusions

Relaxation labeling processes were introduced and studied as mechanisms for employing context and constraints in labeling problems. Over the past few years, a number of researchers have applied relaxation labeling processes to a variety of different tasks. Some of these applications were more successful than others. In all cases, there was an intuition that the compatibility coefficients ought to be doing something useful to the labeling assignment (at least for the first few iterations), but there has never been a succinct and reasonable statement of what useful properties were being enhanced. Lacking a proper model characterizing the process and its stopping points, the choice of the coefficient values and the updating formula are subject only to empirical justification.

In this paper, we have attempted to develop the foundations of a theory explaining what relaxation labeling accomplishes. This theory is based on an explicit new definition of consistency, and leads to a relaxation algorithm with an updating formula which uses a projection operator. It is our hope that these results will help explain why some applications of relaxation labeling have been successful and others have been less impressive, and will assist in a more proper and systematic design of relaxation labeling processes in the future.

In discrete relaxation, a label is discarded if it is not supported by the local context of assigned labels. In the limiting assignment of labels, every retained label is unsupported, whereas every label that has been discarded is unsupported. Of course, in discrete relaxation, support is an all-or-nothing proposition, and is based on a system of logical conditions on neighboring assignments.

In extending this idea to weighted label assignments, support values become ordered real numbers, and are expressed by potentially complicated functions of local weighted assignment values. An unambiguous labeling is consistent, according to our definition, if the support for the instantiated label at each object is greater than or equal to the support for all other labels at that object. A generalization of this idea leads to a definition of consistency for weighted labeling assignments.

We showed that the relaxation labeling process defined by Algorithm 8.2 together with the projection operator specified in Appendix A stops at consistent labelings. We further showed that if one begins sufficiently near a consistent labeling, the dynamic process will then converge to that labeling. The sense in which a consistent labeling constitutes an improvement over an initial labeling assignment relates to the proximity of the two labelings and the precise meaning of the terms in the definition of consistency. We also showed that our relaxation algorithm is approximated by some of the more standard formulas used for labeling, but offer our algorithm as an alternative formulation that is based on a much more mathematically well-founded theory.

When the support values are given by specific polynomial formulas, and provided certain symmetry properties hold among the coefficients, the relaxation labeling algorithm given here is equivalent to gradient ascent using a functional which we have called average local consistency. We used this optimization viewpoint to introduce the algorithm, and were in part motivated to study the more general case by earlier work of others using optimization to assist in labeling problems. However, the symmetry assumption is restrictive, and, as we showed by giving up the optimization functional, unnecessary.

Relaxation labeling processes were originally conceived in terms of cooperation, as a system of local, simple, and sparsely interacting processes. Its current form is still structured in this fashion. Now, however, we can interpret the meaning of the compatibility matrices in terms of the permissible local configurations of consistent labelings.

Much work remains to be done to analyze, extend, and apply the theory presented in this paper. Practical considerations, such as efficient implementations of the projection operator, choice of the step size, and normalization methods need to be addressed. More consideration should be placed on design criteria for particular relaxation applications, as well as on the formal relationships between relaxation labeling and other families of algorithms for solving similar tasks. The incorporation of logical constraints, and the relative merits of using more complicated support functions are intriguing topics for further study. We hope that the answers to some of these questions are facilitated by the foundations presented here.

Appendix A
Updating Direction Algorithm

In Algorithm 7.4 and the relaxation labeling Algorithm 8.2, a solution to Problem 7.1 is required. Problem 7.1, as stated in the text, amounts to maximizing the vector dot product $\bar{q} \cdot \bar{u}$ among all tangent vectors $\bar{u} \in T_{\bar{p}}$ of length less than or equal to one. Both $\bar{p}$ and $\bar{q}$ are given. Thus, we have the following.

Problem: Find $\bar{u} \in T_{\bar{p}} \cap B_1(0)$ such that

$$\bar{u} \cdot \bar{q} \geqslant \bar{v} \cdot \bar{q} \quad \text{for all } \bar{v} \in T_{\bar{p}} \cap B_1(0).$$

If $\bar{u} = 0$ is a possible solution, then $\bar{u} = 0$ is the solution that should be returned.

We recall that

$$T_{\bar{p}} = \left\{ \bar{v} \in \mathbb{R}^{nm} : \sum_{\lambda=1}^{m} v_i(\lambda) = 0, \quad i = 1, \cdots, n, \quad \text{and} \right.$$

$$\left. v_i(\lambda) \geqslant 0 \quad \text{whenever } p_i(\lambda) = 0 \right\}$$

and $B_1(0) = \{\bar{v} \in \mathbb{R}^{nm} : \|\bar{v}\| \leqslant 1\}$.

The algorithm given below is intended to replace the updating formulas in common use in relaxation labeling processes. The projection operator, as given below, has the advantage of being based on a theory of consistency, and permits the analytic proof of convergence results.

A complete discussion of the algorithm, as well as a proof of its correctness, is given in an accompanying paper, "A Feasible Direction Operator for Relaxation Methods," which appears in this issue [10]. Here, we merely give a formal specification of the algorithm. We state the procedure

```
PROCEDURE Projection_Operator(p,q,n,m)
  $
  $  This procedure returns the vector u used to update the weighted
  $  labeling assignment p in a relaxation labeling process.
  $
  $  p  is an element in the assignment space IK, formed as a vector
  $    of vectors.  That is, p = [p1,p2,..,pn], each pi = [pil,..,pim],
  $    each pik is a nonnegative real, the sum over k of pik is 1.
  $  q  is the direction vector, obtained from the support values at the
  $    current labeling assignment p, represented as an n-vector of
  $    m-vectors (in the same form as p).
  $  n  is the number of objects.
  $  m  is the number of labels.
  $
  $  The updating direction u is returned through the procedure name,
  $  and is an n-vector of m-vectors.  A typical call will appear as
  $                    u := Projection_Operator(p,q,n,m)
  $
  $  In SETL, n and m are not needed in the parameter list.  In that
  $  case, the procedure would begin with the following statements:
  $                    n := #p ;  m := #p(1) ;
  $

  $  Begin algorithm:

   u := [ ];                                $ Defines u as a vector

   LOOP for i in [1..n] DO

      D := { k in [1..m] | p(i)(k) = 0. };
      S := {};

      LOOP DO         $ Indefinite loop terminates when QUIT is executed.

        ns := #S
        t  := +/[ q(i)(k) : k in [1..m] | k NOTIN S] / (m-ns);

  $       i.e., t is the sum of q(i)(k) over k not in S, divided by m-#S

        S  := { k in D |  q(i)(k) < t  }
        IF #S = ns  THEN  QUIT;  END IF;

      END LOOP DO;

      u(i) := [ (IF k in S then 0. ELSE q(i)(k) - t ) : k in [1..m] ];

  $                 i.e., u(i)(k) = 0 if k IN S, = q(i)(k)-t otherwise.

   END LOOP for i;

  $   Normalize the vector u :

   Norm := SQRT( +/[uik**2 : ui = u(i), uik = ui(k)] );
   u :=  (IF   Norm = 0. then   u
          ELSE                  [ [uik/Norm : uik = ui(k)] : ui = u(i)] );

   RETURN u;

END PROCEDURE Projection_Operator;
```

Fig. 7.

in the computer language SETL, a very high level programming language which has dynamic allocation and includes set and tuples in its set of primitives [20]. The operators in the SETL language closely resemble standard mathematical sentences (see Fig. 7). A more typical mathematical description of the algorithm is given in the accompanying paper.

The returned vector u will be a solution to the projection problem. It can also be shown that the potentially infinite loop (LOOP DO in the SETL code) will execute at most $m+1$ times for each value of i in $[1,\cdots,n]$. That is, the projection operator is a finite iterative algorithm. Although the SETL specification given above is executable code, in practical applications the projection algorithm will be implemented in Fortran, assembler, or perhaps even special purpose hardware.

We have also included the normalization $\|\bar{u}\| = 1$ (or $\bar{u} = 0$) as part of the algorithm, as required by the statement of the problem. When n is large, however, the calculation of the norm of the unnormalized vector $\bar{u}$ may be costly. As mentioned after Theorem 9.1, the local convergence result does not actually require that $\bar{u}$ be normalized. In fact, the theorem holds true if any norm measure is used, as long as small steps are taken (with respect to that norm) in the direction $\bar{u}$. However, as specified by Step 6 of Algorithms 7.4 and 8.2, the step size must be dynamically adjusted to make sure that step never forces any component of the current assignment labeling to become negative. Both the normalization process and the dynamic step size adjustment are greatly simplified if applied to individual subvectors $\bar{u}_i$ and $\bar{p}_i$ separately. In this case, $\|\bar{u}_i\| = 1$ for $i = 1,\cdots,n$, and the step length α_i is adjusted for each i so that $\alpha_i \leqslant h$, α_i maximized, and $\bar{p}_i + \alpha_i \bar{u}_i$ has nonnegative components. However, this changes the relaxation labeling process, since the direction of updating may not be parallel to $\bar{u}$. Nonetheless, one can still prove that stopping points are consistent labelings and strictly consistent labelings are local attractors.

APPENDIX B
GLOSSARY OF SYMBOLS

$A(\bar{p})$	Average local consistency of a labeling (introduced in Section V).
$B_1(0)$	The ball of vectors of radius 1 centered at 0 (Section VII).
$\delta_{i\alpha}$	Kronecker delta, $\delta_{i\alpha} = 0$ if $i \neq \alpha$, and $= 1$ if $i = \alpha$ (Section V).
$\bar{e}$	An unambiguous labeling vector (Section IX).
$\bar{e}_k$	An m-vector of the form $\bar{e}_k = [0, \cdots, 1, \cdots, 0]$, where the 1 appears in the kth component (Section III).
grad $A(\bar{p})$	Gradient of A evaluated at $\bar{p}$ (Section V).
i, j	Variables to indicate nodes in the labeling graph, or indexes through sets of nodes. j typically indicates a neighbor of i (Section I).
$\mathbb{K}^*$	The space of unambiguous labelings (Section III).
$\mathbb{K}$	Convex space of weighted labeling assignments (Section III).
λ	Variable to either denote a label or to serve as an index through a set of labels (Section I).
Λ_i	Set of labels attached to node i (Section I).
Λ_{ij}	Constraint relation listing all pairs (λ, λ') such that λ at i is consistent with λ' at j.
m	Number of labels in Λ_i (Section I).
n	Number of nodes in G (Section I).
$\mathcal{O}_W \bar{v}$	Projection operator, indicating the projection of a vector $\bar{v}$ onto a space W (Section XI).
$p_i(\lambda)$	Weight indicating the strength with which label λ is associated with node i (Section III).
$\bar{p}_i$	The labeling vector associated with node i: $\bar{p}_i = [p_i(1), p_i(2), \cdots, p_i(m)]$ (Section III).
$\bar{p}$	The complete labeling assignment vector (or, for short, labeling) $\bar{p} = [\bar{p}_1, \bar{p}_2, \cdots, \bar{p}_n]$ (Section III).
$r_{ij}(\lambda, \lambda')$	Compatibility matrix over pairs of labels on pairs of nodes (Section III).
$r_{ijk}(\lambda, \lambda', \lambda'')$	Compatibility matrix over triples of labels on triples of nodes (Section III).
$R_{ij}(\lambda, \lambda')$	Indicator function for the numerical representation of Λ_{ij} (Section I).
$\mathbb{R}^m$	m-dimensional Euclidean space (Section III).
$s_i(\lambda)$	Support given by an (unambiguous or ambiguous) labeling to label λ on node i (Section III).
$S_i(\lambda)$	Support for label λ on i from a discrete labeling (Section I).
$T_{\bar{p}}$	Tangent space at $\bar{p} \in \mathbb{K}$ (Section VI).
$\bar{u}$	The tangent direction in which updating takes place (the projection of $\bar{q}$) (Section VII).
$\bar{v}$	An arbitrary weighted labeling assignment in $\mathbb{K}$ (Section III).
$\#S$	The number of elements in the set S (Appendix A).
$a := b$	The value of b replaces the current value of a (Section II).
$\bar{u} \cdot \bar{v}$	Vector dot product of $\bar{u}$ and $\bar{v}$ (Section V).
$\|\bar{u}\|$	Euclidean norm of $\bar{u}$ (Section VII).

REFERENCES

[1] K. Arrow, L. Hurwicz, and K. Uzawa, *Studies in Linear and Nonlinear Programming*. Stanford, CA: Stanford Univ. Press, 1960.

[2] M. Berthod and O. Faugeras, "Using context in the global recognition of a set of objects: An optimization approach," in *Proc. IFIP*, 1980.

[3] N. P. Bhatia and G. P. Szego, *Dynamical Systems: Stability Theory and Applications* (*Lecture Notes in Mathematics*), vol. 35. New York: Springer-Verlag, 1967.

[4] L. Davis and A. Rosenfeld, "Cooperating processes for low-level vision: A survey," *Artificial Intell.*, vol. 17, p. 412, 1981.

[5] O. Faugeras and M. Berthod, "Improving consistency and reducing ambiguity in stochastic labeling: An optimization approach," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-3, p. 245, 1981.

[6] R. M. Haralick and L. Shapiro, "The consistent labeling problem: Part 1," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-1, p. 173, 1979.

[7] G. E. Hinton, "Relaxation and its role in vision," Ph.D. dissertation, Univ. Edinburgh, Dec. 1979.

[8] D. Kinderlehrer and G. Stampacchia, *An Introduction to Variational Inequalities and Their Applications*. New York: Academic, 1980.

[9] D. Luenberger, *Optimization by Vector Space Methods*. New York: Wiley, 1969.

[10] J. Mohammed, R. Hummel, and S. Zucker, "A feasible direction operator for relaxation methods," this issue, pp. 330-332.

[11] S. Peleg, "A new probabilistic relaxation scheme," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-2, p. 362, 1980.

[12] A. Rosenfeld, R. Hummel, and S. Zucker, "Scene labeling by relaxation operations," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-6, p. 420, 1976.

[13] R. Southwell, *Relaxation Methods in Engineering Science*. Clarendon, 1940.

[14] S. Ullman, "Relaxation and constrained optimization by local processes," *Comput. Graphics Image Processing*, vol. 10, p. 115, 1979.

[15] ——, *The Interpretation of Structure from Motion*. Cambridge, MA: MIT Press, 1979.

[16] D. Waltz, "Understanding line drawings of scenes with shadows," in *The Psychology of Computer Vision*, P. Winston, Ed. New York: McGraw-Hill, 1975.

[17] S. Zucker, "Labeling lines and links: An experiment in cooperating computation," in *Consistent Labeling Problems in Pattern Recognition*, R. Haralick, Ed. New York: Plenum, 1980.

[18] S. Zucker, R. Hummel, and A. Rosenfeld, "An application of relaxation labeling to line and curve enhancement," *IEEE Trans. Comput.*, vol. C-26, pp. 394-403, 922-929, 1977.

[19] S. Zucker, Y. Leclerc, and J. Mohammed, "Relaxation labeling and local maxima selection: Conditions for equivalence," *IEEE Trans. Pattern Anal. Machine Intell.*, to be published.

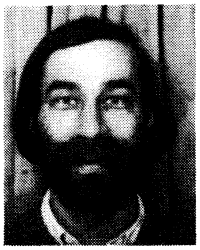
[20] R. B. K. Dewar *et al.*, "Programming by refinement," *TOPLAS*, vol. 1, p. 27, 1979.

Robert A. Hummel (M'82) received the B.A. degree in mathematics from the University of Chicago, Chicago, IL, and the Ph.D. degree in mathematics from the University of Minnesota, Minneapolis.

While in school, he spent summers at the Picture Processing Lab of the Computer Science Center at the University of Maryland. He has also been employed at Stanford Research Institute and the Signal and Image Processing section of Honeywell's Systems and Research

Center in Minneapolis, MN. From 1980 to 1982 he was a Courant Institute Instructor in the Department of Mathematics, New York University's Courant Institute of Mathematical Sciences. He is currently an Assistant Professor of Computer Science in the Courant Institute, and a member of the Robotics and Vision Research Group there. He also serves as a consultant to Martin Marietta Aerospace in Orlando, and is an adjunct faculty member of IBM's System Research Institute in New York. In mathematics, his research interests include variational methods for the study of partial differential equations, fluid mechanics, and foundations of thermodynamics. Other research interests include computer vision and image processing.

Dr. Hummel is a member of Phi Beta Kappa and the American Mathematical Society.



Steven W. Zucker (S'71-M'75) received the B.S. degree in electrical engineering from Carnegie-Mellon University, Pittsburgh, PA, in 1969, and the M.S. and Ph.D. degrees in biomedical engineering from Drexel University, Philadelphia, PA, in 1972 and 1975, respectively.

From 1974 to 1976 he was a Research Associate at the Picture Processing Laboratory, Computer Science Center, University of Maryland, College Park. He is currently an Assistant Professor in the Department of Electrical Engineering, McGill University, Montreal, P.Q., Canada. His research interests include image processing, computer vision, and artificial intelligence.

Dr. Zucker is a member of Sigma Xi and the Association for Computing Machinery.