TR-1123
MCS-76-23763

January, 1980
ENG-79-04037

COOPERATING PROCESSES FOR LOW-LEVEL VISION: A SURVEY

Larry S. Davis
Department of Computer Sciences
University of Texas
Austin, TX 78713

Azriel Rosenfeld
Computer Science Center
University of Maryland
College Park, MD 20742

ABSTRACT

Cooperating local parallel processes can be used as aids in assigning numerical or symbolic labels to image or scene parts. Various approaches to using such processes in low-level vision are reviewed, and their advantages are discussed. Methods of designing and controlling such processes are also considered.

The support of the National Science Foundation under Grants MCS-76-23763 to the University of Maryland, and ENG-79-04037 to the University of Texas, is gratefully acknowledged, as is the help of Eleanor Waters in preparing this paper. The authors also wish to thank Shmuel Peleg and Michael O. Shneier for helpful discussions and comments.

1. Introduction

The early stages of computer vision involve assigning symbolic and numerical labels to image parts. For example, pixels can be assigned symbolic land-use category labels based on their spectral signatures, or numerical stereo or motion disparity labels based on local comparisons between pairs of pictures.

The enormous amount of data comprising an image demands that such labelling processes be very fast. Sequential labelling processes, while they can make full use of context, cannot be speeded up in general. Moreover, the labellings which they compute are often sensitive to the order in which the parts are considered. A more promising approach -- one that is also motivated by studies of biological visual systems -- is to make the processes highly parallel. This requires that each picture part be analyzed and labelled independently of the others. When we do this, however, many errors are made, because contextual information is not adequately used.

A solution to this problem is to assess the labelling possibilities for every part independently and then compare each part's assessments to those of other, related parts, in order to detect and correct potential inconsistencies. Since both the assessment and the comparison can be done independently for every part, each stage of the process is parallel. On the other hand, context is now being used at the comparison stage, when related parts are able to communicate and "cooperate." To keep the computational cost low, the comparisons should be local; they should involve only parts that are directly related (e.g., neighboring pixels). This localness can be compensated for by iterating the comparison process, in order to allow

information to propagate.

These considerations lead naturally to the design of a "cooperative" approach to labelling picture parts which allows context to be used in the labelling process while still permitting fast parallel implementation and low computational cost. Such processes are called "relaxation" processes, because of their resemblance to certain iterative processes used in numerical analysis. Very generally, a relaxation process is organized as follows:

a) A list of possible labels is independently selected for each part, based on its intrinsic characteristics. A measure of confidence can also be associated with each possible label.

b) The possibilities (and confidences) for each part are compared with those for related parts, based on a model for the relationships between the possible labels of picture parts. Labels are deleted or confidences are adjusted to reduce inconsistencies.

c) Step (b) can be iterated as many times as required.

This approach is very general: We have not specified how to formulate label relationship models, choose possibilities, estimate confidences, or adjust them; nor have we discussed when the process should be iterated, and if so, how many times. The next three sections of this paper discuss these issues, and survey applications of such processes to problems in low-level computer vision.

2. Cooperation/Competition

a relaxation process is a computational mechanism which allows a set of "myopic" local processes associated with picture parts to interact with one another in order to achieve a globally consistent interpretation of a picture. This interaction involves the updating of each picture part's self-assessment which is represented as a discrete or fuzzy labelling. A discrete labelling simply associates a set of possible labels, or names, with each picture part, while a fuzzy labelling additionally associates a likelihood with each label.

Labels are usually specified extensionally by actually listing the appropriate labels for each picture part. The list is a subset of some given, finite universe of labels. For some applications the natural label set is infinite. For example, the label for a picture part might represent the range, or distance, from the sensor to some specific point in the picture part such as its centroid. In such cases, a labelling may need to be specified intensionally; for example, an interval of numbers may be used to specify the range -- i.e., we assume that the true range is between a nearest distance r_1 and a farthest distance r_2. All the applications we will consider in this paper use only finite universes of labels.

2.1. Neighborhood models

A relaxation process is determined by specifying a model for the neighborhood of a picture part and a model for the interaction between labellings of neighboring picture parts.

The neighborhood model for a relaxation process specifies which pairs of picture parts directly communicate with one another in the relaxation process, and determines the topology of the graph on which the relaxation

process operates. This graph has individual picture parts as nodes. Its arcs connect those pairs of parts that communicate with one another. The neighborhood model is usually designed to establish connections only between "nearby" parts to satisfy the locality constraint.

A neighborhood model is specified by a set of neighbor relations $r = \{r_1, r_2, \ldots, r_n\}$. Each r_i is a binary relation defined over the appropriate set of picture parts. For example, if the picture parts are pixels, then the neighborhood model might specify that a pixel is connected to every pixel in its 3x3 neighborhood. In this case, there are still several possibilities for the relations contained in the set r. For example, r might be the set {directly above, directly below, etc.} which would distinguish between pairs of points that are horizontally adjacent, vertically adjacent, etc., or it could be the singleton relation "in the 3x3 neighborhood." In the latter case, the connections between pairs of pixels would not be recoverable from the graph on which the relaxation process will operate. The choice of r will, in general, be determined by the isotropy of the universe of labels. For example, if we are designing a relaxation process for edge reinformcement, then the relative positions of pixels are crucial since edges generally "line up," while if we are designing a relaxation process to enhance an image's grey levels, then the positional information may not be required.

When the picture parts are regions rather than pixels, then connections might be formed between adjacent regions only. In some situations, it might be necessary to distinguish between regions that are above, below, inside, surrounding, etc.

The neighborhood model determines which pairs of picture parts directly communicate through the relaxation process. The next section discusses the

various ways in which they may communicate.

2.2 Interaction models

The interaction model defines how a picture part changes its labelling based on the labellings of its neighbors. An interaction model is composed of two parts:

1) a knowledge representation for the relationships between labels, and
2) a mechanism, or procedure, for applying the knowledge in (1) to change, or update, labellings.

For discrete labellings the simplest knowledge representation is a set of the pairs of labels that can simultaneously be associated with pairs of neighboring picture parts. It can be represented by a binary relation R defined over the universe of labels D. Intuitively, $(d,d') \in T$ if a pair of neighbors can simultaneously be labelled with d and d'. In general, there is a binary relation associated with each neighbor relation.

The most obvious updating mechanism is a label discarding process, which looks at pairs of picture parts at a time. A label, d, can be deleted from the labelling of a picture part if, for some neighboring picture part, that neighbor does not contain a label, d', in its labelling with $(d,d') \in R$. This is, essentially, Waltz's filtering algorithm [1]. Rosenfeld et al. [2] show that label discarding can, in principle, be applied in parallel at every picture part and that by iterating the process of discarding labels a unique, maximally consistent labelling is computed. The process can be generalized in a variety of ways -- e.g., the knowledge representation might be in terms of n-ary relations (for example, a 3-ary relation is required

to specify that a picture part is between two others). The label discarding process now considers a picture part and n-1 of its neighbors at a time, rather than one neighbor at a time. There are many other possibilities based on computing lower-order projections of n-ary relations. See Haralick et al. [3] and Haralick and Shapiro [4] for a detailed discussion.

The binary relation knowledge representation can be generalized to fuzzy labellings by specifying a real-valued compatibility function, C, whose domain is DxD. As before, in general, a compatibility function is defined for each picture relation in the set r. A variety of applications have used compatibility functions whose range is [-1,1]. Intuitively, if $C(d,d') = -1$, then d and d' are maximally incompatible, and the strong presence of d' at one picture part (i.e., d' has a high likelihood at that part) should depress the likelihood of d at a neighboring picture part. If $C(d,d') = 1$, then d and d' are maximally compatible, and the strong presence of d' at a picture part should increase the likelihood of d at a neighboring picture part. Finally, if $C(d,d') = 0$, then the presence of d' at a picture part should have no effect on the likelihood of d at a neighboring part. Intermediate values of C should have intermediate effects.

As an example, suppose we are designing a relaxation process to enhance the results of a local line detection algorithm. Then the set of labels may be {horizontal (h), vertical (v), left-diagonal (dℓ), and right-diagonal (dr), and the set r might contain the relations vertically-adjacent (V), horizontally-adjacent (H), left-diagonally-adjacent (L) and right-diagonally-adjacent (R)}. If, in the class of images being considered, linear features are thin and have few corners (i.e., the curvature is ordinarily low), then

we would expect, e.g., that $C_V(v,v)$ would be high, while $C_V(v,h)$ would be low, since an h vertically adjacent to a v would form a right angle. $C_H(v,v)$, on the other hand, would be low, since the linear features are thin.

Several mechanisms have been suggested for applying this knowledge representation to updating labellings. For example, Rosenfeld et al. [2] suggested the formula:

$$p_i'(d) = p_i(d)(1 + Q_i(d))/N$$

where

$$Q_i(d) = \sum_j m_{ij} \sum_{d'} C(d,d')\, p_j(d')$$

and N is a normalizing factor which guarantees that $\sum p_i(d) = 1$. The m_{ij} values can be used to give higher weight to some neighbors at part i than others. $Q_i(d)$ measures the overall support of the neighborhood of part i for label d; it takes on values in the range [-1,1] and can be interpreted similarly to C. The above operation is applied in parallel at every part and for every label. The p' values then replace the p values, and the operation can be iterated.

Variations on the above theme are possible and lead to better results in some applications. For example, one can apply a "max-min" rule where

$$Q_i(d) = \min \{\max \{C(d,d')p_j(d')\}\}$$

which reduces to the discrete algorithm described above when C and p are constrained to take on only the values 0 or 1.

There are several disadvantages to the relational knowledge representation for the interactions between labels. First, it is a single-level representation scheme. The solution to many image understanding problems requires that images be described at several levels of abstraction. Attempting

to compile all interactions between conceptually higher-level pictorial entities down to interactions between only the lowest level pictorial features is almost always cumbersome and inefficient, and is sometimes impossible. Section 2.3 discusses hierarchical relaxation systems.

A second important shortcoming of the relational framework is that the algebraic combination of evidence treats all of the interactions between labels uniformly, which is often not desirable. Furthermore, there are classes of intuitively plausible constraints that can only be represented very inefficiently in a relational framework. For example, the very simple constraint

> A picture part can be a d_1 only if all adjacent picture parts can be d_2's

requires an n-ary relation to represent it, where n is the maximum degree of any node in the graph on which the relaxation procedure operates.

Such problems can be overcome by adopting a more powerful representation for label interactions than relations. For example, constraints between labels can be represented using logic statements [5]. This allows a much wider class of constraints to be efficiently represented and applied to the analysis of a picture. The natural mechanism for applying the constraints is, then, a general inference procedure. Such a scheme has not yet been applied to any image understanding problem; its application to linear feature detection is currently under investigation.

2.3. Hierarchy

Very often, a natural and economical solution to an image analysis problem requires that pictorial entities be described at several levels of detail. For example, to recognize an image segment as the top view of an airplane

based on the shape of its boundary might require recognizing airplane pieces as engines, wings, tail sections, etc., and then grouping them into larger pieces of airplanes, and finally into a complete airplane shape. Or, as a second, more complex, example, reading a word in cursive script involves segmenting the word into primitive parts such as strokes, grouping the strokes into large letter pieces, those pieces into letters and finally the letters into a word.

As discussed above a single level relaxation system is specified by a neighborhood model and a label interaction model. To design a hierarchical relaxation system having k levels, one needs to not only define a neighborhood model and an interaction model at each level, but also a construction model (which, given the labelling of pieces at level m, can construct the pieces and their labellings at level m+1), and an across-level neighbhorhood model. This last model is ordinarily based simply on constituency -- i.e., a level m+1 piece is linked to each of the level pieces from which it was formed [6].

An important design criterion for such processes is that the construction models and the interaction models be consistent. Intuitively, the consistency constraint means that the relations between level m labels implied by the construction models for all higher levels do not contradict the explicit relations between level m labels mentioned in the level m interaction model. A simple example should help clarify this point.

Suppose that we are constructing a hierarchical relaxation system for reading cursive script, and that for the particular corpus of words that we wish to read, there are no words in which the letter "h" precedes the letter "u." Now, suppose that at the large letter piece level, an h ends with a

piece p_1 and a u begins with a piece p_2, and that no other letter contains either a p_1 or a p_2. Then clearly, the compatibility of p_1 and p_2 at the large letter piece level should be as low as possible. If it were not, then the construction model taking letters into words would be inconsistent with the interaction model for large letter pieces.

One possible approach to guaranteeing such consistency is to specify only the construction models, and then compile the interaction models from the construction models. This not only guarantees that the interaction models are consistent with the construction models, but also avoids the tedious task of specifying all the constraints contained in the interaction models.

For example, in reading cursive script from a known corpus, the letter cooccurrence probabilities can be compiled directly from the corpus, and these can be used as an interaction model at the letter level. Then, given a decomposition of letters into large letter pieces a similar process can produce cooccurrence probabilities for large letter pieces, etc. This was done by Hayes [7] in his handwriting analysis system.

As a second example, in [8] Davis and Henderson describe a hierarchical shape analysis system. Shapes are modeled by hierarchical relational networks which describe the arrangement and geometrical properties of shape pieces at several levels of detail. The representation is designed in such a way that local constraints about the appearance of the shape can be automatically compiled from the representation. Thus the representation serves as a set of construction models, and the compiled constraints are used as interaction models.

Although the compilation of interaction models from construction models is a powerful idea in the design of hierarchical relaxation systems, it does

not address the issue of how one determines whether or not additional constraints, not derivable from the construction model, can be consistently added to the interaction models. This situation might arise when analysis of one part of an image yields information that can be used to guide the analysis of other parts, or if prior knowledge is available that is not ordinarily available. The following simple example illustrates the problem. Suppose that we are attempting to recognize airplanes, and that prior information is available about the angle that the wings of planes will make with the fuselage. How can we determine that this extra information is consistent with the existing airplane model? (If it is not, no shapes will be recognized as airplanes!) How does the relaxation process even make use of this information, assuming that it is consistent with its interaction models? For currently existing systems, there is no effective means for checking the consistency of externally specified information with current knowledge, or for uniformly applying such information to enhance the relaxation process. This points out another advantage of the logic representation mentioned in Section 2.2. If construction models are specified as statements in logic, then interaction models can still be compiled from the construction models. Furthermore, the consistency of added information with current knowledge can be determined, and the general inferencing capabilities associated with logic would enable such a system to make use of any additional information as well.

3. Applications

A wide variety of labelling processes can be used at various stages of computer vision. Many of these processes operate at the pixel level -- i.e., the parts to be labelled are individual pixels, and the interaction is between neighboring pixels. In general, image segmentation can be regarded as pixel labelling, with the labels defining a partition of the image into subsets. Thus relaxation methods are applicable to most of the standard image segmentation techniques, including pixel classification based on gray level or color (thresholding, multispectral classification), as well as detection of local features (peaks or spots, ridges or curves, edges, corners, or matches to arbitrary templates). Many examples of such methods are given in Sections 3.1 and 3.2.

Relaxation methods can also be applied to situations involving several images (e.g., disparity measurement in motion or stereo), or several sets of labels simultaneously applied to a single image (e.g., "intrinsic image" labels); see Section 3.3. The can also be used to label picture parts that are larger than single pixels, i.e., windows or regions, and to detect specified local configurations of such parts, as briefly discussed in Section 3.4.

3.1. Pixel classification based on gray level or color

Suppose that a scene is composed of a few objects or regions each of which is homogeneous in color. The colors of the pixels in an image of that scene should then display clustering behavior: There should be clusters in the scatter plot of color values, corresponding to the color characteristics of the regions. Under these circumstances, a natural way to segment the image is to classify the pixels as belonging to these clusters. In fact, this is the standard method of segmenting multispectral terrain images into

terrain types or land use classes, based on clustering the spectral signatures of the pixels. It is also widely used to segment black-and-white images into light and dark regions by thresholding the gray levels so as to separate peaks on the histogram. Analogous methods can be used for arrays of other types of values, e.g., range data.

Conventionally, the pixels are classified independently. In order to use a cooperative approach, possible class memberships must be determined, or class membership confidences estimated, for each pixel, and the results compared with those for neighboring pixels. To define an appropriate interaction model, some assumptions must be made about the kinds of neighbors that we expect a pixel to have. Since the scene consists of a few homogeneous regions, most pixels will be in the interior of a region. The neighbors of those pixels should all be alike. Some pixels, of course, will be on interregion boundaries; in this case some of the neighbors of the pixel will belong to the same region as the pixel itself, but others will belong to a different region. Neighborhoods at which three or more regions meet will be rare, and will be ignored here.

The situation just described is characteristic of a large class of cooperative pixel labelling problems: the neighbors of a pixel belong, with rare exceptions, to at most two classes, one of which is the class containing the pixel itself. Several types of interaction models can be used in such situations:

1) The neighborhood used can consist, for each pixel, of those neighbors that most resemble the pixel. If the neighbors cluster into two classes, this is straightforward; if not, one can use a fixed number of "best" neighbors, on the assumption that these neighbors

are the ones most likely to belong to the same region as the pixel. For the chosen neighbors, the interaction model is then quite simple: like reinforces like [9].

Alternatively, we can examine a set of one-sided neighborhoods of the pixel, and choose the one in which the gray level or color is most homogeneous. This one is presumably contained within a single region, so that we can safely use a like-reinforces-like scheme on it [10-11].

2) If we do not want to commit ourselves to choosing a fixed set of neighbors, we can assign weights to the neighbors, or define link strengths between the pixels and the neighbors, such that neighbors similar to the pixel get high weights (or link strengths). In the interaction model, the reinforcement contributions are then proportional to the weights. For example, the m_{ij} factors in equation (2) can be chosen so as to give some neighbors more weight than others. If the reinforcement process is iterated, the weights can themselves be adjusted at each iteration, based on revised estimates of neighbor similarity -- e.g., the m_{ij} might change from one iteration to the next [12-14].

3) Finally, we can simply use the same neighborhood for every pixel, and let like reinforce like. For pixels in the interior of a region, this behaves as desired; but on the border of a region, the pixel labels are likely to remain ambiguous, since they are being influenced by neighbors that belong to two regions. In fact, sharp corners on the borders will be smoothed out, since a pixel at such a corner will have most of its neighbors in another region, so that the reinforcement process will make it confident that it too belongs to that region [15-16].

It should be pointed out that, rather than reinforce label confidences, we can adjust the pixel gray level or color values themselves; in other words, we can use the methods just described to smooth an image without blurring the edges between regions.

The preceding discussion assumed that each region is "flat", i.e., has relatively constant gray level or color. This is a reasonable assumption about some classes of scenes (e.g., characters on a printed page, chromosomes against a uniform background), but is not correct for others. More generally, the image can be modeled as piecewise linear and the relaxation process can examine a set of one-sided neighborhoods of the pixel, and choose the one that best fits a plane. Within the choosen neighborhood, the reinforcements should depend on closeness of fit to the hypothesized plane, rather than on similarity [17].

A similar approach applies, in principle, if we want to classify pixels based on the values of local properties measured over their neighborhoods, assuming that the image consists of regions that are homogeneously textured. Here, however, larger neighborhoods should be used, since local properties tend to be more variable than single-pixel properties. Of course, when we use large neighborhoods, the problems encountered at region borders become more severe.

3.2. Local feature detection

If the labelling task involves local feature detection or template matching, we must use neighborhood and interaction models appropriate to the type of feature or pattern being detected. In the following paragraphs we discuss the detection of spots (i.e., peaks), streaks (ridges, curves), edges, and corners, as well as matches to an arbitrary template.

To detect peaks, i.e., local maxima, the neighborhood must be large enough to contain a peak. The peak label at a pixel is then positively reinforced by the presence of lower values at its neighbors, and negatively reinforced by higher values, where the amounts of reinforcement depend on the differences in value. In other words, small reinforces large, while large competes with large. Detection of pits (local minima) is exactly analogous. The same approach can be used to detect peaks on waveforms, histograms, or scatter plots [18-19].

To detect ridges or ravines, i.e., high-valued lines or streaks on a low-valued background or vice versa, as in linear feature detection, we use a neighborhood somewhat larger than the streak width. The reinforcement model should now depend on the orientation of the streak; high values should reinforce one another along the streak, while low values should reinforce high values across the streak. To implement this, we initially estimate a streak confidence for each orientation, or more simply, estimate a single streak confidence and an associated streak direction. For neighbors in the direction along the streak, high values reinforce one another, provided the orientations are consistent; for neighbors in the direction across the streak, low values reinforce high ones, as in the case of peak detection [20-21].

Detecting edges is similar to detecting streaks, since an edge is a streak-like locus of high rates of change. Note that in this case directions must be measured modulo 360° rather than modulo 180°; in other words, for a given direction, we must take into account the sign of the rate of change, so that high edge values reinforce one another only if their dark sides and light sides match, and they compete otherwise. If desired, we

can associate edge values with the "cracks" between adjacent pairs of pixels, rather than with the pixels themselves; this is more appropriate if the edges are sharp [22-23].

For both edges and streaks, our model implicitly assumes that they are straight or smoothly curved; if they have sharp corners or angles, dissimilar directions will be present in a single neighborhood, and these will compete with one another. To detect corners, we must allow a pixel to interact with pairs of its neighbors, rather than with each neighbor separately; we can then reinforce the "cornerity" value of the pixel if there exist pairs of neighboring edge or streak values that have sharply different directions. [Alternatively, we can detect corners in the gray level (or color) domain based on the presence of suitable combinations of high and low levels at neighbors, e.g., a high value on one side and low values on several other sides; but this requires us to work with k-tuples of neighbors for $k > 2$.] At the same time, low cornerity values at the neighbors of a pixel should reinforce a high value at the pixel, just as in the case of peak detection. Analogous methods can be used to detect corners on ideal borders or curves represented by chain codes; here again, cornerity is reinforced by the presence of neighboring slopes that differ sharply, and by low neighbor cornerity [24].

In general, we can employ a cooperative approach to detect matches of a given template with the image by considering the template as composed of pieces; detecting matches with the pieces by some conventional method; and reinforcing a match to a given piece based on the occurrence of matches to the other pieces in approximately the correct relative positions. This approach is preferable to straightforward matching of the entire template for

two reasons: it has lower computational cost, and it is less sensitive to geometrical distortion. Note that in this reinforcement process, if there are matches to a given piece in several neighboring positions, we can use the best one, but we should not sum their influences, since only one of them can be correct; thus a reinforcement rule using the max, rather than the sum, is more appropriate here [25-26].

3.3. Processes involving multiple properties or multiple images

A more complex class of cooperating processes can be used to assign two or more interrelated sets of labels to the pixels in an image. As an example, consider the gray level and edge labelling processes discussed in Sections 3.1 and 3.2. These two sets of labels are not independent; for example, the gray levels on the light side of an edge are more likely to belong to a light than to a dark class, and vice versa. Thus we can design a compound cooperating process in which both sets of labels interact; such processes should yield better results than if we use either of the individual processes alone [27-28].

The gray level at a given pixel of an image is the resultant of several "intrinsic" properties at the corresponding point of the scene, including illumination, reflectivity, and surface slope. It is impossible to separate the effects of these factors by examining the pixels individually; but one can attempt to separate them using a cooperative process, based on assumptions about how the factors vary from point to point. For example, let us assume that the scene is composed of regions over which the intrinsic properties are constant. It may then be possible to determine which property is changing at a given edge, by analyzing the gray level variations at the edge. If this can be done, we can try to estimate the property values

cooperatively, by hypothesizing initial values and letting like reinforce like except across edges [29].

Finally, we consider cooperating processes that involve more than one image. Given two images of the same scene, taken at different times or from different positions, it will not be possible in general to register the images globally, since parts of the scene may have moved, or their projections on the image may have shifted by different amounts because they are at different distances from the sensor. However, we can try to match pieces of one image with pieces of the other to determine a piecewise correspondence; from the variations in this correspondence we can then estimate the motion or distance information. The accuracy of these estimates can be enhanced using a cooperative approach, if we assume that the scene is made up of parts each having a uniform motion or distance; the approach is analogous to that used in Section 3.1 [30-31].

3.4. Region-level processes

Cooperating processes can also be used to assign labels to windows or regions of an image, or to detect configurations of regions that match given models. Such processes are briefly discussed in the following paragraphs.

As a simple example, suppose that we have broken up an image into windows, and want to classify the textures in the windows. If we use small windows, the classifications become unreliable; but if we use large ones, border effects become a major factor, since it is hard to classify windows that overlap two or more differently textured windows. One solution is to use small windows, and adjust the classifications (or the feature values) cooperatively, based on those of neighboring windows, in such a way that

windows belonging to different regions (most likely) do not influence one another; this is analogous to the cooperative approach to pixel classification described in Section 3.1 [32].

More generally, suppose that we have segmented an image into regions, and want to classify the regions, based on their geometrical or textural properties. If we know what pairs of classifications are possible for neighboring pairs of regions in given relative positions, we can use this knowledge to cooperatively adjust the label possibilities or confidences. For example: (1) In labelling the edges in a blocks-world scene as convex, concave, or occluding, we can use the constraints imposed when the edges meet at junction [1]. (2) In assigning regions in an indoor scene to classes such as "door", "doorknob", "wall", and "light switch", the light switch label is reinforced, and the doorknob label weakened, by the wall label on a surrounding region, and vice versa for the door label on a surrounding region [33].

Matching a configuration of regions to a given model is analogous to matching a piece of an image to a template. We can represent the regions, their properties, and their relationships by a "scene graph" in which the nodes and arcs are labelled by property or relation names (and values). The model can be similarly represented by a labelled graph, and we can then attempt to find occurrences of the model graph as a subgraph of the scene graph. Just as in the template case, this can be done cooperatively by finding scene graph nodes that match model graph nodes, and reinforcing matches for which the proper neighboring nodes are present [34-35].

4. Issues

Relaxation processes have proved very useful for deriving relatively unambiguous labellings of image or scene parts at a variety of levels. The design and control of such processes, however, are not as yet well understood. Given a labelling task, how doe we choose appropriate neighborhood and interaction models? (In other words, how do we represent our knowledge about the given problem domain in the form of an interactive local process?) Given such a process, how many times should it be iterated, and how should its performance be evaluated? In this section we briefly review some of the approaches that have been proposed to these problems of knowledge representation and control in relaxation processes.

4.1 Knowledge representation

As mentioned in Section 2.2, for discrete relaxation processes the interaction model is defined by a set of compatible label pairs; but for fuzzy labellings, the compatibility relation must be quantitative. It can be defined, for example, by specifying a "compatibility coefficient" for each pair of labels on each pair of neighboring parts. These coefficients can be defined in a problem-specific manner; for example, the compatibility between two given edge or line directions at a pair of neighboring pixels could be taken as inversely proportional to the bending energy required to bend a spline so that it changes direction in the given way.

Another possibility is to define compatibilities on probabilistic grounds. Consider the probability ratio $r(d,d') \equiv p(d,d')/p(d)p(d')$, where the numerator is the joint probability of the pair of labels (d,d') on the given pair of neighboring objects, and the terms in the denominator are the

prior probabilities of the two labels. Intuitively, if d and d' are compatible, p(d,d') should be greater than p(d)p(d'); if d and d' are independent, they should be equal; and if d and d' are incompatible, p(d,d') should be less than p(d)p(d'). Thus we have r(d,d') > 1, = 1, and < 1 iff. d and d' are compatible, independent, or incompatible, respectively. If we want compatibilities that lie in the range [-1,1], we can use log r rather than r; this is poistive, zero, or negative according to whether d and d' are compatible, independent, or incompatible. (The log does not automatically lie in the range [-1,1]; if we want it to, it must be truncated and rescaled.) Note that log r is the mutual information of the pair of labels d, d'. The probabilities can be estimated by counting occurrences of d and d', and joint occurrences of both. The use of mutual information to define compatibilities is suggested in [36]. If we drop the restriction that the compatibilities lie in the range [-1,1], we can use r itself, rather than log r, as a compatibility function. In fact, in a Bayesian approach to relaxation developed by Peleg, the compatibility coefficients turn out to be the r's [37].

4.2. Control

A second critical question concerns the control of relaxation processes: when should the iteration be stopped? How can its progress be evaluated?

For a discrete relaxation process, termination criteria are straightforward to formulate and justify. For example, when binary (or higher-order) relations are used as a knowledge representation, then the process terminates when no further labels can be discarded from any picture part. At this point, each label at each picture part (if any remain) has a consistent label at every neighboring picture part. Or, if logic statements are used as a

knowledge representation, then the process terminates when no new inferences can be formed. In both cases, the destination of the process is a consistent labelling, the notion of consistency is well-defined and it is straightforward to prove that the relaxation process has as its "fixed point" a consistent labelling.

For a probabilistic relaxation process, the situation is more complicated. One possible approach is that the relaxation process should be iterated until the probability densities for each picture part converge. There are, however, both practical and theoretical disadvantages to this approach:

a) In practice, relaxation processes often converge to results which are quite poor, even though the first several iterations lead to significant improvements.

b) There are very few theoretical results concerning convergence, and these simply characterize sufficient conditions for convergence, rather than necessary conditions [38]. Moreover, the limit points have not been characterized as solutions to a well-defined problem, except in some specific cases [39].

Various criteria have been proposed for evaluating the performance of relaxation processes [40], but none of them seem to be satisfactory. Convergence (i.e., decrease in rate of change) is not an acceptable criterion, since the limit point may not be a desirable labelling. Unambiguity (e.g., low entropy) is also not acceptable, since there are many unambiguous labellings, most of which are highly inconsistent with the given initial labelling. Combinations of these criteria might be used [41], but these are also subject to similar objections [42]. A more promising approach uses a composite

criterion for evaluating a labelling based on its consistency with both the initial labelling and the model (i.e., the compatibilities) [43]. This area is still the subject of active research.

5. Concluding remarks

Relaxation processes have potential speed advantages because they can be implemented in parallel (hardware permitting). They have been successfully applied to a wide variety of labelling problems by a growing number of investigators; our survey makes no claim to completeness. In spite of these successes, little is as yet known about the design and control of these processes. However, a number of promising approaches to their theoretical formulation are being pursued, and it is hoped that a deeper understanding of their nature will soon be achieved.

References

1. D. Waltz, Understanding line drawings of scenes with shadows, in P. H. Winston, ed., The Psychology of Computer Vision, McGraw-Hill, NY, 1975, 19-91.

2. A. Rosenfeld, R. Hummel, and S. W. Zucker, Scene labelling by relaxation operations, IEEE Trans. Systems, Man, Cybernetics 6, 1976, 420-433.

3. R. M. Haralick, L. S. Davis, D. L. Milgram, and A. Rosenfeld, Reduction operators for constraint satisfaction, Information Sciences 14, 1978, 199-219.

4. R. M. Haralick and L. G. Shapiro, The consistent labelling problem: Part I, IEEE Trans. Pattern Analysis Machine Intelligence 1, 1979, 173-183.

5. L. S. Davis, in preparation.

6. L. S. Davis and A. Rosenfeld, Hierarchical relaxation for waveform parsing, in A. Hanson and E. Riseman, eds., Computer Vision Systems, Academic Press, NY, 1978, 101-109.

7. K. C. Hayes, Jr., Reading handwritten words using hierarchical relaxation, TR-783, Computer Science Center, University of Maryland, College Park, MD, July 1979.

8. L. S. Davis and T. C. Henderson, Hierarchical constraint processes for shape analysis, TR-115, Computer Sciences Dept., University of Texas, Austin, TX, November 1979.

9. L. S. Davis and A. Rosenfeld, Noise cleaning by iterated local averaging, IEEE Trans. Systems, Man, Cybernetics 8, 1978, 705-710.

10. F. Tomita and S. Tsuji, Extraction of multiple regions by smoothing in selected neighborhoods, IEEE Trans. Systems, Man, Cybernetics 7, 1977, 107-109.

11. M. Nagao and T. Matsuyama, Edge preserving smoothing, Computer Graphics Image Processing 9, 1979, 394-407.

12. A. Lev, S. W. Zucker, and A. Rosenfeld, Iterative enhancement of noisy images, IEEE Trans. Systems, Man, Cybernetics 7, 1977, 435-442.

13. A. Scher, F. R. D. Velasco, and A. Rosenfeld, Some new image smoothing techniques, IEEE Trans. Systems, Man, Cybernetics 10, 1980, in press.

14. J. O. Eklundh and A. Rosenfeld, Image smoothing based on neighbor linking, TR-773, Computer Science Center, University of Maryland, College Park, MD, June 1979.

15. J. O. Eklundh, H. Yamamoto, and A. Rosenfeld, Relaxation methods in multispectral pixel classification, IEEE Trans. Pattern Analysis Machine Intelligence 2, 1980, in press.

16. A. Rosenfeld and R. C. Smith, Thresholding using relaxation, submitted for publication.

17. R. M. Haralick and L. Watson, A facet model for image data, Proc. IEEE Conf. Pattern Recognition Image Processing, August 1979, 489-497.

18. L. S. Davis and A. Rosenfeld, Iterative histogram modification, IEEE Trans. Systems, Man, Cybernetics 8, 1978, 300-302.

19. S. Peleg, Iterative histogram modification, 2, IEEE Trans. Systems, Man, Cybernetics 8, 1978, 555-556.

20. R. Eberlein, An iterative gradient edge detection algorithm, Computer Graphics Image Processing 5, 1976, 245-253.

21. S. W. Zucker, R. A. Hummel, and A. Rosenfeld, An application of relaxation labeling to line and curve enhancement, IEEE Trans. Computers 26, 1977, 394-403, 922-929.

22. B. J. Schachter, A. Lev, S. W. Zucker, and A. Rosenfeld, An application of relaxation methods to edge reinforcement, IEEE Trans. Systems, Man, Cybernetics 7, 1977, 813-816.

23. A. R. Hanson and E. M. Riseman, Segmentation of natural scenes, in A. Hanson and E. Riseman, eds., Computer Vision Systems, Academic Press, NY, 1978, 129-163.

24. L. S. Davis and A. Rosenfeld, Curve segmentation by relaxation labelling, IEEE Trans. Computers 26, 1977, 1053-1057.

25. L. S. Davis and A. Rosenfeld, An application of relaxation labelling to spring-loaded template matching, Proc. 3rd Intl. Joint Conf. on Pattern Recognition, November 1976, 591-597.

26. S. Ranade and A. Rosenfeld, Point pattern matching by relaxation, Pattern Recognition, in press.

27. S. W. Zucker and R. A. Hummel, Toward a low-level description of dot clusters: Labeling edge, interior, and noise points, Computer Graphics Image Processing 9, 1979, 213-233.

28. A. Danker, Blob detection by relaxation, TR-795, Computer Science Center, University of Maryland, College Park, MD, July 1979.

29. H. G. Barrow and J. M. Tenenbaum, Recovering intrinsic scene characteristics from images, in A. Hanson and E. Riseman, eds., Computer Vision Systems, Academic Press, NY, 1978, 3-26.

30. D. Marr and T. Poggio, Cooperative computation of stereo disparity, Science 194, 1976, 283-287.

31. S. T. Barnard and W. B. Thompson, Disparity analysis of images, TR-79-1, Computer Science Dept., University of Minnesota, Minneapolis, MN, January 1979.

32. T. H. Hong, A. Y. Wu, and A. Rosenfeld, Feature value smoothing as an aid in texture analysis, TR-844, Computer ScienceCenter, University of Maryland, College Park, Maryland, December 1979.

33. H. G. Barrow and J. M. Tenenbaum, MSYS: A system for reasoning about scenes, TN-121, Artificial Intelligence Center, SRI, Inc., Menlo Park, CA, April 1976.

34. L. Kitchen and A. Rosenfeld, Discrete relaxation for matching relational structures, IEEE Trans. Systems, Man, Cybernetics 10, 1980, in press.

35. L. Kitchen, Relaxation applied to matching quantitative relational structures, IEEE Trans. Systems, Man, Cybernetics 10, 1980, in press.

36. S. Peleg and A. Rosenfeld, Determining compatibility coefficients for curve enhancement relaxation processes, IEEE Trans. Systems, Man, Cybernetics 8, 1978, 548-555.

37. S. Peleg, A new probabilistic relaxation scheme, IEEE Trans. Pattern Analysis Machine Intelligence 2, 1980, in press.

38. S. W. Zucker, Y. G. Leclerc, and J. L. Mohammed, Continuous relaxation and local maxima selection--conditions for equivalence, Proc. 6th Intl. Joint Conf. on Artificial Intelligence, August 1979, 1014-1016.

39. S. Ullman, Relaxation and constrained optimization by local processes, Computer Graphics Image Processing 10, 1979, 115-125.

40. G. Fekete, Relaxation: evaluation and applications, TR-796, Computer Science Center, University of Maryland, College Park, Maryland, August 1979.

41. O. Faugeras and M. Berthod, Scene labeling: an optimization approach, Proc. IEEE Conf. Pattern Recognition Image Processing, August 1979, 318-326.

42. S. Peleg and A. Rosenfeld, A note on the evaluation of probabilistic labelings, TR-805, Computer Science Center, University of Maryland, College Park, Maryland, August 1979.

43. S. Peleg, Monitoring relazation algorithms using labeling evaluation, TR-842, Computer Science Center, University of Maryland, College Park, Maryland, December 1979.