Image Smoothing and Segmentation by Multiresolution Pixel Linking: Further Experiments and Extensions

TSAI-HONG HONG, K. A. NARAYANAN, SHMUEL PELEG, AZRIEL ROSENFELD, FELLOW, IEEE, AND TERESA SILBERBERG

Abstract—**A recently developed method of image smoothing and segmentation makes use of a "pyramid" of images at successively lower resolutions. It establishes links between pixels at successive levels of the pyramid; the subtrees of the pyramid defined by these links yield a segmentation of the image into regions over which the smoothing takes place. This paper investigates several variations on the basic linking process with regard to such factors as initialization, criteria for linking, and iteration scheme used. It also studies generalizations in which the links are weighted rather than forced, and in which interactions among the pixels at a given level are also allowed. Finally, it extends the approach to links based on more than one feature of a pixel, e.g., on color components or local property values.**

I. Introduction

SUPPOSE that an image is composed of a few types of regions each having approximately constant gray tone. In principle, the image can be segmented into these regions by gray tone thresholding, i.e., by slicing the grayscale into intervals, and classifying each pixel according to the interval in which its gray tone lies. However, if the image is noisy, this pixel-by-pixel segmentation process may make many errors, since the noise will cause some of the pixels belonging to one type of region to have gray tones lying in the intervals corresponding to another type. Segmentation could become more reliable if we first smoothed the image to reduce its noisiness.

An image can be smoothed by local averaging, i.e., averaging the gray tone of each pixel with the gray tones of a set of its neighbors. However, this process will blur the boundaries between the regions, since a pixel near such a boundary has neighbors lying in both its own region and the adjacent region. If we knew which neighbors belonged to the same region as the pixel, we could use only these neighbors in the average. In other words, the quality of the smoothing process would be improved if we could first segment the image into the appropriate regions, so that smoothing could be performed within the regions only, not across their borders.

These remarks suggest that it might be preferable to perform smoothing and segmentation concurrently, using some type of cooperative process. An example is the combined smoothing and neighbor linking process defined in [1]. Here weights are assigned to the links between a pixel and its neighbors based on their similarity; the image is smoothed by weighted averaging of each pixel with its highest weighted neighbors; concurrently, the weights are adjusted as the similarities between neighbors change. The process is iterated, with weighted averaging and weight adjustment alternating. Note that this process does not involve classification of the pixels, but does yield a segmentation of the image into regions based on the connectedness relation defined by the links, if we threshold their weights.

This paper deals with another approach to concurrent smoothing and segmentation based on linking, using versions of the image at different resolutions and defining links between overlapping "pixels" at successive resolutions. In a low-resolution image, the pixels interior to regions have gray tones that are less noisy, since a pixel at low resolution represents an average and is thus less variable. On the other hand, the lower the resolution, the less likely it is that a pixel is contained in a single region; most pixels will overlap two or more regions. The approach considered here, which was first described in [2], takes advantage of both high and low resolutions by using a cooperative process in which the images of successive resolutions interact. A description of this approach will now be given; see [2] for further details.

II. Multiresolution Pixel Linking

Let the size of the original image be 2^n by 2^n. To define the reduced-resolution versions of the image, we make use

Manuscript received January 12, 1981 and April 3, 1981; revised April 9, 1982. This paper is based on three technical reports: TR-977, "Multiresolution pixel linking for image smoothing and segmentation" (Silberberg, Peleg, Rosenfeld; November 1980); TR-989, "Iterative image smoothing and segmentation by weighted pyramid linking" (Narayanan, Peleg, Rosenfeld, Silberberg; December 1980); and TR-1025, "Multiband pyramid linking" (Hong, Rosenfeld; March 1981). This work was supported in part by the Defense Advanced Research Projects Agency and the U.S. Army Night Vision and Electro-Optics Laboratory under Contract DAAG-53-76-C-0138 (DARPA Order 3206) [TR's 977 and 1025], and by the National Science Foundation under Grant MCS-79-23422 [TR-989].

T. H. Hong, A. Rosenfeld, and T. Silberberg are with the Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD 20742.

K. A. Narayanan is with the Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD 20742, and the ISRO Satellite Centre, Bangalore, India.

S. Peleg was with the Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD. He is now with the Department of Computer Science, the Hebrew University of Jerusalem, 91904 Jerusalem, Israel.

0018-9472/82/0900-0611$00.75

of an exponentially tapering "pyramid" of arrays of sizes 2^{n-1} by 2^{n-1}, 2^{n-2} by $2^{n-2}, \cdots$, 4 by 4, 2 by 2, so that the kth level has size 2^{n-k} by 2^{n-k}. To avoid border effects, all these arrays are regarded as cyclically closed, i.e., the first column is regarded as lying to the right of the last column, and the top row below the bottom row. The elements of each array will be called pixels or nodes. Many different schemes can be defined for constructing such pyramids [3], but in our experiments we used only the simple scheme that will now be described.

We will assign gray tones to the nodes at each level ($k > 0$) by taking (weighted) averages of the gray tones of 4-by-4 blocks of nodes at the level below it. The blocks corresponding to adjacent nodes overlap by 50 percent; this is why the reduction in area from level to level is by a factor of four, not a factor of 16. For example, suppose node (i, j) at level $k > 0$ corresponds to the block of nodes

$$\begin{array}{cccc} (u, v) & (u+1, v) & (u+2, v) & (u+3, v) \\ (u, v-1) & (u+1, v-1) & (u+2, v-1) & (u+3, v-1) \\ (u, v-2) & (u+1, v-2) & (u+2, v-2) & (u+3, v-2) \\ (u, v-3) & (u+1, v-3) & (u+2, v-3) & (u+3, v-3) \end{array}$$

at level $k-1$ (where $(u, v) = (2i-2, 2j+1)$). Then node $(i+1, j)$ corresponds to the block

$$\begin{array}{cccc} (u+2, v) & (u+3, v) & (u+4, v) & (u+5, v) \\ (u+2, v-1) & (u+3, v-1) & (u+4, v-1) & (u+5, v-1) \\ (u+2, v-2) & (u+3, v-2) & (u+4, v-2) & (u+5, v-2) \\ (u+2, v-3) & (u+3, v-3) & (u+4, v-3) & (u+5, v-3) \end{array}$$

where all additions and subtractions are modulo 2^{k-1}. It is easily seen that any node (u, v) below the top level (i.e., $k < n-1$) belongs to four blocks corresponding to nodes on the level above it—in our example, the nodes (i, j), $(i-1, j)$, $(i, j+1)$, and $(i-1, j+1)$. [Note that only for the last of these nodes does (u, v) belong to the center 2-by-2 portion of its block; for the other three, (u, v) is a border point of their blocks.] The level $k-1$ nodes in the block corresponding to a given node at level k will be called its *sons*, and the level k nodes to whose blocks a given node at level $k-1$ belongs will be called its *fathers*. Thus every node at level > 0 has 16 sons, and every node at level $< n-1$ has four fathers. Note that since there are only 16 nodes at level $n-2$, each of them is a son of all four nodes at level $n-1$, so that every node in the pyramid is a descendant of every one of these "top" nodes.

The node linking process is as follows: the reduced resolution images are initially defined by unweighted averaging of the gray tones in each block. The value (gray tone) of each node is then compared with the values of its four fathers, and a link is established between the node and its most similar father, i.e., the father whose value is closest to the node's value. After this has been done at every level, we recompute the value of each father by averaging only those sons that are linked to it. (If no sons are linked to a father, we give it the value zero.) Based on these new averages, a node's most similar father may have changed, so we next change the links as necessary, then recompute the averages, then change the links again, and so on.

The relinking process at any given level of the pyramid has the effect of shifting pixels from one class (= father) to another, where pixel P shifts from class A to class B only if its value is closer to the mean of class B than to that of class A. From this viewpoint, we see that relinking is a special case of the well-known ISODATA clustering algorithm in one dimension, and it can be shown [4] that this process is guaranteed to converge. In fact, in our experiments, the relinking process always stabilized after a few iterations.

To see what this process does, let us define the *base* of a node as the set of pixels on the lowest level (i.e., in the original image) that are linked (through as many intermediate stages as necessary) to that node. Thus initially the base of every node is a square block of image pixels. If the base of a node initially lies mostly inside a region, the node is most likely to become linked to nodes on the level above that also lie (mostly) in that region; thus its recomputed average will become closer to the region average. As the process is iterated, nodes at relatively high levels acquire values that approach the average values of regions, even though they are too large to fit into a region. Slight initial biases in the node averages at high levels will result in high-level nodes being driven toward values that correspond closely with the averages of regions or sets of similar regions in the image. For further discussion of the process, see [2].

Suppose that there are not more than four types of regions in the image. For each type, there should be at least one node at the top level of the pyramid whose value converges to the average gray tone of the regions of that type. This node will be linked to nodes which are linked to nodes... which are linked to the pixels belonging to these regions. In other words, this node becomes the root of a tree whose leaves are the pixels that lie in regions of the given type. If there are fewer than four types of regions, there may be two such trees corresponding to the same region type, representing different subsets of the pixels in these regions. If we know how many region types there are supposed to be, we can suppress some of the nodes at the

top level (i.e., forbid anyone to link to them), keeping only as many top-level nodes as there are types. (Alternatively, we can "merge" some of the top-level nodes together, averaging together their values and using this average as the value for each of them.) In this way, we can insure that the number of trees (having distinct values) is the same as the desired number of region types. The method breaks down if there are more than four types of regions; pixels belonging to two or more types would then be forced to belong to the same tree.

In summary, the iterative linking and averaging process is defined as follows.

a) Initialize the node values by simple block averaging of each node's 16 sons.
b) Link each node to that one of its four fathers whose value is closest to its own.
c) Recompute the node values by averaging the values of only those sons that are linked to the node.
d) Change the links in accordance with these new values.
e) Repeat steps c)–d) as many times as desired. Typically, there is little change after the first few iterations.

At any stage of this process, the links define a set of (up to four) trees rooted at the top level of the pyramid, and we associate with each pixel the value at the root of its tree. Thus the process smooths the image to an extreme degree, giving each pixel its tree average as a smoothed gray tone. At the same time, it segments the image into (up to four) subsets, where each subset consists of the pixels which are the leaves of one of the trees.

The smoothing and segmentation accomplished by this process can be compared with those achieved by the pixel linking process of [1]. In [1] the links are all at the pixel level, and the smoothing is local. Even if the link strengths all converged to values of one (within a region) and zero (between regions), many iterations would be required to obtain the global average of each region at each pixel of the region, since it takes O (region diameter) iterations for information to propagate across the region. In the process described here, on the other hand, the links are between levels, and information can propagate "across" a region in O (log region diameter) iterations, since nodes comparable in size to the region are only (log region diameter) levels above the pixel level. Moreover, in our process smoothing can take place even over sets of nonconnected regions of the same type, whereas the process of [1] can smooth only within a connected region.

The concept of linking each node with its most similar father can also be compared with the smoothing processes described in [5]–[6], where a set of neighborhoods lying on various sides of a pixel are examined, and the pixel's value is replaced by the average of the least variable of these neighborhoods (since this neighborhood presumably lies almost entirely within the pixel's region). A generalization [7] fits linear functions to the neighborhoods, and uses the neighborhood for which the fit is best; the pixel's value is then replaced by the value of that neighborhood's linear function at the pixel's position. These methods all use neighborhoods of only a single size, which limits the speed with which the smoothing can propagate, as already mentioned. Note also that in our approach, we have used the most similar neighborhood (i.e., the one whose average is closest to the pixel's gray tone), rather than the least variable neighborhood.

(a) (b) (c)

Fig. 1. Three images used in experiments. (a) Tank. (b) Blood cells. (c) Chromosomes.

III. Variations

In the experiments described in this section, several variations on the basic pyramid linking process were tried. These variations were concerned with how to initialize the node values; how to choose the father to which a node is linked, and in particular what to do in case of ties; and how the iteration process is sequenced. In the following paragraphs we describe the variations, and then show the results obtained by using combinations of these variations on a standard set of images (which were also used in [2]): an infrared image of a tank, a portion of a blood smear, and a portion of a chromosome spread. These images are shown in Fig. 1. All results are shown for a stage at which the iteration process has stabilized; this is usually after about ten iterations.

a) *Initialization*: In the method used in [2], the value of each node was initialized by averaging the values of all 16 of its sons. An alternative, which (as we shall see) seems to give better results, is to initialize by averaging the values for only four of the sons, namely those whose positions in the image are closest to that of the node. (The position of a node is understood to be at the center of its block.) Note that in this alternative scheme, the initial averages are all nonoverlapping.

b) *Father Selection*: In the method of [2], each node is linked to the father closest in value to the node. A more general idea is to take into account both closeness in value and closeness in position. We can compute link strengths based on a formula such as $\Delta(D + s)$, choosing the father for which $\Delta(D + s)$ is smallest, where Δ is the difference in value, D is the Euclidean distance between positions, and s is a parameter which is used to vary the effect of the D contribution (for large s, differences in D have little effect). There is no obvious basis for choosing the value of s; various values were tried in our experiments.

b') *Ties*: If two fathers have the same link merits, we resolve the tie based on any arbitrary ordering of the fathers, e.g. NW, NE, SE, SW. The choice of this ordering should not significantly affect the results.

c) *Sequencing*: In [2], links are determined for all levels; then averages are recomputed for all levels; and this pro-

cess is repeated. An alternative is to iterate level by level: as soon as the links from the nodes at level k are redefined, the averages at level $k + 1$ are recomputed, and the links from level $k + 1$ are then redefined based on these new averages. This alternative seems to be more reasonable; when we have new values at level 1 based on the links between levels 0 and 1, we should use these new values as inputs to the new values at level 2, and so on.

d) *Top Level Nodes*: The number ($\leqslant 4$) of nodes used at the top level should be the same as the desired number of region types—2 for the tank and chromosomes, 3 for the bloodcells. We can insure that only two or three nodes at the top level are used by initializing the values of the remaining node(s) to a very high number, thus insuring that no nodes will ever link to them. As a refinement, we can fix the top-level nodes that we do use to have values that represent estimates of the expected region averages; we will show some examples using this variation. We will also show examples of results obtained when we use all four nodes at the top level, even though the desired number of region types is less than four. As we shall see, the process then tends to create somewhat artificial discriminations within the regions.

We first show the results obtained when we use the desired number of nodes at the top level, but do not attempt to set the values of those nodes to the expected region averages. Fig. 2 (top two rows) shows these results for the four combinations of initialization and sequencing schemes. We see that in the chromosome case (Fig. 2(c)), four-son initialization gives better results; when 16-son initialization is used, some of the small chromosomes are lost, probably because too much of the background is initially averaged with them, so that they link to a top-level node whose value converges to the background value rather than to the chromosome value. The initialization scheme has little effect on the results for the other two images, and the iteration sequencing scheme has little effect on any of the images. The order used for tie-breaking also has little effect, as we see from the bottom left pictures in Fig. 2 (which use the same initialization and sequencing schemes as the top left pictures). Finally, the bottom right pictures in Fig. 2 show what happens when we give some weight to Euclidean distance ($s = 5$) in choosing the links (otherwise, same as top left); note that this too improves the results in the chromosome case, and has little effect in the other two cases. It seems from these results that four-son initialization is preferable to 16-son initialization, and that it may also be preferable to give some weight to Euclidean distance in choosing links; but the other variations make little difference. The exact shapes of the tank and cell nucleus are somewhat sensitive to variations because the correct links for blocks near the borders of these regions will be somewhat ambiguous, due to the noisiness or texturedness of the regions.

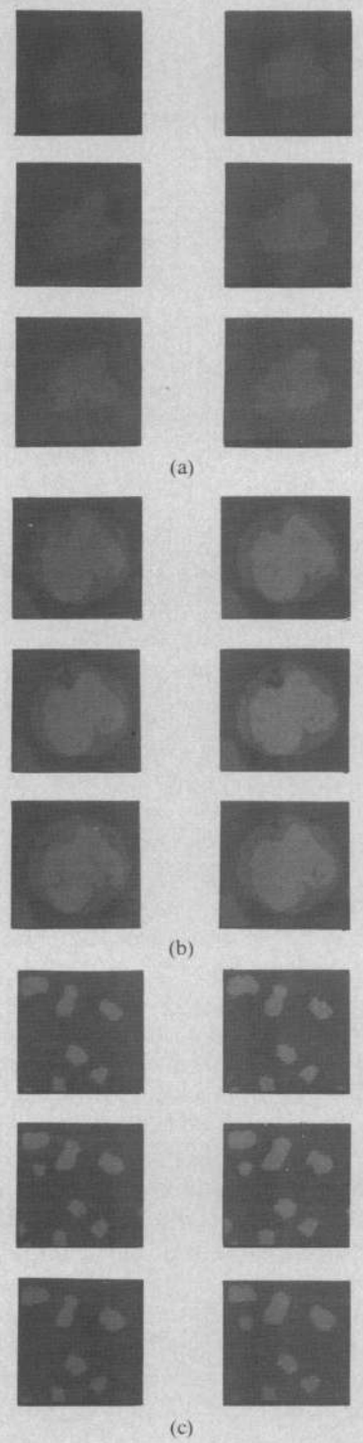


Fig. 2. Effects of varying the initialization, sequencing, tie-breaking rule, and linking criterion. In all cases, the number of nodes used at the top level is equal to the desired number of region types. Top row: initialization using averages of 16 sons (left: iteration for all levels at once; right: iteration level by level). Middle row: analogous, but initializing using averages of the four sons closest in position. Bottom left: same as top left, but using a different tie-breaking order. Bottom right: same as top left, but using a linking criterion that depends on Euclidean distance as well as on difference in value.

Fig. 3 shows analogous results when the top-level nodes are given estimates of the average region gray levels as fixed values. Again, the variations make little difference for the tank and cell images, but they are significant for the chromosome image. The loss of the small chromosome has now become dependent on the iteration sequence and even on the tie-breaking order (!); and when we use the four-son initialization method, a large chromosome is lost (!). Ap-

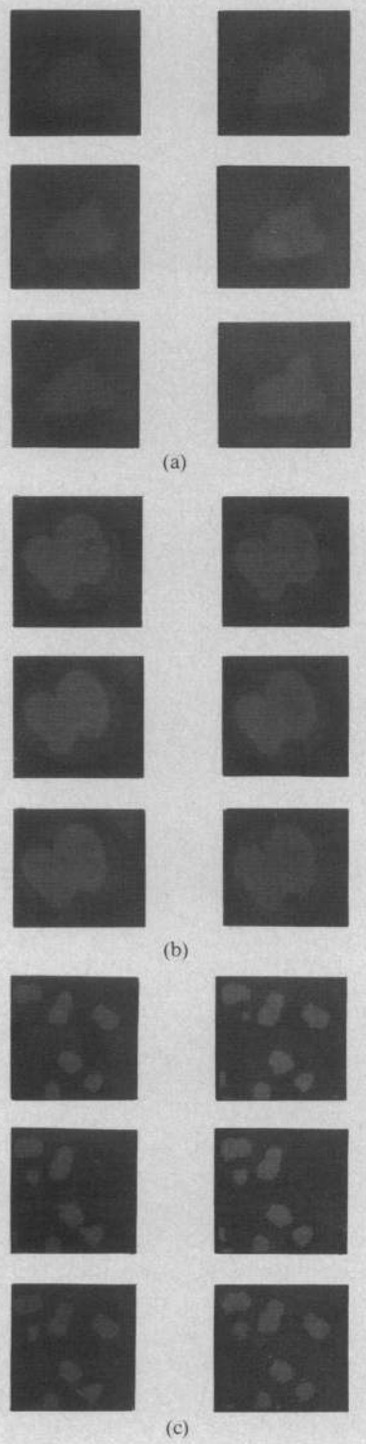


Fig. 3. Analogous to Fig. 2, but initializing the top-level nodes with estimates of the region averages.

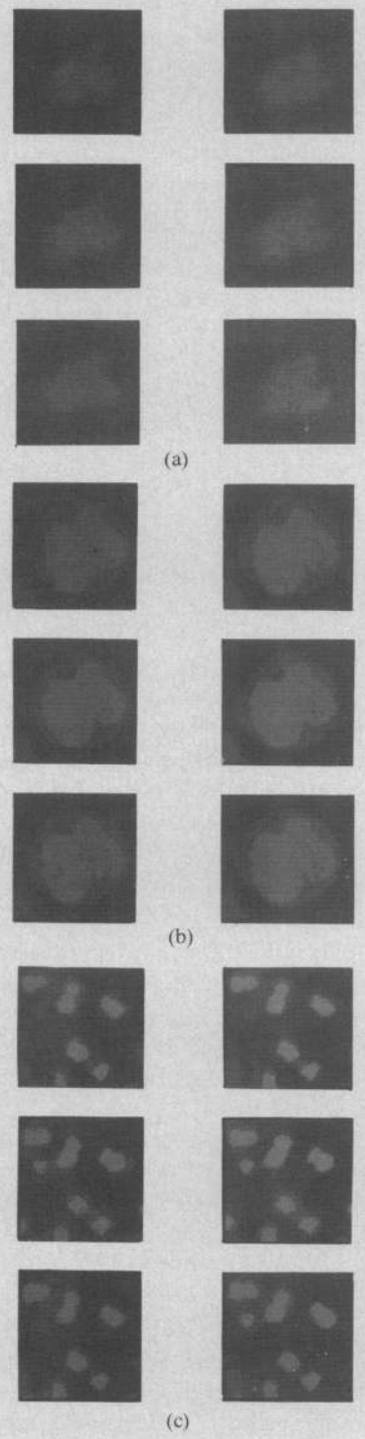


Fig. 4. Analogous to Fig. 2, but using all four nodes at the top level.

parently, attempting to fix the values of the top-level nodes as equal to the estimated region averages can actually degrade the performance of the pyramid linking process.

Fig. 4 gives analogous results when all four top-level nodes are used, so that the process tries to find four region types in each image.[1] The resulting artifacts are especially apparent for the chromosome image, where the background gets segmented into three subregions that differ appreciably in average gray tone. Here again, when we use 16-son initialization, the small chromosomes become part of the background, but this does not happen when four-son initialization is used, nor when weight is given to Euclidean distance in choosing links. The other variations have little

[1]If the input image contains fewer than four gray tones (e.g., if we threshold the chromosome or tank image into two levels or the cell image into three), the process (after convergence) turns out to yield an image with only two or three gray tones, even though all four top-level nodes are used.

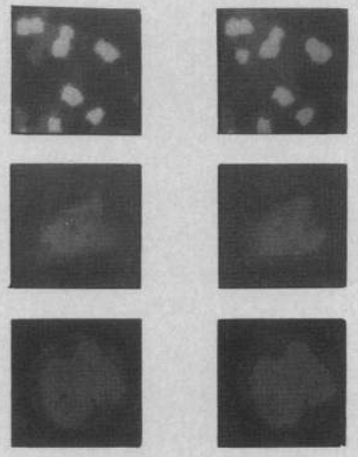


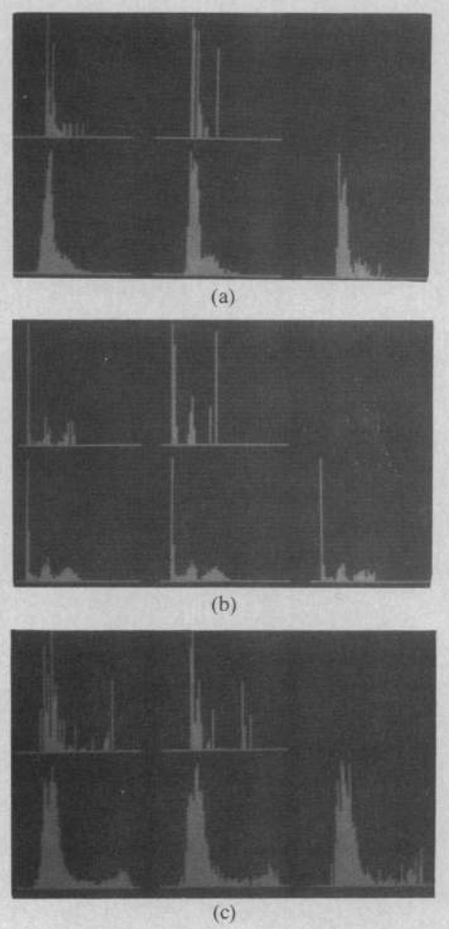
Fig. 5. Analogous to the top left and bottom right pictures in Fig. 2, but breaking ties randomly.

effect, and all of the effects are minor for the cell and tank images (the tank region splits up into "noisy" subregions in various ways, but does not get badly confused with the background). Thus these results support the conclusions derived from Fig. 2.

When the process is applied to a perfectly regular input pattern such as a checkerboard, it breaks down and fails to segment the pattern into two region types, unless ties are broken *randomly*. Fig. 5 shows results analogous to those in Fig. 4 (left column corresponds to Fig. 4 top left, and right column to bottom right), but using random tie-breaking; the results are quite similar.

The smoothing effect of the process as we follow the links from level to level can be assessed by constructing histograms corresponding to each level's view of the image. Suppose that, for a given k, we give each pixel a gray tone equal to the value of the node at level k to which it is linked. When we do this for $k = 0, 1, 2, \cdots$, we obtain a sequence of successively smoother and simpler images, whose histograms become successively more spiky, until finally, the histogram obtained from the top level consists of (at most) four spikes. Such histograms for the three images, after one iteration of the linking and reaveraging process, are shown in Fig. 6 for levels $0, 1, 2, 3, 4$. (16-son initialization was used, and links were chosen based on value similarity only.) If we did not want to rely on the iterative process to converge to a good segmentation, we could still consider using a single iteration of the process to improve the separation of the histogram peaks, so that segmentation by thresholding based on the histogram would be easier.

Fig. 7 shows how the process performs on a synthetic image composed of disk-shaped regions having different normal distributions of gray tones. In this image the objects and background have (approximately) normally distributed gray tones with standard deviation 5 (on a grayscale of 0–63); the mean object gray tone is 15 in every case, and the mean background gray tone its 50, 45, 40, 35, 30, and 25, respectively, in the six parts of the figure (top to bottom). The left column shows the input image, the



(a) (b) (c)

Fig. 6. Histograms obtained, after one iteration of linking and reaveraging, when the node values at a given level are assigned to the pixels having those nodes as ancestors, for levels $\begin{matrix} 3 & 4 \\ 0 & 1 & 2 \end{matrix}$.

center column its histogram, and the right column shows the results using the basic linking process of [2], with two classes. We see that even in the last case, where the input histogram is unimodal, the process (noisily) extracts the objects.

A basis for quantitatively evaluating the results of the pyramid linking process (or any other image smoothing process) is suggested by the work of Martelli and Montanari [8]. They point out that a good smoothing process should yield an image that is both smooth and similar to the original image. This suggests that we can evaluate smoothing results in terms of a cost function that measures the "roughness" of the smoothed image (e.g., by the sum of its absolute gradient magnitudes) and its discrepancy from the original image (e.g., by the sum of their absolute pointwise differences). Table I shows the values of these cost components for the images in Fig. 7 (the gradient magnitudes are estimated using the Sobel operator). As we might expect from visual inspection of Fig. 7, the roughness decreases in all cases, while the discrepancy is not large, so that even the sum of roughness and discrepancy decreases (the discrepancy is initially zero). Note that the linking process was not specifically designed to minimize the cost function; we are using the function only as an ad hoc

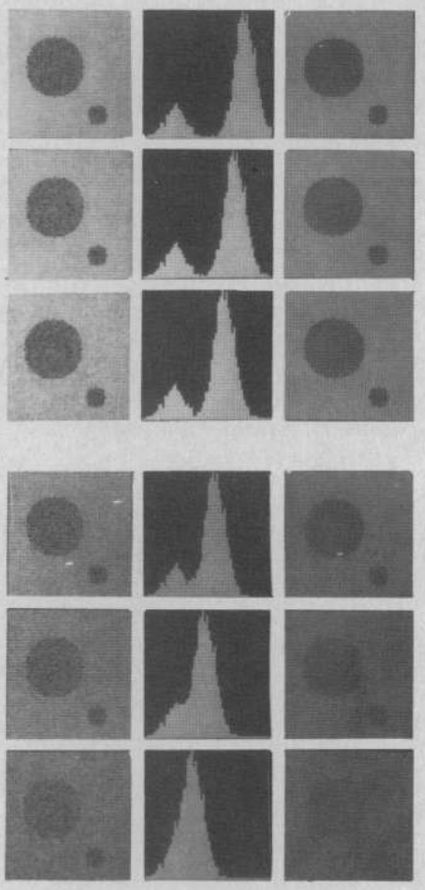


Fig. 7. Results of applying the basic process of [2] to a set of synthetic images (see text). Left column: input images; center column: their histograms; right column: outputs (each pixel displayed with the gray tone at the root of its tree).

TABLE I
ROUGHNESS AND DISCREPANCY VALUES FOR THE CASES IN FIG. 7.

Background Mean	Input Roughness	Output Roughness	Discrepancy	Output Roughness + Discrepancy
50	95986	19635	16334	35969
45	95801	19167	16377	35544
40	95211	18726	16373	35099
35	94202	21261	16332	37593
30	92369	32605	16408	49013
25	89370	58502	14799	73301

evaluation measure. It should be pointed out that the object/background edges influence the roughness measure (note the decline in input roughness value as the contrast at these edges decreases), but we have ignored this fact since only a small fraction of the pixels lie on edges.

IV. WEIGHTED LINKING

The construction of the linked pyramid as described up to now involves forced choices; each node is required to link to one of its candidate father nodes on the level about it, namely the one to which it is most similar (and/or closest). An alternative would be to use weighted links, where the link weights between a node and its candidate fathers depend on their degrees of similarity. As we shall now see, when the link weighting function is appropriately defined, the weighted linking process causes the weights to converge to values of zero and one, thus yielding a segmentation of the image (based on the trees of linked nodes) similar to that obtained in the forced choice scheme. Another possibility is to use horizontal interactions of nodes within each pyramid level to adjust the link weights; this also yields good results, as will be seen in the next section.

Initially, as in the basic method, a value $g(i, j, l)$ is assigned to each node (i, j) at level l by unweighted averaging of the values of its sons, i.e.,

$$g(i, j, l) = \frac{1}{16}\Sigma g(i', j', l-1).$$

The pyramid is built up to level $l = N - 1$, which is 2×2.

Weights are assigned to the four fathers of each node depending on their similarity to that node. For $k = 1, 2, 3, 4$, let $d(k)$ be the absolute difference between the values of the node and its kth father; then we define the link weight $w(k)$ between the node and that father by

$$w(k) = \frac{1/d(k)^2}{\Sigma_{h=1}^{4} 1/d(h)^2}.$$

Note that these weights are nonnegative and their sum is 1. If any $d(k)$ is zero, we set $w(k) = 1$, and $w(h) = 0$ for $h \neq k$; if more than one $d(k)$ is zero, we use an arbitrary tie-breaking rule.

After the link weights have been computed, the node values are recomputed as weighted averages, i.e.,

$$g(i, j, l) = \frac{\Sigma w(i', j', i, j) g(i', j', l-1)}{\Sigma w(i', j', i, j)}$$

where $w(i', j', i, j)$ is the link weight between node $(i', j', l-1)$ and its potential father node (i, j, l). We can now recompute the difference $(d(i', j', i, j) = |g(i, j, l) - g(i', j', l-1)|)$ and the weights, then recompute the values using these new weights, then recompute the weights again, and so on.

Since there are four nodes (2×2) at the top level of the pyramid, four trees of high-weight links tend to be formed, yielding a segmentation of the input image into four classes. If we know the desired number of classes (≤ 4), we can eliminate some of these top nodes (e.g., by fixing their values to something very dissimilar to the gray levels in the image, so that the weights of links to these nodes will be very weak), thus yielding a segmentation into fewer classes.

Fig. 8 (bottom row) shows the results of applying 11 iterations of this iterated weighted linking process to the three test images (top row). Two nodes were used on the top level for the tank and chromosome images, and three for the blood cell image, thus yielding segmentations into two and three classes, respectively. Fig. 9 (top row) shows results when all four nodes are used on the top level; this yields four classes for each image, but these classes are compatible with the desired segmentation. The second and third rows of Fig. 9 show the results of variations on the

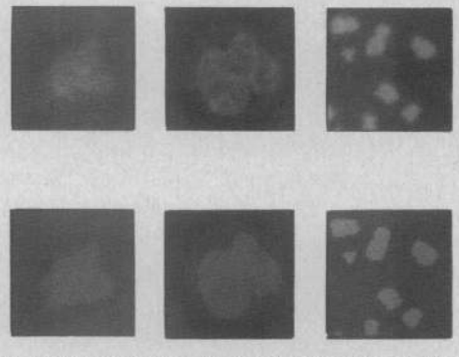


Fig. 8. Three test images (top row) and the results of 11 iterations of weighted linking, using two nodes at the top level (three, for the blood cell image).

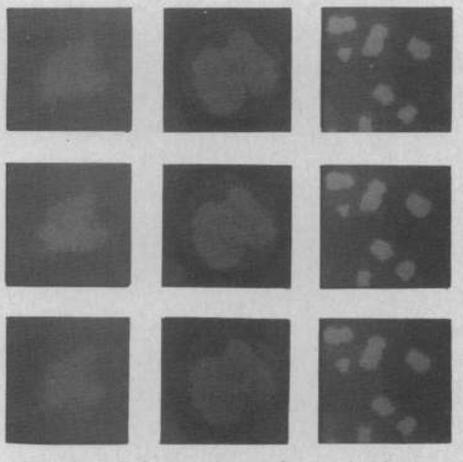


Fig. 9. Results using all four nodes at the top level (top row), giving weight to difference in position as well as difference in value (middle row), and using a different sequencing of the iteration process (bottom row).

node difference computations (giving some weight to difference in position as well as to difference in value) and on the sequencing of the iteration process, as in Section III; the results are very similar to those in the top row.

These results show that the weighted linking process is not sensitive to the variations investigated in Section III, at least for these test images. A more critical issue is the choice of the weighting function itself. Our results were obtained using the inverse quadratic weighting function $w(k) = d(k)^2/\sum d(h)^2$. If we use an inverse linear function, $w(k) = d(k)/\sum d(h)$, the weights do not converge to zeros and ones; rather, they all tend to become equal, so that the images do not get segmented, and they become smoothed into a uniform shade of gray. (This can be overcome if we fix the values of the top-level nodes to the mean values of the classes expected from the segmentation; but as was seen in Section III, when this is done, the process becomes sensitive to noise, and in particular some of the chromosomes merge into the background.) In the next section we will see that good results can be obtained even from inverse linear weighting if we use a within-level node interaction process to adjust the weights.

The quadratic weighting is more forced-choice-like than the linear function, in the sense that it tends to exaggerate slight differences in link weights. Evidently, if we used sufficiently high powers of the differences, the results would be tantamount to forced choice: the smallest difference would dominate over the others. In our implementation, which used weights of limited precision (7 bits), this effect was further strengthened, since the reciprocals of large differences were truncated to zero. We have no rigorous proof that any particular weighted linking scheme must converge; but in our examples, we always did obtain convergence.

V. Within-Level Interaction

In the (weighted) pyramid linking process, all interactions between nodes are "vertical," between nodes at consecutive levels; there is no direct interaction between nodes at the same level. There seems to be some advantage in allowing such interactions. For example, as we shall now see, good results can be obtained using inverse-linear link weighting if within-level interaction is used to adjust the link weights or the node values.

The basis for using within-level interaction is that, particularly at the lower levels of the pyramid, neighboring nodes often belong to the same region of the image. Thus we may be able to obtain more accurate node values or link weights by using some type of smoothing that favors neighboring nodes which belong to the same region as the given node.

Two well-known smoothing methods that favor neighbors belonging to the same region are median filtering and selective averaging. In median filtering, the value at a point is replaced by the median of the neighbors' values. If the point P lies in the interior of a region, the median is close to the mean; while if it lies near the border of a region, the median tends to come from one of the neighbors that belongs to the same region as P. In selective averaging [9], the value at P is averaged with the values of its r neighbors whose values are closest to P's (we used $r = 2$ and 4 in our experiments). These r neighbors tend to belong to the same region as P, even if P lies near a region border; thus this process too tends to smooth within regions without blurring their borders.

Fig. 10 shows the results of using selective averaging with $r = 2$ to smooth the node values before each iteration of the (inverse-linear) weighted linking process. This was done at levels 0 through l for $l = 0, 1, 2, 3, 4$. The results for the tank and blood cell images are generally good; but the chromosome results are somewhat sensitive to the number of levels to which the smoothing was done (they are best for $l = 2$, though the chromosomes are distinguished from the background for $l = 3, 4$ also). Fig. 11 shows analogous results for selective averaging with $r = 4$ and $l = 0, 1, 2$; here again, the chromosome results are best for $l = 2$. (When $r = 4$ is used, the results deteriorate for $l > 2$; this is because at higher levels, it is unlikely that a node has as many as four neighbors that belong to the same region that it does.) Fig. 12 shows analogous results for median filtering ($l = 0$ only); they are not very good for the chro-

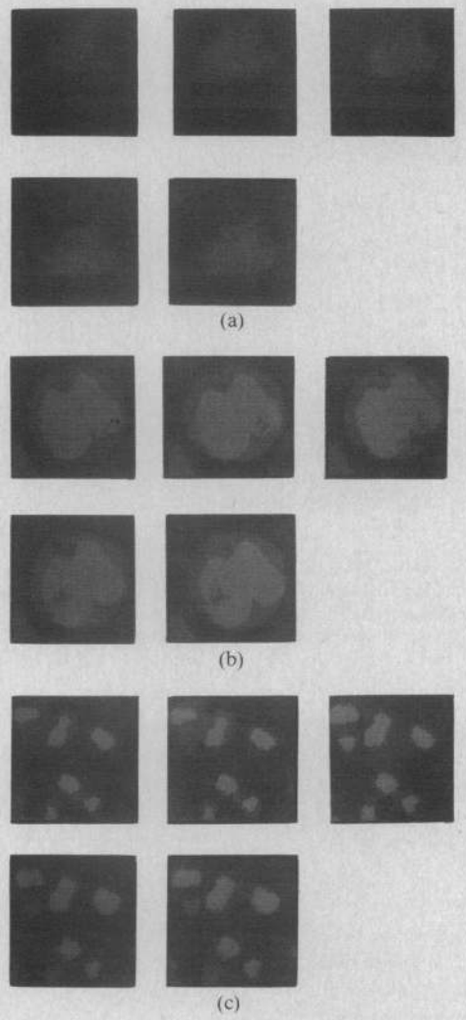


Fig. 10. Results using two-neighbor selective averaging to smooth the node values before each iteration of weighted linking, at each of levels 0 through l, where $l = 0, 1, 2, 3, 4$.

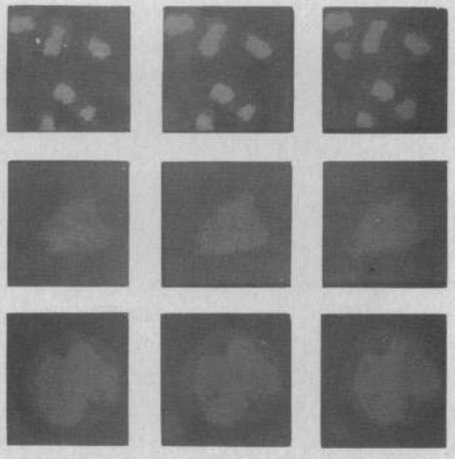


Fig. 11. Results using four-neighbor selective averaging for $l = 0, 1, 2$.

mosome image, and they get worse for $l > 0$ (for similar reasons).

Similar results are obtained if we use within-level node interaction to smooth the link weights, rather than the node values. Here, in adjusting the weight of the link between a node P and one of its candidate fathers, we consider only those neighbors of P that have the same candidate father. For example, to smooth the link weight from node P to candidate father Q by selective averaging, we examine the neighbors of P that also have father Q; pick the r of these neighbors $P_1, \cdots, P_r$ whose values are closest to that of P; and average the link weight from P to Q with the link weights from $P_1, \cdots, P_r$ to Q. (Note that when this is done, the link weights from P to its four candidate fathers may no longer sum to one, and must be renormalized.) Fig. 13 shows the results of doing this for $r = 2$ (at levels 0 through l, for $l = 0, 1, 2, 3, 4$) at each

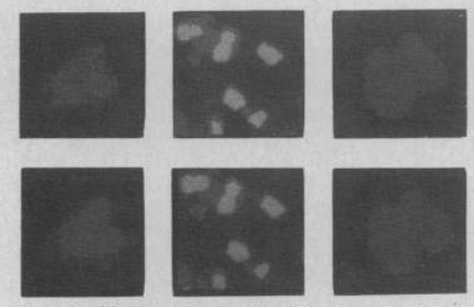


Fig. 12. Results using median filtering ($l = 0$ only).

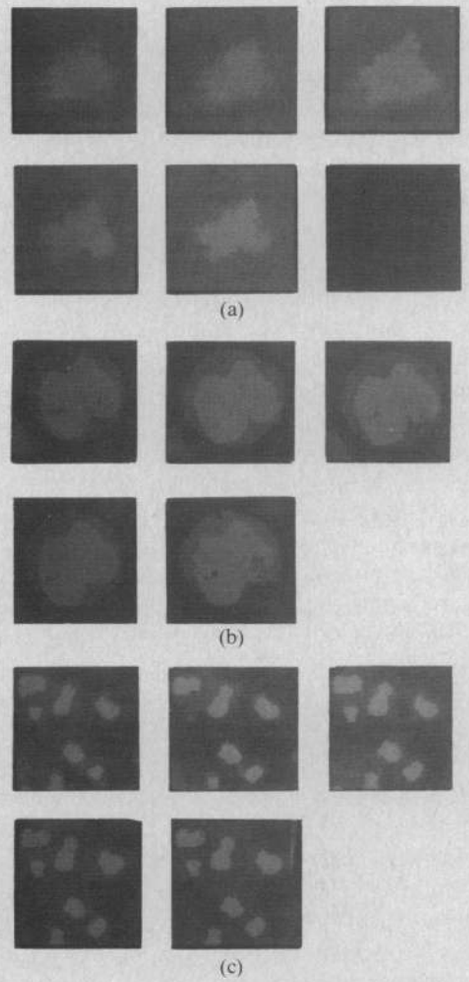


Fig. 13. Results using two-neighbor selective averaging to smooth the link weights rather than the node values ($l = 0, 1, 2, 3, 4$).

iteration of the inverse-linear weighted linking process. The chromosome results are now somewhat more stable, but the tank and blood cell results begin to deteriorate at $l = 4$.

Other smoothing methods, e.g., those described in [5]–[7], could have been used in place of selective averaging or median filtering; for small values of l, these would probably have given better results, since they tend to smooth more strongly.

VI. Multiband Linking

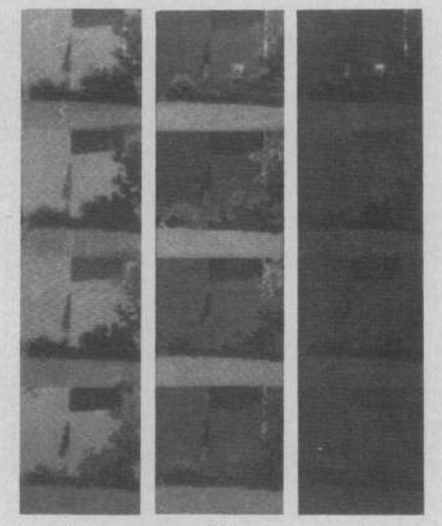
The linking process need not be based on average gray tone. We can use any property that can be computed for the pixels of the image, and extend this property to the higher levels of the pyramid by averaging. If desired, we can begin the process by dividing the image into blocks, computing a property (e.g., a textural property) for each block, and building a pyramid starting from the resulting array of property values; linking in this pyramid yields a (blocky) segmentation of the image into (at most four) textured regions [10].

The linking process need not be based on a single property; we can compute a property vector for each pixel (or block) and extend it to the higher levels of the pyramid by componentwise averaging. This section illustrates this generalization with some simple examples of pyramid linking based on properties other than gray tone and on pairs of properties. Intuitively, pairs of properties should be capable of producing more refined segmentations than single properties. When we apply the linking process to a pair (or k-tuple) of properties, we use city block distance to measure the difference between a node and its father—i.e., if the property values are g and h for the node, G and H for the father, we use $|g - G| + |h - H|$.

Our first example uses color components as properties. Fig. 14(a) shows the red, green, and blue bands of a color image of part of a house, showing sunlit and shadowed brick, bushes, and grass. Fig. 14(b) shows the results of the pyramid linking process (10 iterations) applied to each band separately, where each pixel is given the value at the root of its tree. When we examine the sets of pixels belonging to each tree, we find that the red segmentation does not distinguish bushes from shadowed brick; the green segmentation breaks the bushes up into a grass-like and a shadow-like part; and the blue segmentation is quite noisy. (Fig. 15(b) shows the four classes of each segmentation using four distinct gray tones.) Fig. 14(c) shows results when we use two colors at a time, displayed as follows:

br	***rg***	***gb***
rg	***gb***	***br***

Here, e.g., ***br*** means that the pair of colors was (blue, red), and that the displayed gray tone for each pixel is the red component of the value at the root of the pixel's tree. Fig. 15(c) shows the four classes for the segmentations using the pairs of colors *gb*, *br*, and *rg*, where each class is displayed using a distinctive gray tone. We see that the (blue, red) segmentation does discriminate (most of) the bushes from the shadow. Quantitatively, Table II shows that the roughness measures are usually lower when we use two colors (especially when one of them is blue), while the discrepancies remain about the same.



(a) (b) (c)

Fig. 14. (a) House image: red, green, and blue components. (b) Results of pyramid linking applied to each band separately; each pixel displayed with the value at the root of its tree. (c) Results of using pairs of bands: each pixel displayed with one component of the value at the root of its tree (see text).

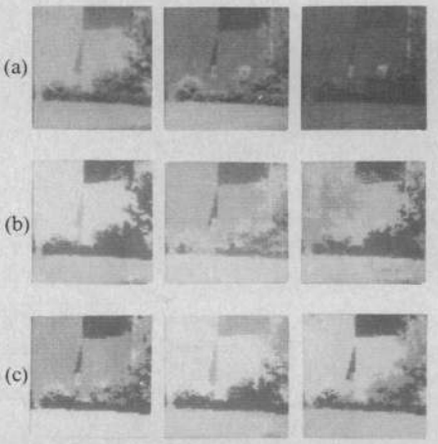


Fig. 15. (a) House image: red, green and blue components (same as Fig. 14(a)). (b) Results of pyramid linking applied to each band separately: each class displayed with a distinctive gray tone. (c) Results using pairs of bands: green/blue, blue/red, red/green.

Our second example involves a local property that measures the variability of the gray tones in the neighborhood of each pixel. Fig. 16(a) shows such a local "busyness" measure computed for each pixel of the red, green, and blue images. For the 3-by-3 neighborhood

a	b	c
d	e	f
g	h	i

this measure is defined by

$$\min(|a-b| + |b-c| + |d-e| + |e-f| + |g-h| + |h-i|, |a-d| + |d-g| + |b-e| + |e-h| + |c-f| + |f-i|).$$

TABLE II
SINGLE-BAND VERSUS MULTIBAND PYRAMID LINKING

Case	Output Roughness	Discrepancy
r	79322	466641
g	91289	349669
b	96978	261291
br	78309	472484
rg	83924	472220
rg	70945	350320
gb	74485	351298
gb	58858	260814
br	61326	261378

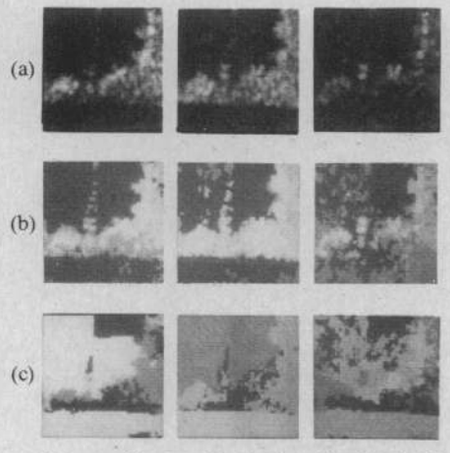


Fig. 16. (a) "Busyness" values in the three bands. (b) Results of pyramid linking applied to these values. (c) Results using (intensity, busyness) feature pair in each band.

Here the min is used to decrease the response to step edges, while retaining a high response in isotropic "busy" regions, so that the measure represents "busyness" rather than "edgeness." Fig. 16(b) shows segmentation results (each class displayed with a distinctive gray tone) based on busyness alone; the green result, in particular, gives fair discrimination between the bushes and the other regions, but the other regions themselves—as should be expected—are not distinguished. When we use two features, gray tone (in the given band) and busyness, the segmentations obtained (Fig. 16(c)) are not as useful, though that for the green image does yield the brick (both sunlit and shadowed) as a single class.

VII. CONCLUDING REMARKS

In Section II of this paper we defined a "pyramid linking" process for image smoothing and segmentation, and discussed its relationship with several previously described smoothing processes. Pyramid linking is basically a heuristically motivated process; we gave plausibility arguments indicating how it should perform, and we pointed out that it belongs to a family of ISODATA-like algorithms and is guaranteed to converge, but we have not given a theoretical model for the class of situations in which it performs "correctly" (such a model would involve relationships among the sizes and average gray tones of approximately constant-valued regions in the image, and probably has no simple formulation). But in spite of its mathematical ill-definedness, the process seems to be of some practical value.

In Section III we investigated a number of variations on the basic pyramid linking process. The results suggest the following tentative conclusions.

a) It appears to be preferable to use schemes in which some weight is given to the relative positions of nodes, both in initializing their values and in choosing links, especially in cases involving regions that consist of many small connected components. Apparently, when we take relative positions into account, we have a better chance of preserving the integrity of small regions.

b) It is desirable to specify the desired number of region types (i.e., pixel classes), by allowing only that number of nodes at the top level to be "active." Otherwise, the process tends to split some of the classes artificially. On the other hand, using estimates of the average gray levels of the classes to fix the values of the nodes at the top level may degrade the results, perhaps because it introduces premature biases that are not compatible with the early stages of the linking process.

c) The process is relatively insensitive to the sequencing of the iterations and to the node ordering used for breaking ties.

We also showed that the process yields good segmentations of synthetic images (for which the "correct" segmentation is known), and we suggested that its performance can be evaluated in terms of how it reduces the "busyness" of the image while at the same time keeping the difference from the original image small.

Sections IV and V showed that the forced-choice linking process can be "softened" into a process making use of weighted links, provided that a suitable nonlinear weighting scheme is used, or that interactions among nodes within each level are allowed in order to smooth the node values or the link weights. Section VI showed that the linking process can be based on features other than gray tone, or on pairs of features, and that using pairs often yields improved results.

Pyramid linking is somewhat different from the standard pyramid-based schemes of image segmentation (e.g., Pavlidis [11]), which split an image into blocks (top-down) rather than linking blocks into trees (bottom-up). On the advantages of combining the two approaches see [12].

The computing time required by pyramid linking (on a 64 by 64 image, using a PDP-11/45 computer) is about 3 to 4 minutes. However, the method is of interest in spite of its relatively high computational cost as compared to simple segmentation techniques, since it could be implemented very efficiently on a tree-structured cellular processor; designs for such processors are currently a topic of active research.

It should be emphasized that the methods described in this paper have been tested only on a small set of images, and we cannot be certain how they will perform on images of other types. We have given no performance comparisons between pyramid linking and conventional segmentation schemes, e.g., based on pixel classification, though we have

pointed out that pyramid linking has the advantage of being less "myopic". In any case, the pyramid linking approach (and its predecessors) certainly seems to have useful properties, and deserves consideration as a possible alternative to existing methods of image smoothing and segmentation.

References

[1] J. O. Eklundh and A. Rosenfeld, "Image smoothing based on neighbor linking," *IEEE Trans. Pattern Anal. Machine Intel.*, vol. PAMI-3, pp. 679–683, 1981.

[2] P. Burt, T. H. Hong, and A. Rosenfeld, "Segmentation and estimation of image region properties through cooperative hierarchical computation," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-11, pp. 802–804, 1981.

[3] P. J. Burt, "Fast filter transforms for image processing," *Computer Graphics Image Processing* vol. 16, pp. 20–51, 1981.

[4] S. Kasif and A. Rosenfeld, "Pyramid linking is a special case of ISODATA," TR-1096, Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD, September 1981; also *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-13, Jan./Feb. 1983, to be published..

[5] F. Tomita and S. Tsuji, "Extraction of multiple regions by smoothing in selected neighborhoods," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-7, pp. 107–109, 1977.

[6] M. Nagao and T. Matsuyama, "Edge preserving smoothing," *Computer Graphics Image Processing*, vol. 9, pp. 394–407, 1979.

[7] R. M. Haralick and L. Watson, "A facet model for image data," *Computer Graphics Image Processing*, vol. 15, pp. 113–129, 1981.

[8] A. Martelli and U. Montanari, "Optimal smoothing in picture processing: An application to fingerprints," in *Proc. IFIP Cong.* vol. 71, TA-2, pp. 86–90.

[9] L. S. Davis and A. Rosenfeld, "Noise cleaning by iterated local averaging," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-8, pp. 705–710, 1978.

[10] M. Pietikäinen and A. Rosenfeld, "Image segmentation by texture using pyramid node linking," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-11, pp. 822–825, 1981.

[11] T. Pavlidis, *Structural Pattern Recognition*. New York: Springer, 1977.

[12] M. Pietikäinen, A. Rosenfeld, and I. Walter, "Split-and-link algorithms for image segmentation," *Pattern Recognition*, vol. 15, 1982, in press.

Automatic Identification of Noise Pollution Sources

PANTELIS MOUKAS, JOHN SIMSON, AND LEONARD NORTON-WAYNE, MEMBER, IEEE

Abstract—**Instrumentation currently available for the automatic monitoring of noise nuisance has the shortcoming that although the intensity, duration, and time of occurrence of noises may be recorded, their source often cannot be identified. Research directed towards providing improved instrumentation which can identify sound sources is described. Our results suggest that application of statistical pattern recognition to recorded sounds can differentiate sources which are structurally dissimilar (e.g. trains, fixed-wing aircraft, helicopters) with an accuracy of better than 95 percent. The work is continuing to discriminate sounds which are structurally similar (e.g. different types of aircraft), and to produce hardware capable of field application.**

I. Introduction

NOISE POLLUTION is a particularly annoying consequence of life in a technologically advanced society. Surveys show that road traffic noise causes most annoyance but aircraft noises are also particularly objectionable. However, trains, noisy neighbors, and even the hum of electrical transformers can cause an intolerable nuisance. The nuisance caused by noise pollution is subjective, in that the nature of the sound and the conditions under which it is heard rather than merely its amplitude and frequency content determine the extent of disturbance.

One important step towards limiting this pollution is to analyze the nature of the nuisance by monitoring, i.e., by obtaining records of the location, time, severity, duration, and cause of incidents, preferably in a form reliable enough to be acceptable in a court of law. Several instruments are already commercially available which can record times at which the ambient noise level exceeds a preset threshold, of periods up to one week. These cannot however identify the source of the noise. Consequently, a joint project involving the Scientific Branch of the Greater London Council and the Department of Systems Science at the City University is being undertaken, directed towards providing more intelligent instrumentation to remedy this shortcoming, and thereby render existing instrumentation more versatile. The first stage of this investigation has involved analysis of recorded sounds, using a general purpose digital computer, to investigate methods of signal processing capable of identifying the sounds. The first application of this work will be to design prototype hardware for distinguishing helicopters from all other loud sounds. This will then be

Manuscript received April 14, 1981; revised March 9, 1982.

P. Moukas is with the Department of Systems Science of the City University, London, England.

J. Simson is with the Scientific Branch of the Greater London Council.

L. Norton-Wayne is with the School of Electronic and Electrical Engineering, Leicester Polytechnic, Leicester, England.

0018-9472/82/0900-0622$00.75