REGION EXTRACTION AND DESCRIPTION THROUGH PLANNING[1]

Paul A. Nagin
Allen R. Hanson
Edward M. Riseman

COINS Technical Report 77-8

May 1977

ABSTRACT

This paper examines several image segmentation algorithms which have been explored in the development of the VISIONS system. Each of these algorithms can be viewed as a variation on a basic theme: the clustering of activity in feature space via histogram analysis, mapping these clusters back onto the image, and then isolating regions by analysis of the spatial relationships of the cluster labels. It is shown that the interaction between these two representations of data (global feature information and spatial information) provides a view that is lacking in either.

The scene segmentation algorithms contain the following stages:

(1) PLAN: reduce the amount of detail in the scene to a bare minimum by performing a fast simple segmentation into primary areas using spatial and/or quantization compression.

(2) REFINE: resegment the scene with careful attention directed to the textural complexities of each region.

The primitive transformations which are used include histogram clustering, region growing, data reduction by narrowing the quantization range, and/or data reduction by spatially collapsing the data while extracting features. These algorithms have been implemented using a parallel, hierarchical computational structure. Comparisons of performance on several images are given.

[1]This research was supported by the Office of Naval Research under Grant N00014-75-C-0459.

I. INTRODUCTION

Outdoor scenes contain many different types of visually complex objects. These objects appear in an image as a set of regions and our goal is to decompose digitized images by exploiting various features of these regions. In the design of VISIONS [1,2] an assumption was made that semantic information would not be available at least in the initial stages of processing. Although the desirability and effectiveness of such an assumption is currently under debate in the literature [3,4,5,6], we believe that a segmentation system initially should be data-directed and then later receive feedback from interpretation processes.

The isolation of objects is not the goal of this work. A region may contain or be contained within an object. The relevance of such relationships is a function of the goal or "focus" of the visual analysis. There is no "correct" segmentation in general -- it is dependent on the goals of the system. For instance, one could ask which of the following sets of objects is the appropriate level of description for Figure 1:

(1) Outdoor scene

(2) House + trees + sky + grass

(3) Windows + doors + roof + leaves + clouds + blades of grass + ...

Clearly, each of the above descriptions is appropriate, given particular goals. Under our assumptions it is certainly not the responsibility of a general (non-semantically directed) low-level system to choose between them.

We propose, then, to provide a multi-level description of the scene based on region properties. Regions can be extracted using the invariance of gross features of "macro-texture" and then be refined into subregions on the basis of features which provide a more detailed representation of the data.

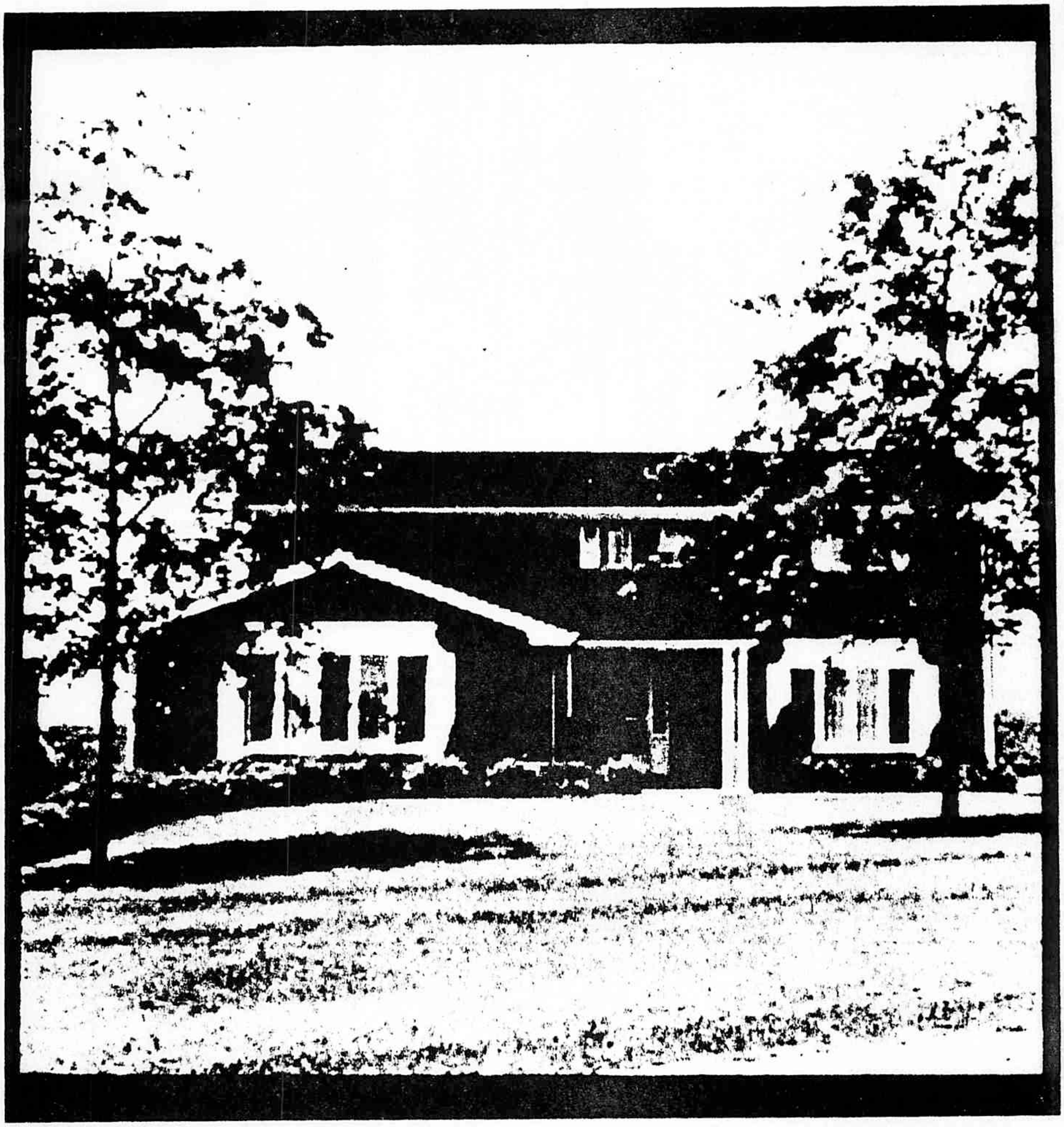


Figure 1: Example Scene

A region at any stage can be related to the parent region derived from a coarser degree of segmentation or to descendant subregions using finer degrees of segmentation. These results might be stored in a tree structure [7] in which relationships between the nodes represent descriptive properties of the structure of the visual elements. For example, a textured region might have descendant arcs to large subregions representing macro-texture elements, such as large dark shadow areas or light leafy branch clumps, which often appear in images of trees. This approach leaves responsibility to a high-level system to sort out the correct level of description by fitting interpretations to different levels, or extracting useful levels depending on size and properties of regions. Although there are many interesting possibilities, the details have not been worked out yet and will not be addressed further in this paper.

We employ (simulate) a hierarchical parallel data structure called a processing cone [8] to facilitate the segmentation process. The higher, more spatially compressed layers of the cone are used to effect a global view of regions of the scene. Values of features can be extracted by applying a programmable function in parallel to local windows and collecting their results (in parallel) into cells whose effective receptive field increases from level to level. The importance of using this operation of data reduction lies in its ability to effectively collapse textural areas -- via the extraction of some feature -- so that they tend to be more homogeneous in feature space.

There is a second kind of global view of the data which is also exploited. The spatial information can be ignored and instead we can simply focus on the statistical distribution of features of the data.

Standard representations for global feature analysis are 1- or 2-dimensional histograms of features. Using this representation, clusters of activity (i.e., clearly defined modes of the distribution) might be extracted.

Various descriptors which arise from the spatial and statistical techniques used in scene segmentation can be assigned to the regions. For instance, the texture of a region may be describable as any of the following:

(1) homogeneous; or

(2) smooth gradient; or

(3) speckled (grainy, blobby), etc.

In a later section, we will show that it is possible to extract these desscriptors using a spatial adjacency (co-occurrence) matrix [11,12]. For a given feature of a region, the adjacency matrix indicates the number of times that feature value i has occurred spatially adjacent to feature value j. A large diagonal entry indicates a homogeneous region of activity in the image. Conversely, an off-diagonal entry indicates frequent adjacency of two values which might be a meaningful textural pattern.

Our contribution to the growing body of segmentation algorithms lies in the interactions which we develop between the two types of global views described above. Neither view alone is adequate to deal with natural scenes. A simple region grower, which merges points based on local spatial information cannot, in general, deal with textural variation. Likewise, histograms of multi-object scenes suffer from ambiguities due to overlapping distributions which obscure the feature activity of individual regions. A system is needed to exploit the strengths of both representations.

The algorithms which we have designed are executed in two stages:

(1) PLAN - coarsely segment; and

(2) REFINE - carefully segment.

Crudely stated, the purpose of the planning stage [9,10] is to reduce the amount of detail in the scene to a bare minimum. This will have the effect of decomposing the scene into grossly similar regions (primary areas), thereby easing the problem of histogram overlap mentioned above. Once we have found these large regions, the segmentation can be refined, using a subset of a larger pool of features and more sophisticated processes for clustering and region growing. In the next stage of development of the VISIONS system, this pool of features could provide the basis for careful object verification.

The next section of the paper examines a series of problems inherent in the histogram clustering analysis. Following this is a detailed presentation of the segmentation strategies proposed here. Finally, we present several results of applying the algorithm to real data (Figures 1 and 2).

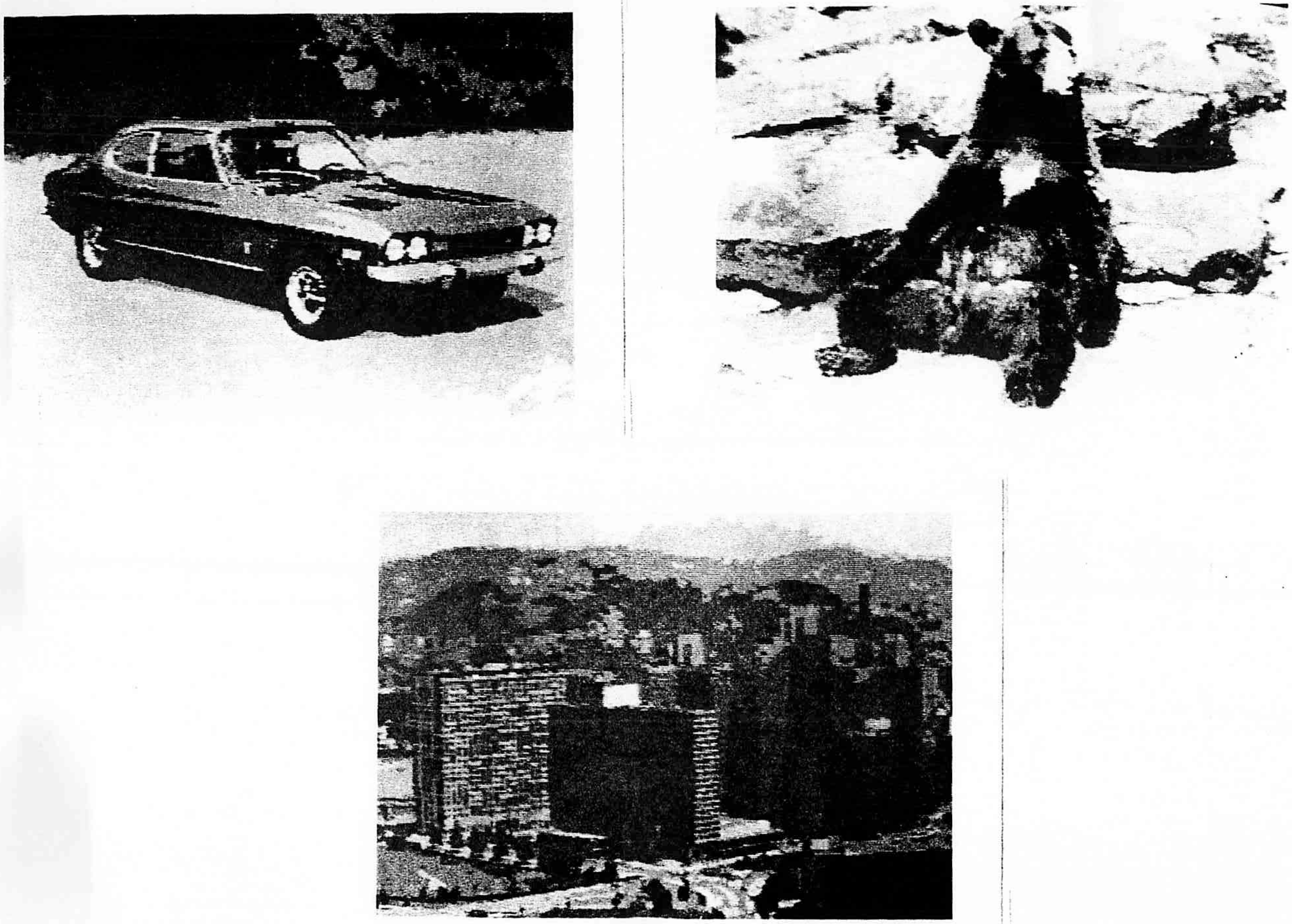


Figure 2: Other outdoor scenes which are analyzed here

II. REGION FORMATION VIA CLUSTER ANALYSIS OF HISTOGRAMS

This section outlines some of the problems inherent in histogram clustering. For expository reasons, these problems are presented using extremely idealized examples; their real world counterparts are much less well-defined.

Histograms provide a global statistical view of the data which is independent of the (global) spatial relationships in that same data. While the goal of processing in the spatial domain is to isolate regions, the goal here is to label clusters (or modes) of feature activity. Ideally, if each cluster of some histogram corresponded to a particular region in space, then it would be simple to assign to each point in the image the label of the cluster to which it belongs. Subsequent application of a simple region growing algorithm can then be used to merge all contiguous points with the same cluster label into a distinct symbolic region [13]. The two stages of transformations -- assigning cluster labels and region growing -- may be expressed functionally as follows:

CLUSTER and REGION GROW: $P_i \longrightarrow C_A \longrightarrow R_\alpha$

where:

P = {pixels in an image and their corresponding feature value}

C = {cluster labels corresponding to clusters which have been found in the histogram of an image}

R = {region labels found in the following manner: Pixels P_i and P_j will be given the same region label R_α, if:

(1) they have been assigned the same cluster label, and

(2) they are spatially connected}

At the most general level it is clear that such a histogram would, of necessity, be multi-dimensional. That is, there is no single feature of the input data which can be expected to discriminate all regions of a typical image. We have rejected the use of N-dimensional histograms (where $N > 2$, or possibly 3) because of the difficulty humans encounter in trying to understand the feature distribution in N-space; the algorithms cannot be easily evaluated because distributions in N-space cannot be examined in a straightforward manner. One-, two-, and three-dimensional histograms can be displayed graphically, require relatively modest storage, and can be clustered reasonably fast. On the other hand, clustering in N-space, in addition to being costly, leads to fragmentation of the scene since differences in additional features will cause regions to split into sub-regions.

The following examples in this section will serve to illustrate the clustering process and its problems. For convenience, all the examples will be presented in terms of histograms of a single feature -- let us say intensity or brightness -- without any loss of generality in the conclusions that are drawn. The cluster labelling algorithms proceed as follows:

I - form the histogram;

II(a) - smooth the histogram;

II(b) - set Θ equal to the deepest valley which occurs between the highest peaks;

II(c) - label all points in the histogram which are to the left of Θ as C_A and those to the right as C_B (denoting clusters A and B);

III(a) - assign to each point in the image the label of the corresponding cluster to which it belongs;

III(b) - grow regions across adjacent points which bear the same cluster label; and

IV - clean up by merging small regions into a larger surrounding region.

Note that in the description of this algorithm it is implied that there are only two clusters and one division point. It is possible, of course, to generalize to as many clusters as are reasonably defined in the data; we will return to this point later.

The key effect of the cluster labelling process is to allow all points within a cluster to be considered equally similar. Consider the following case. Two adjacent points in the image (x_1,y_1) and (x_2,y_2) have feature values $f(x_1,y_1)$ and $f(x_2,y_2)$ which fall in the same cluster C_A. Depending upon whether they are near each other in the cluster or on opposite sides of the cluster, their distance

$$d = \| f(x_1,y_1) - f(x_2,y_2) \|$$

will be relatively large or small. In order for a spatial region growing process to put them in the same region, it must use threshold θ where $\theta > d$. But then all spatially adjacent points whose difference is less than θ will be grouped. This can cause severe problems when two distinct clusters are less than d apart. This condition is not unusual since clusters may be large while their boundaries are not far apart.

The cluster labelling process deals with this problem quite nicely. All points of a cluster are provided a label which in effect says that they are equivalent or zero distance apart in labelled feature space. Now points on the opposite sides of clusters can be considered "closer" than points in the next cluster, no matter how close that next cluster is.

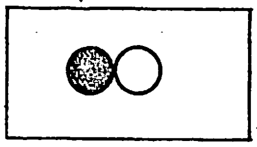
II.1 Segmentation by Recursive Cluster Analysis

A very simple segmentation problem is illustrated in Figure 3a. An intensity image is presented which contains a dark and a bright region. The histogram of the image (Stage I) is clearly bi-modal and can be seen as a composite of two normal distributions. The goal now is to isolate the two clusters (Stage II) by determining the minimum Θ which lies between the two modes. Having labelled the two clusters, we are now in a position to re-examine the image. The clustering process has effectively compressed the feature scale to two values, and by mapping these back to the corresponding image points (Stage III) we obtain a much simpler picture; one in which all local textural variations have been ignored. Finally, since it is possible that one cluster label may generate many spatially distinct regions (although this does not occur in the example), it is necessary to region grow. In this context, the region grower would be defined so as to link all image points which are spatially adjacent and bear the same cluster label.

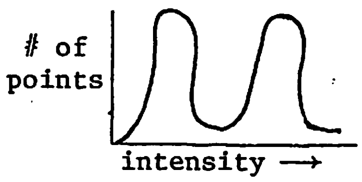
Figure 3b shows an example of the technique shown in Figure 3a using real data. The regions under consideration are the sky and grass areas from Figure 1. The example is somewhat artificial in that these two regions do not comprise the entire image; we have contrived to show an easy segmentation case, i.e., one in which the distributions of the objects are quite separable. In the examples that follow it will be evident that the situation presented here is an exception. Most histograms of complex scenes are quite messy since the distributions of the individual objects tend to overlap one another leading to difficulties in cluster separation.

Simple Segmentation Algorithm

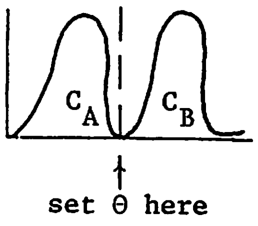
Image



Stage I: Histogram



Stage II: Cluster and Label



Stage III: Map Labels Back to the Image and Region Grow the Labels

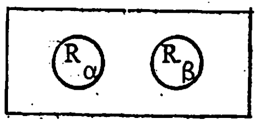


Figure 3a

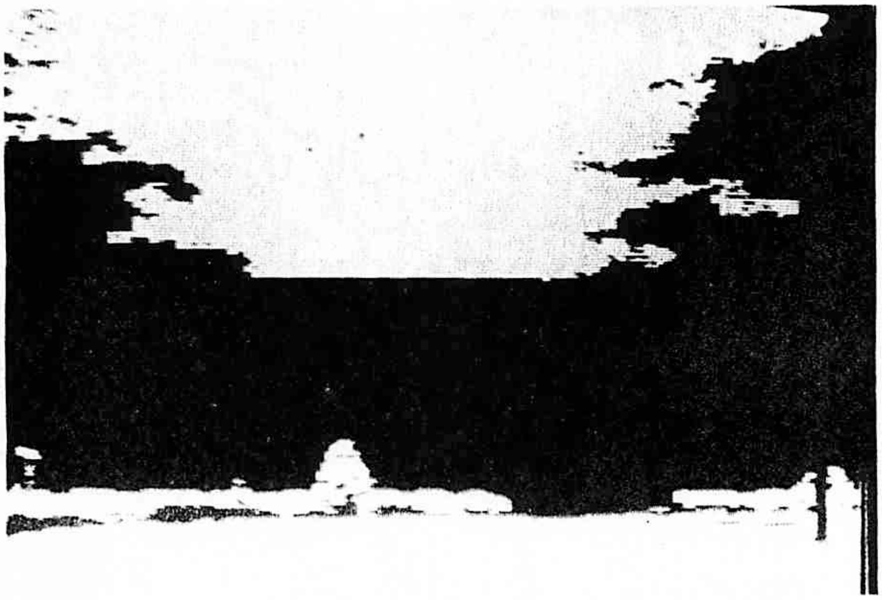
Simple Segmentation Algorithm: Example

```
11     4    *
12    37    *
13    72    **
14   339    ****
15  4751    *******************************************
16  5897    ******************************************************
17  4792    ********************************************
18  1227    ************
19   249    ***
20   257    ***
21   243    ***
22   190    ***          ---set theta at 22
23   213    ***
24   204    ***
25   200    ***
26   244    ***
27   429    ******
28   624    *******
29  1262    *************
30  2002    *******************
31  2364    ************************
32  3669    **********************************
33  1862    ******************
34  1214    ************
35   884    *********
36   275    ***
37   122    **
38    39    **
39    10    *
40     5    *
```

C_A (rows 11–21)

C_B (rows 22–38)

(a) Histogram of partial image



(b) Result of mapping cluster labels back to image.

Figure 3b

Figure 4 indicates the effect of overlapping distributions in a particular feature. Two of the regions would be separable as in Figure 3a, but the introduction of the third region obscures the other disbributions, leaving only one discernable cluster. The mapping of the cluster label back onto the image allows the spatial relationships of feature activity to be integrated into the process. In this case, growing regions in the image space provides information which allows us to discriminate the two disconnected regions R_α from R_β. The proper segmentation is accomplished by repeating the clustering process on R_α via the sequence of operations shown in Figure 3a. This recursive partitioning of a region was employed by Ohlander [13] and Price [10] quite effectively. The recursive segmentation stops when a region is found to be unimodal in all features under consideration. In practice, of course, we might not know a priori which, if any, regions would need to be reprocessed; all would have to be examined. This is very disconcerting in that it can lead to a proliferation of histogramming opeations.

Figure 5 indicates that even a recursive analysis might not succeed. In general, the particular location of a region is quite arbitrary; this example differs only slightly from the previous cases: a small change in the spatial arrangements of regions. The algorithm fails because the mapping of the cluster label to the image does not lead to spatial splitting of distinct subregions. This example is particularly annoying (and realistic) in that Region 3 might be joined to Region 2 at only a few points. Heuristics can be defined to deal with each such problem, but, in general, one wishes to avoid a proliferation of heuristics for a variety of special problems.

Recursive Segmentation

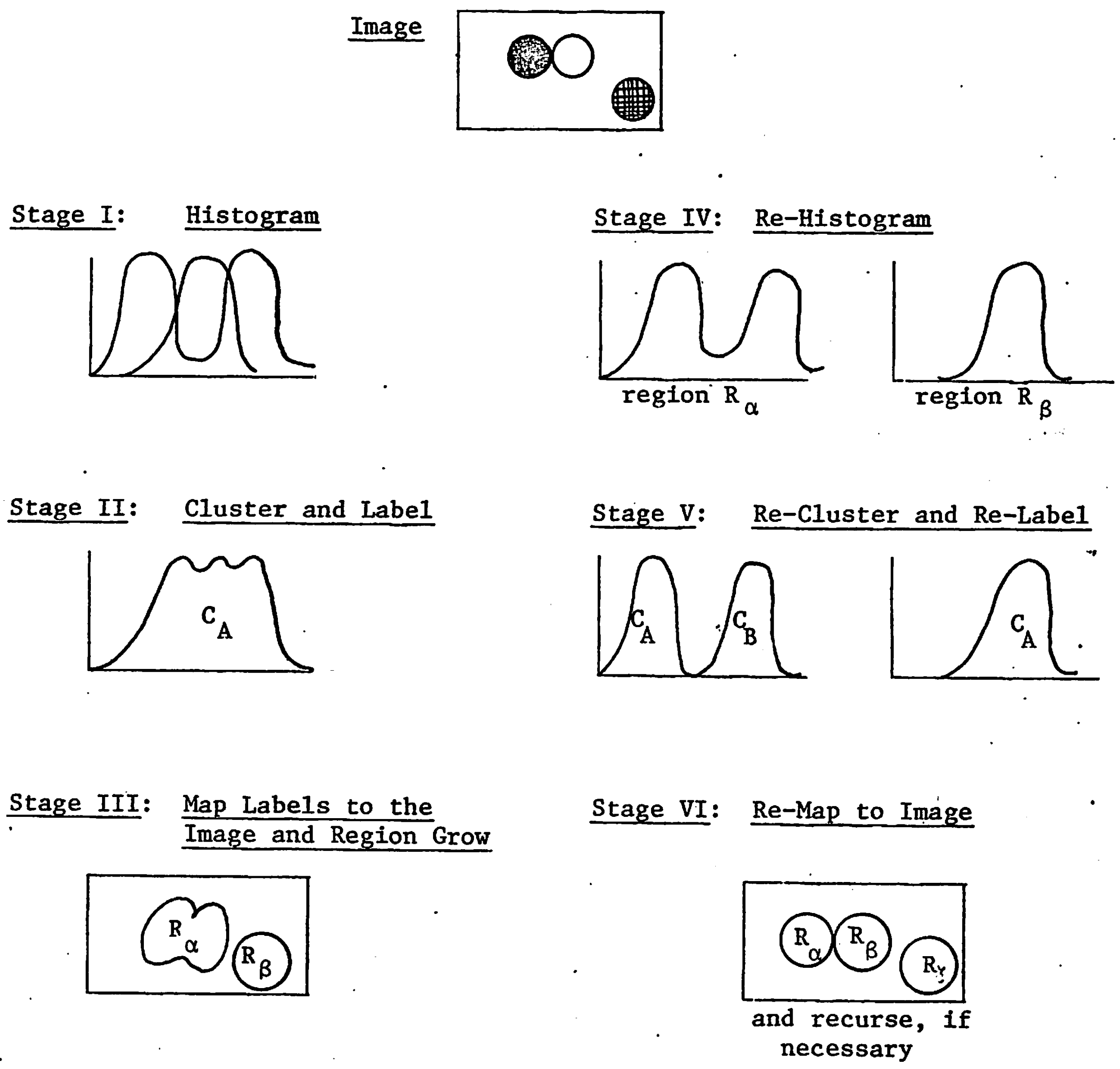
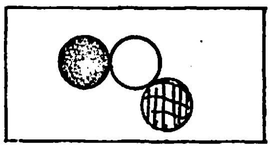


Image
Stage I: Histogram
Stage II: Cluster and Label
C_A
Stage III: Map Labels to the Image and Region Grow
R_α
R_β
Stage IV: Re-Histogram
region R_α
region R_β
Stage V: Re-Cluster and Re-Label
C_A
C_B
C_A
Stage VI: Re-Map to Image
R_α
R_β
R_γ
and recurse, if necessary

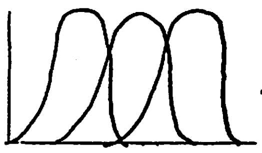
Figure 4

Recursive Segmentation -- Failure

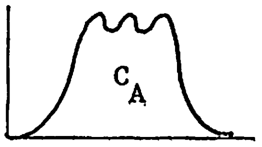
Image



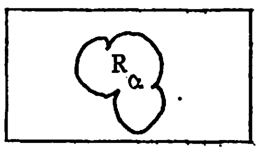
Stage I: Histogram



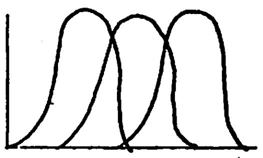
Stage II: Cluster and Label



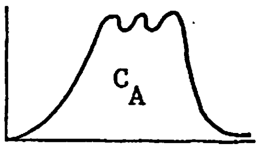
Stage III: Map Labels to the Image



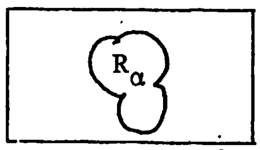
Stage IV: Re-Histogram



Stage V: Re-Cluster and Label



Stage VI: Re-Map to Image



no segmentation has occurred

Figure 5

For example, in this case, a region could be snipped in two at narrow points; one way to do this is by shrinking the region in from its perimeter for a few layers of pixels and detecting those isolated subregions which may have been formed.

II.2 Reducing the Errors by Conservative Cluster Formation

Let us now look more carefully at the clustering process. So far, the illustrations have shown distributions which were either highly overlapping or quite distinct. More difficult situations are created as the distributions increasingly overlap at their tails. The area of significant overlap involves points whose cluster identity is most uncertain.

The clustering algorithm discussed previously makes use of "liberal" clustering; i.e., a liberal policy of defining clusters. This leads to expectations that the cluster will include points which cannot be reliably associated with that mode. As a result, upon mapping the cluster label and growing regions, it is expected that regions or parts of regions will be erroneously labelled.

The errors which have been introduced by the liberal clustering process can be minimized by analyzing their effect in the spatial domain. A simple error reducing heuristic would involve suppressing the outer layer of all regions, i.e., those which are "all boundary" will be completely suppressed. This is desirable in that it will have a tendency to clean up noisy areas since regions of size no less than 9 points are needed in order to have a non-empty core. Second, weakly adjacent regions that are spatially linked by a very thin trail of pixels will be separated. Finally, it is observed that there are instances in which the points lying along the spatial periphery of a region also lie at the tail of the distribution. In these cases the peripheral points are those which have a low probability of being reliably affiliated with the region and ought to be suppressed until a further analysis can decide where they should belong.

The analysis just described does not fully explain the problem -- it focusses only upon the spatial relationship between the pixel and the region. However, the relationship of the histogram point to the cluster (the feature values of the pixel and the region) should also be used. A more effective analysis of the problem of liberal clustering might involve assigning a confidence value that a point in the histogram belongs to a particular cluster. A straightforward confidence measure for a point is some function of the distance of the point from the cluster mean (e.g., the number of standard deviations). The suppression pass could then be qualified so that only low confidence peripheral points would be suppressed.

As an alternative to the liberal formation of clusters, it could be argued that it is better to leave ambiguous portions of the distribution unlabelled rather than have to undo their effects at a later time; errors can be reduced by demanding confidence in decisions until they are more certain, a standard technique in pattern classification. To this end, we propose a "conservative clustering" analysis as depicted in Figures 6a and 6b. The original clustering algorithm at Stage II would be amended as follows:

II(a) - smooth the histogram;

II(b) - set Θ equal to the deepest valley which occurs between the highest peaks;

Conservative Cluster Formation

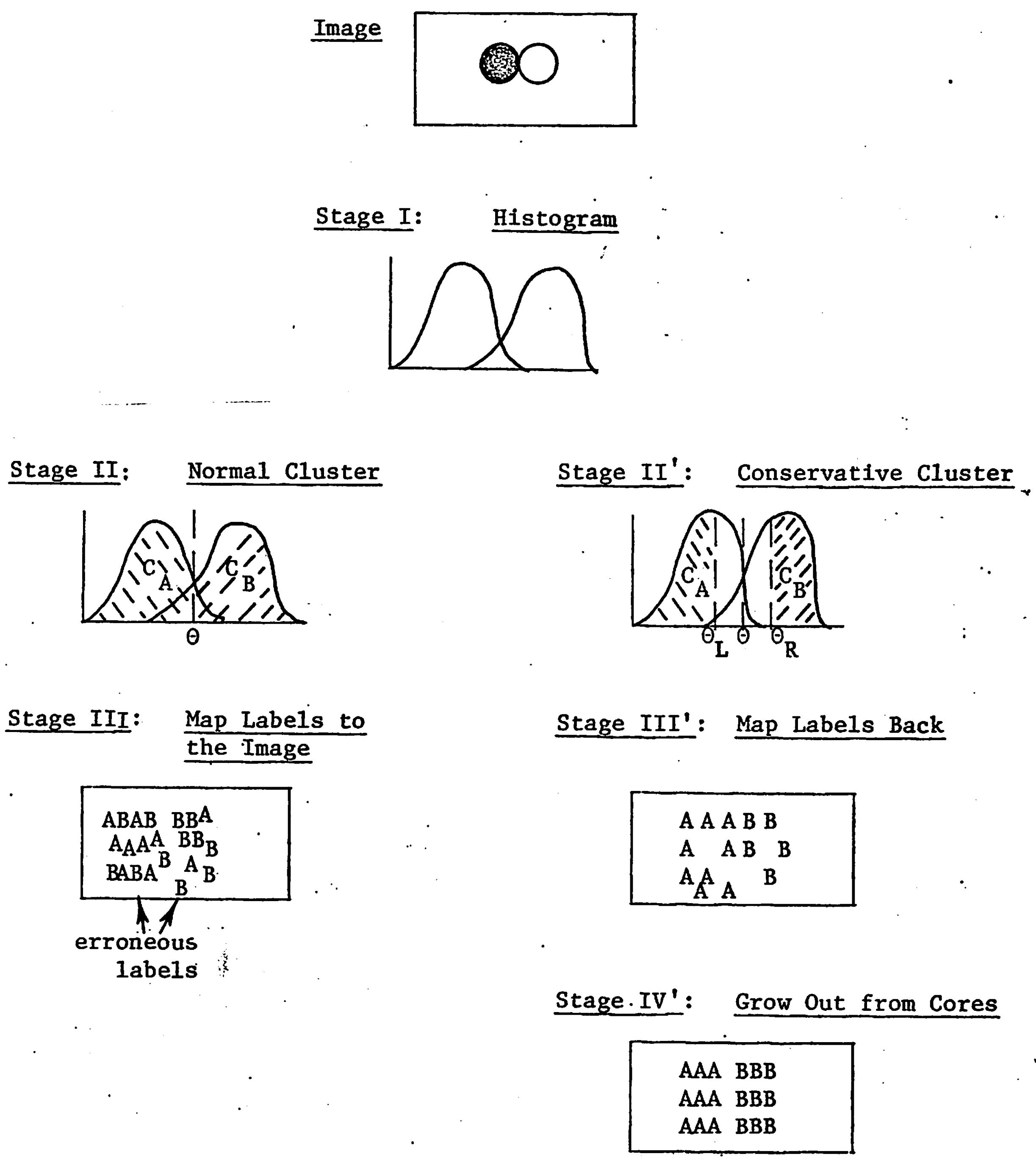
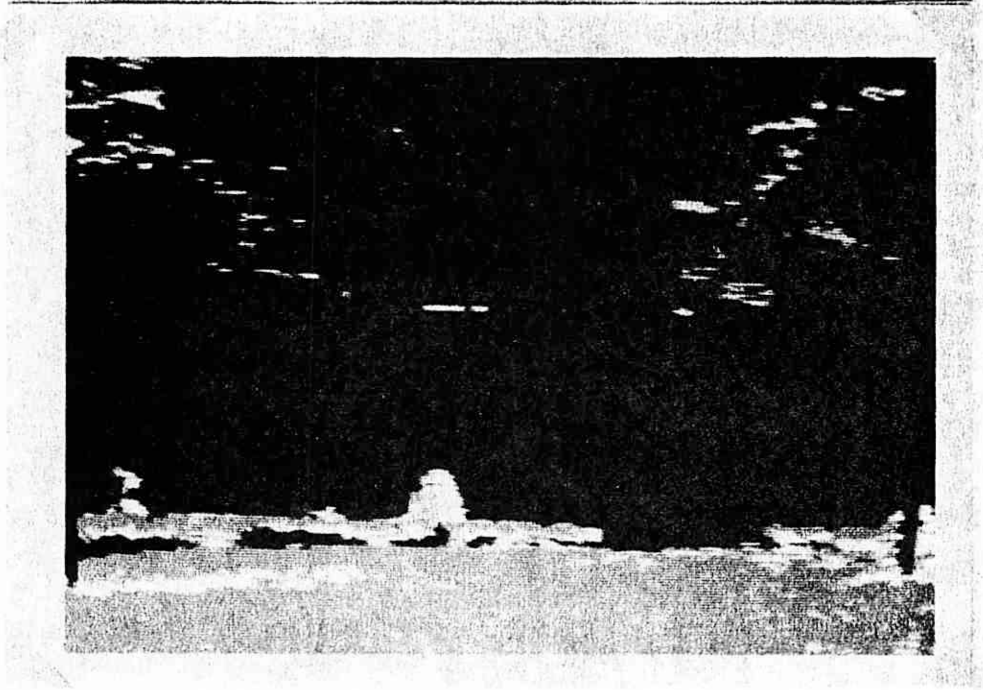


Figure 6a

```
11     4   *
12    37   *
13    72   **
14   399   ****   ___   ___   ___   ___   ___   ___   ___   ___C_A___
15  4731   ********************************************
16  5697   *****************************************************
17  4792   ********************************************
         ___   ___   ___   ___   ___   ___   ___   ___
18  1227   ************
19   249   ***
20   257   ***
21   243   ***
22   190   ***
23   213   ***
24   204   ***
25   200   ***
26   244   ***
27   429   *****
28   624   ******
         ___   ___   ___   ___   ___   ___   ___   ___
29  1262   ************
30  2082   ********************
31  2364   ************************
32  3669   *********************************
                                                          C_B
33  1662   ******************
34  1214   ************
35   604   ********
         ___   ___   ___   ___   ___   ___   ___   ___
36   275   ***
37   122   **
38    53   **
```

(a) Histogram of partial image with conservative thresholds indicated



(b) Result of mapping clusters back to image pixels. Bright areas represent unlabelled histogram points. Internal "holes" can be filled in heuristically. The shadow appears as a complete region and can be labelled as such.

Figure 6b

II(c) - set $\Theta_L = \Theta - K$ and
set $\Theta_R = \Theta + K$ where K is some constant or a function of the cluster widths;

II(d) - label all points to the left of Θ_L as C_A and all points to the right of Θ_R as C_B.

When the cluster labels are mapped back to the image, we will have labelled only those points which have a high probability of being parts of regions; these we call the region cores. Note that the region core is not necessarily spatially internal to the region. It maps onto the region leaving missing points or "holes" at various places in the region. The points of the region which are not in the core have a lower probability of being in the region, but often they will be adjacent to (possibly several) high probability points of the region.

It is at this point that the global feature information can be brought together with the spatial information in the image. The low probability points could be added onto the core by region growing in the context of high probability points -- rather than making the decision in the absence of such information. The global feature analysis can be expressed in terms of the mean and standard deviation of the region core, as well as the distance of a point not in the core to the mean of the core. The spatial information is available as the neighboring points around a given point which is not in the core; the number of neighbors which are in the core, as well as their confidence, can be used to decide whether to add the given point in a region growing process. There are many liberal or conservative region growing strategies which can employ this rich interface of local and global analysis. In Figure 6a, we have depicted Stage 5 as a region growing process of a few iterations to retrieve those points which have a relatively high likelihood of being part of the region.

III. PLANS

One-dimensional histograms of scenes containing multiple objects cannot be expected to reveal distinct distributions for each of the individual objects. The feature distributions of objects often overlap, and the histogram analysis leads to erroneous region labelling. In order to deal with this situation without semantics, the low-level system will perform a segmentation in two stages.

The first stage of segmentation is called plan generation. The goal of the planning stage is to partition coarsely with respect to capturing detail in the picture. It is expected that the planning algorithms will make mistakes such as the overmerging of regions and suppression of fine detail. The reason for using the planning stage is that its output, although coarse, will tremendously reduce the complexity of the scene; it will generate a set of subscenes where some of the major regions may contain as few as one or two "objects".

Once the plan has been generated, each region can be carefully segmented into subregions. Hopefully it is at this stage--called refinement--that the system will generate a set of regions which are closer to a one-to-one correspondence with the parts of objects in the scene. Of course, the problem of focus of attention will persist--does the scene contain, say, 10 leafy regions or one tree region? The goal here is to generate regions which are reliable; that is, given a particular focus, the refined segmentation should provide, as often as possible, high confidence regions in correspondence with objects or parts of objects.

The refinement algorithms will use two-dimensional histograms to break up overmerged regions. The presence of texture can cause clusters in a histogram of any dimensionality to become wide, to smear, or to become multi-modal. When dealing with many differently textured areas, two-dimensional clustering becomes difficult, and even under human direction the cluster labelling and mapping process becomes quite error-prone. The strategy of reducing cluster interference by problem decomposition becomes quite useful. By using the sub-scenes provided by the regions of the plan regions, two-dimensional histograms become manageable tools for exploiting pairwise feature dependencies.

It is a further goal of the refinement algorithms to associate a texture descriptor with each region. This descriptor will be used as an aid to object recognition during the semantic interpretation phase of analysis. We desire measures of the structural and/or statistical characteristics of the micro-regions of a region. Adjacency matrices [14] have been mentioned as a possibility; if there are N types of subregions, one can form an N × N adjacency matrix where the entry a_{ij} is a count of the number of times subregion type i is adjacent to subregion type j. For example, measurements across the tree region could result in the formation of regions with different hue values and an adjacency matrix might show a large off-diagonal entry, indicating blue (sky) adjacent to green (leaves). Simple functions of the adjacency matrix, such as angular second moment difference (ASMD), might indicate relative heterogeneity of the underlying spatial distribution. This type of feature has been used effectively by Haralick [11] in classification of texture in ERTS images. Other measures of texture such as variance and edge per unit area can also be computed over features of the plan regions [3,15].

To summarize, the plan and refinement stages provide a hierarchical decomposition of the scene into regions which will correspond to objects at some level of description or focus of attention. The final output of the low-level system will be a set of segmentations, each one of which will provide a more microscopic view of the scene. Each further partition will carry various feature descriptors of the regions involved.

III.1 Histogram-Guided Quantization Compression or Gross Clustering

A standard data compression technique in image processing is that of gray-sacle transformation by histogram clustering. Our use of this technique (which we shall refer to as compression) involves reducing the number of bits per pixel from, say, 6 to 2 producing an extremely coarse resolution in the values of the feature involved. The compression can be performed without global analysis by linear scaling, but this results in many erroneous contours being formed in the image. In Figure 7, linear scaling to 8 buckets causes undesirable breaks in the clusters of the distribution. A region grower applied to the transformed image often would produce a set of regions which have been artificially fragmented.

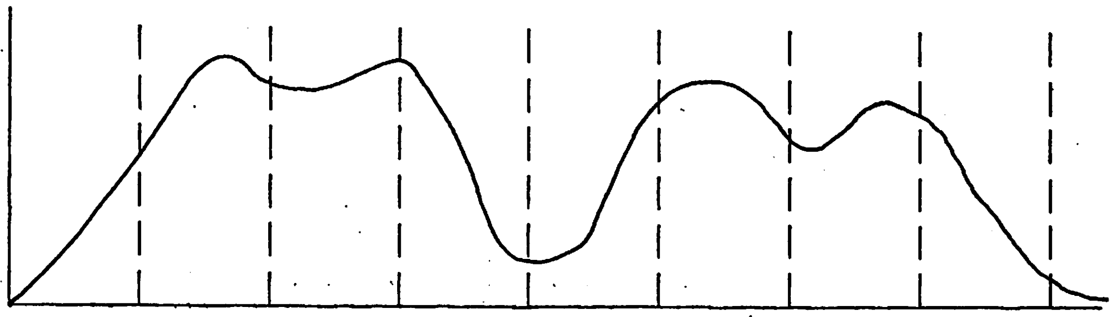


Figure 7 - Linear compression to 8 buckets.

As an alternative to blind scaling, a more sophisticated quantization compression could be applied to the statistical distribution of the feature under consideration. Instead of defining buckets at predefined linear or non-linear intervals, the histogram could be used to guide the algorithm so that the designation of the intervals would be defined by the gross cluster points (minima and maxima) of the distribution. This is, of course, a generalization of the clustering algorithm presented in the previous section.

A simple measure of the reliability of a bucket designation (refer to Figure 8) is how well the peaks (A_1 - A_4) within buckets are distinguished from the value of the distribution at the breakpoints (B_0 - B_4) of the buckets. For the breakpoint B_2 this could be the $\mathrm{MIN}[A_2/B_2,A_3/B_2]$. The effectiveness of the breakpoint set (or cluster boundaries) will be some measure of the sequence of ratios, such as average or maximum.

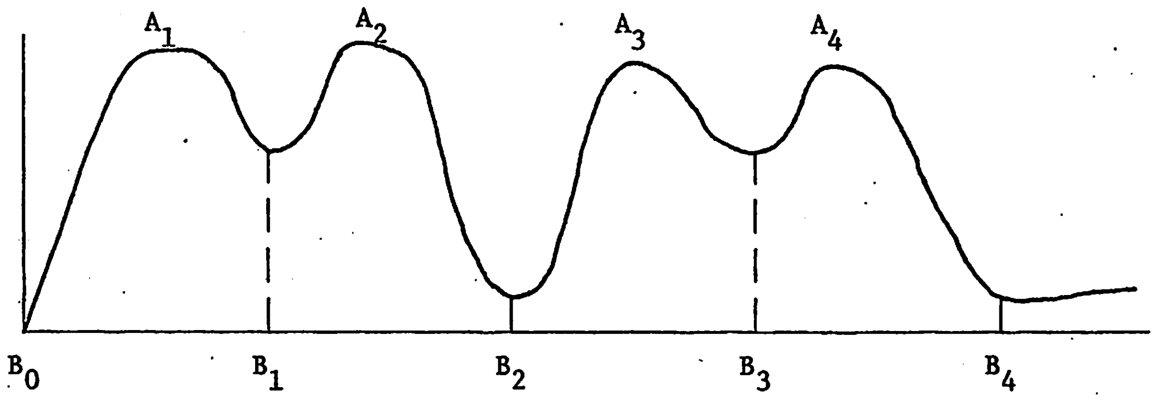


Figure 8 - Example of compression quantization scale that has been linearly defined.

In this example, B_1 and B_3 would not be considered reliable because the ratios A_1/B_1, A_2/B_1, and A_3/B_3, A_4/B_3 are small. By the same criterion, B'_0 in Figure 9 would be considered a reliable breakpoint since A'_1/B'_0 and A'_2/B'_0 are large.

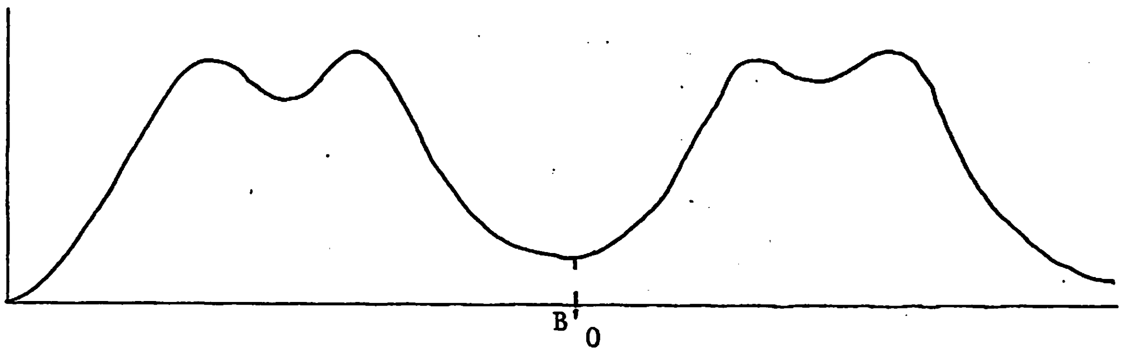


Figure 9 - Setting of intervals that more reliably reflect the grossest level of cluster structure.

Rather than deal with N-dimensional ($N > 1$) histogram analysis in the first development of plans (where mistakes are expected), the features initially will be examined independently. The kind of compression indicated above will be performed separately on each of, say, 2 (or 3) features followed by a simple intersection of the 2 (or 3) resulting images. One way to think of the resulting image is as an image of symbolic labels. Let us assume for the moment that only one breakpoint will be used to compress each feature into two buckets. This would form two (or three) binary images. The intersection of these is equivalent to encoding each point with one of a very few labels (4 or 8) of the compressed feature scales, where each label represents a hypervolume in N-space.[1] A region grower could now be applied which would link spatially adjacent points which carried the same label. These regions then would form the plan.

Examples of Quantization Compression

The above procedure is demonstrated in the following set of figures (10 - 13). The three features used are Y, I, and Q (see appendix) which are the standard color representations of the television industry. These features are a linear transformation of the red, green, and blue input data into a second color space. Designed as color opponents they measure, respectively, intensity, cyan-orange, and green-maroon. The histogrammed features have been compressed by hand using the criterion demonstrated in Figures 7 - 9. In most cases, the setting of the threshold is obvious and we are currently developing algorithms to automate this process. Note that the "I" feature often does not produce the well-defined bimodality of the other two features; in such cases the feature is not used.

The photograph accompanying each figure demonstrates the technique mentioned above; that is, intersecting binary images which have been formed after thresholding the histograms of the features involved. The images used were selected from among the set used by Ohlander.

[1] Of course, if the data called for it, each feature could be broken into more than two buckets allowing finer resolution for the feature scale, and therefore more possible labels for each pixel. We believe that it is only useful to think of the values as labels in the coarsest resolutions. Once there are a few values per feature, the textural variations will begin to fragment the image into many different subregions. There will be so many possible labels that the set of features ought to be thought of as a vector of features in the usual sense.

```
LEVEL 0, FUNCTION 1, RANGE  46, MIN VALUE    7, MAX VALUE [illegible], IN
  7-  7     7 *
  8-  8    53 *
  9-  9   122 **
 10- 10   229 ****
 11- 11 [illegible] *******
 12- 12   683 *********
 13- 13   622 **********
 14- 14   712 **********
 15- 15   678 *********
 16- 16 [illegible] ***********
 17- 17  1017 **************
 18- 18  1066 **************
 19- 19  1035 **************
 20- 20 [illegible] *************
[illegible] **************
[illegible] ******************
[illegible] ****************
[illegible] **************
[illegible] **************
[illegible] ***************
[illegible] ****************
[illegible] **********************
[illegible] *******************
[illegible] *********************
 31- 31 [illegible] **********************
[illegible] *************************
[illegible] ******************************
[illegible] ********************************
[illegible] *****************************************
[illegible] ***********************************************
[illegible] **************************************
[illegible] *************************
[illegible] *************            ---set theta=39
[illegible] ******************
 41- 41 [illegible] *****************************
[illegible] ************************************
[illegible] ******************************
[illegible] ************************
[illegible] *************************
[illegible] ***********************************
[illegible] *********************************************
[illegible] **************************************
 49- 49   562 ********
 50- 50   266 ****
 51- 51    67 *
 52- 52    21 *
```

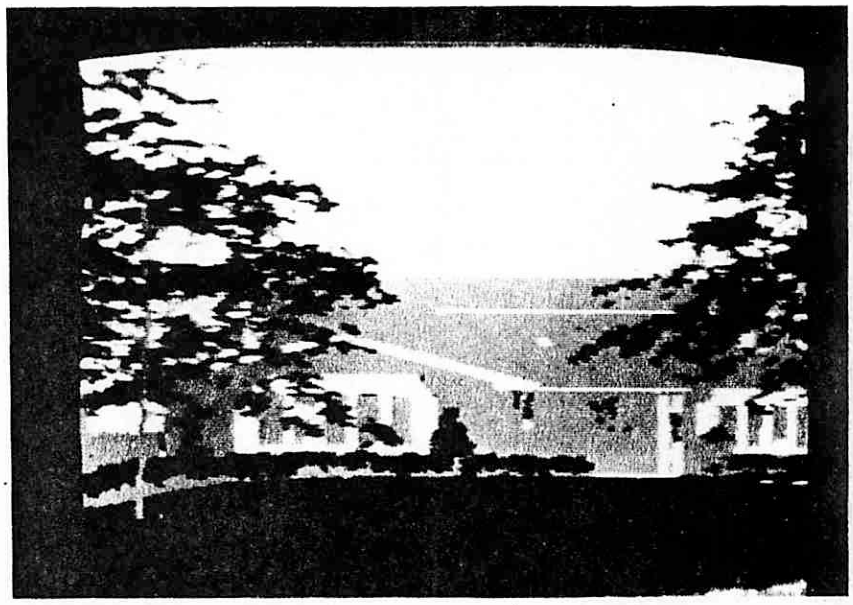
(a) "Y" Histogram

```
LEVEL 0, FUNCTION 2, RANGE  47, MIN VALUE    0, MAX VALUE  46, IN
  0-  0     9 *
  1-  1    72 *
  2-  2   296 ****
  3-  3   490 ******
  4-  4   821 **********
  5-  5  1127 **************
  6-  6  1064 *************
  7-  7   986 ************
  8-  8   498 ******
  9-  9  1165 **************
 10- 10   471 ******
 11- 11  1497 ******************
 12- 12  1058 *************
 13- 13  1619 ********************
 14- 14  1321 ****************
 15- 15  1606 ********************
 16- 16  1812 **********************
 17- 17  2061 *************************
 18- 18  2321 *********************************
 19- 19  2443 ******************************
 20- 20  4015 ***********************************************
 21- 21  2303 *********************************
 22- 22  4193 *************************************************
 23- 23  3659 *******************************************
 24- 24  3639 *******************************************
 25- 25  3777 ********************************************
 26- 26  4015 ***********************************************
 27- 27  4046 ************************************************
 28- 28  3465 *****************************************
 29- 29  3334 ***************************************
 30- 30  2047 *************************
 31- 31  1187 ***************
 32- 32   633 ********
 33- 33   265 ****
 34- 34   141 **
 35- 35   102 **
 36- 36   134 **
 37- 37   131 **
 38- 38   297 ****
 39- 39   132 **
 40- 40   173 ***
 41- 41    44 *
 42- 42    43 *
 43- 43    12 *
 44- 44     8 *
 45- 45     2 *
 46- 46     2 *
```

(b) "I" Histogram

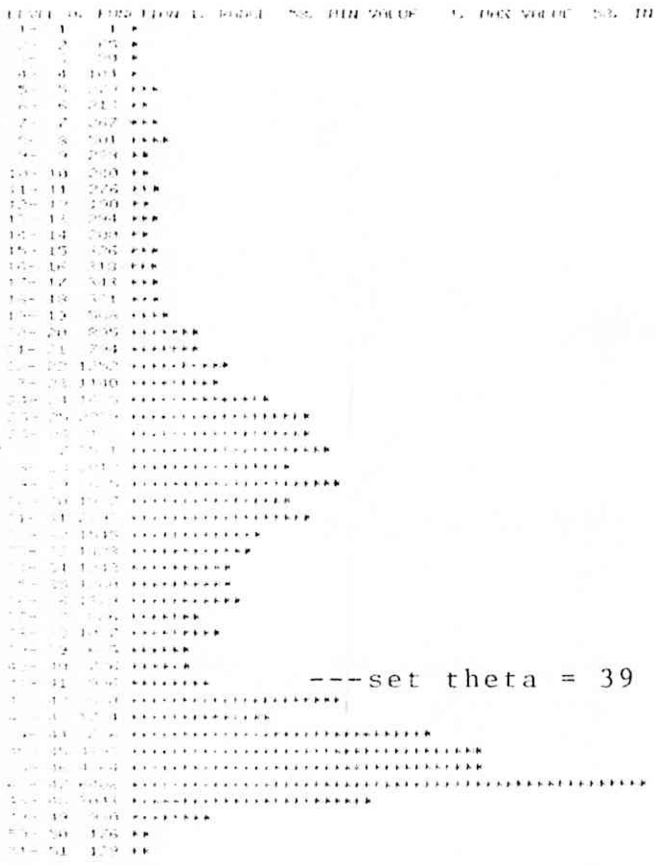
```
LEVEL 0, FUNCTION 3, RANGE  42, MIN VALUE    0, MAX VALUE  41, IN
  0-  0    11 *
  1-  1     1 *
  2-  2     4 *
  3-  3    12 *
  4-  4    37 *
  5-  5    50 *
  6-  6   105 *
  7-  7   193 **
  8-  8   572 ****
  9-  9   923 ******
 10- 10  1142 ********
 11- 11  1264 *********
 12- 12  1518 **********
 13- 13  1670 ***********
 14- 14  2123 **************
 15- 15  6952 ******************************************
 16- 16  8015 ************************************************
 17- 17  7016 ******************************************
 18- 18  3353 ********************
 19- 19  1773 ************
 20- 20  1625 ***********
 21- 21  1759 ***********
 22- 22  1401 *********                ---set theta = 22
 23- 23  1409 *********
 24- 24  1571 **********
 25- 25  1156 ********
 26- 26  1259 ********
 27- 27  1373 *********
 28- 28  1378 *********
 29- 29  1936 *************
 30- 30  2450 ****************
 31- 31  2842 *****************
 32- 32  3906 ************************
 33- 33  1990 *************
 34- 34  1298 *********
 35- 35   855 ******
 36- 36   314 **
 37- 37   147 *
 38- 38   100 *
 39- 39    22 *
 40- 40    10 *
 41- 41     1 *
```

(c) "Q" Histogram

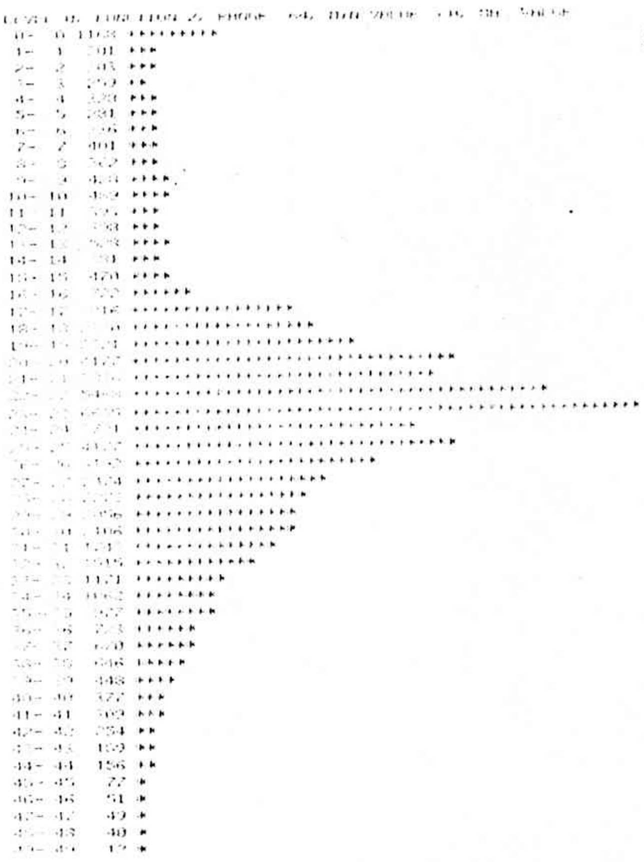


(d) PLAN based on Y and Q

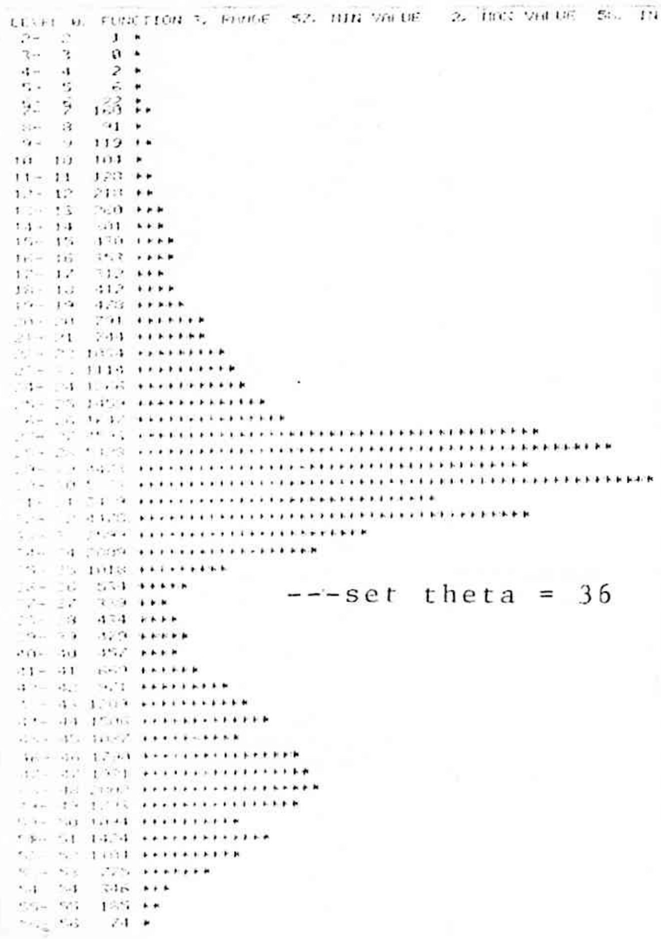
Figure 10: House Scene PLAN



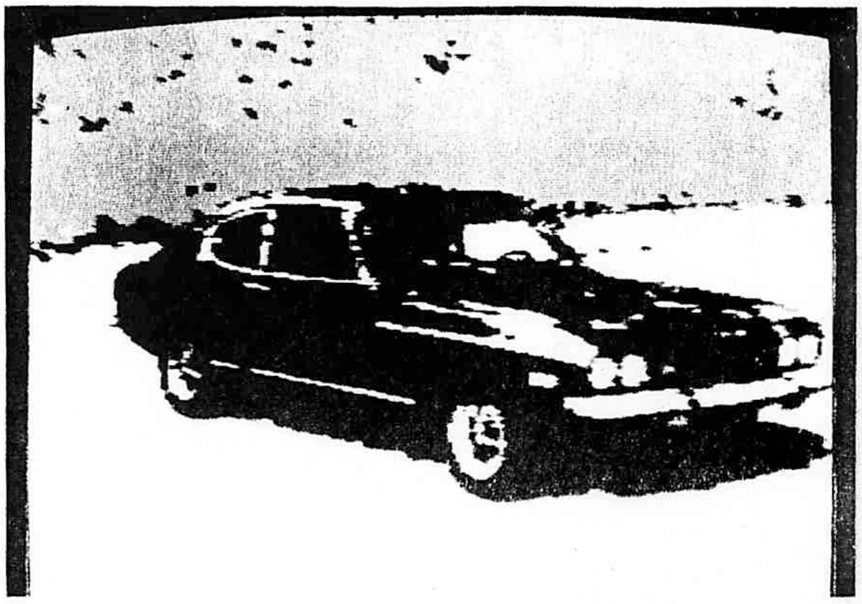
(a) "Y" Histogram



(b) "I" Histogram

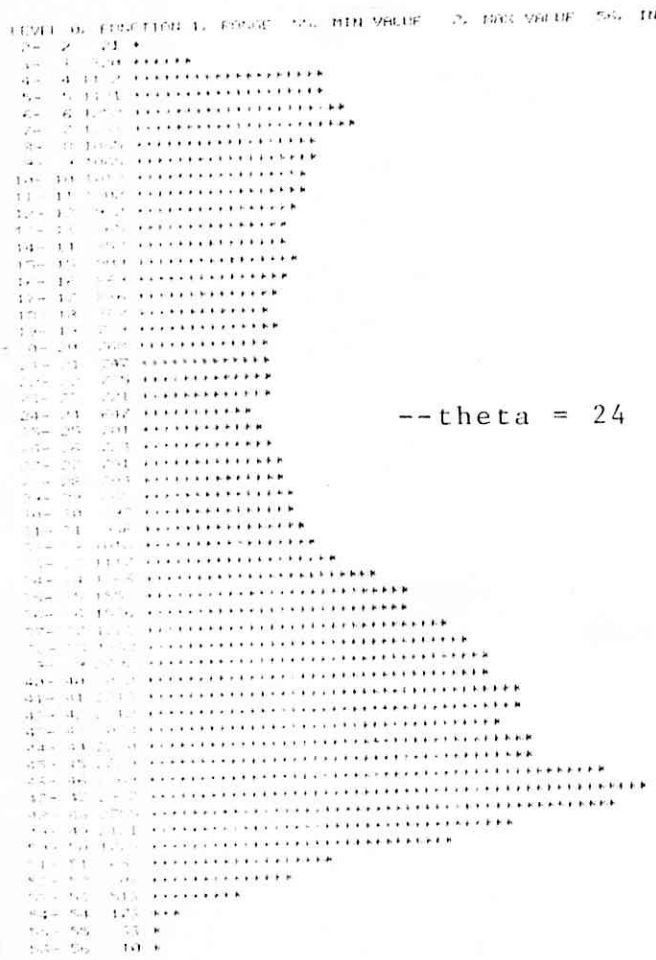


(c) "O" Histogram

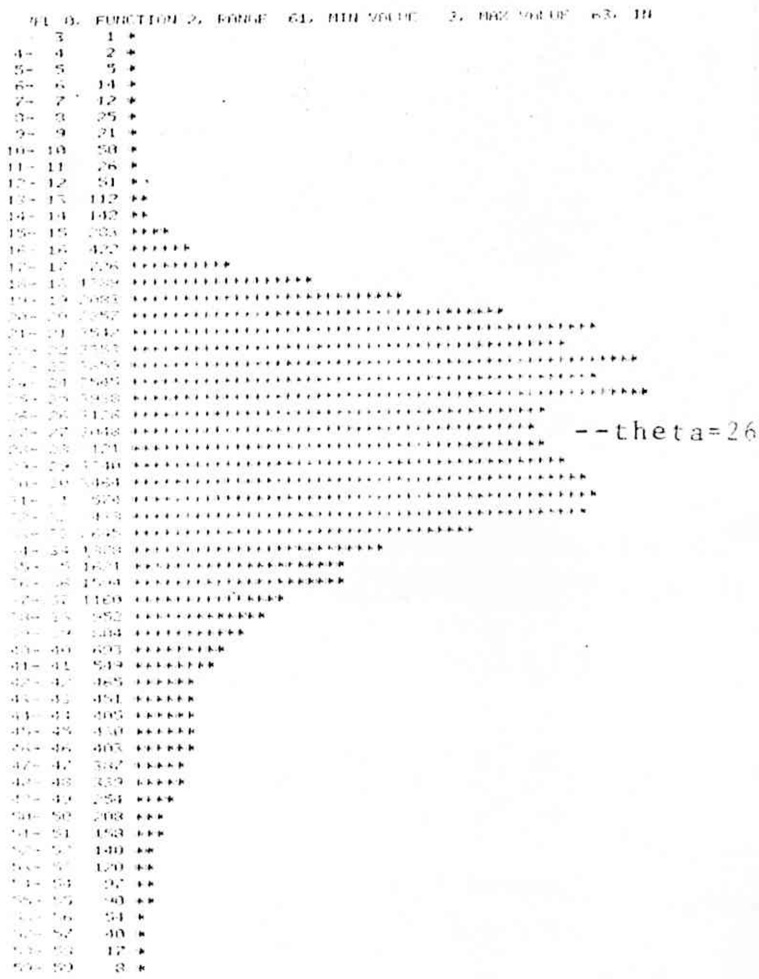


(d) PLAN based on Y and O

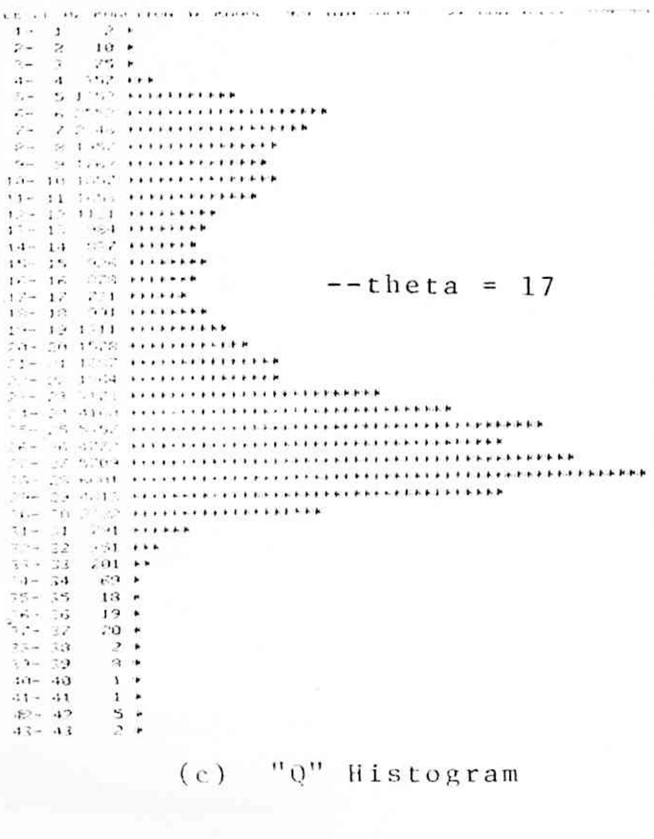
Figure 11: Car Scene PLAN



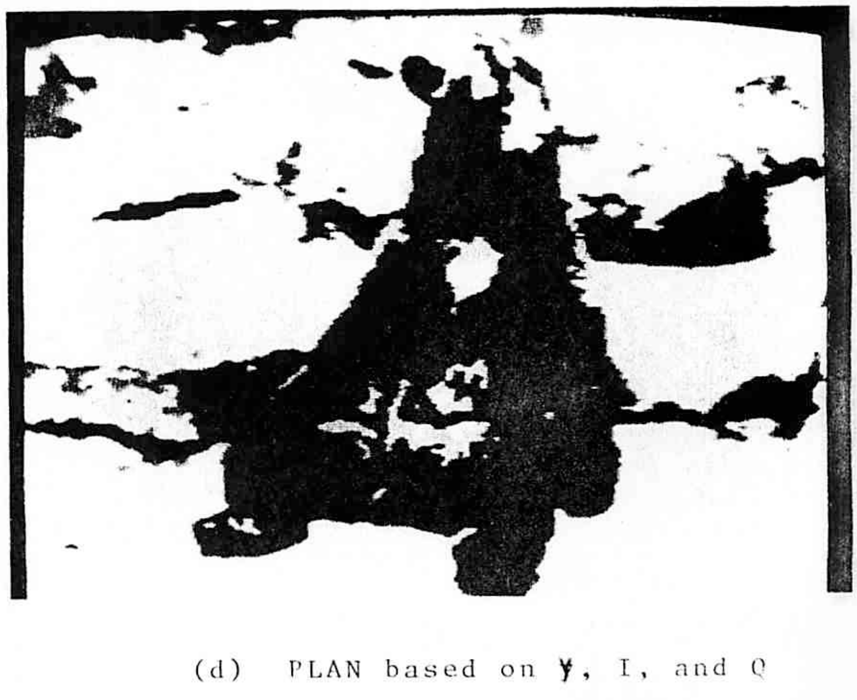
(a) "Y" Histogram



(b) "I" Histogram

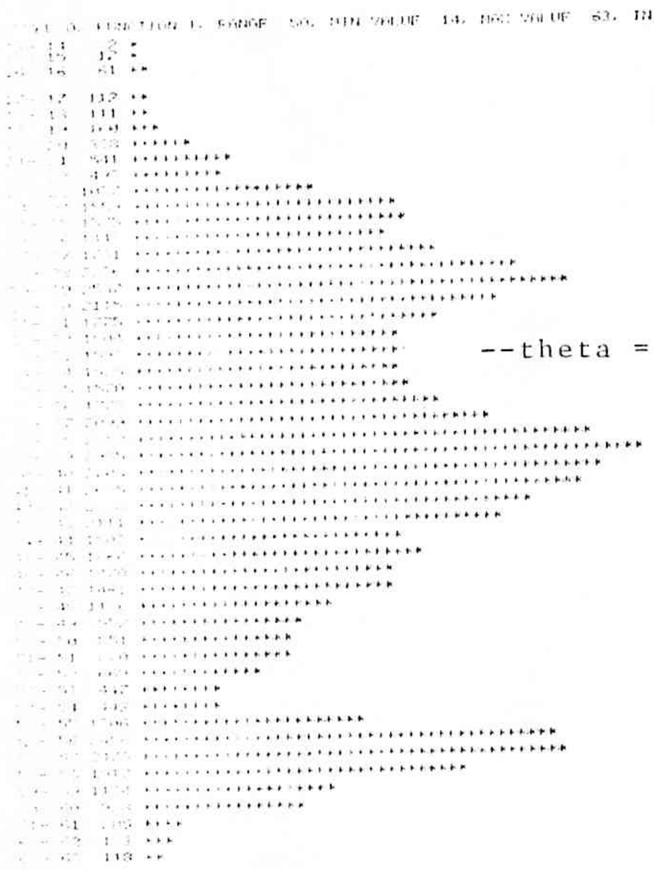


(c) "Q" Histogram

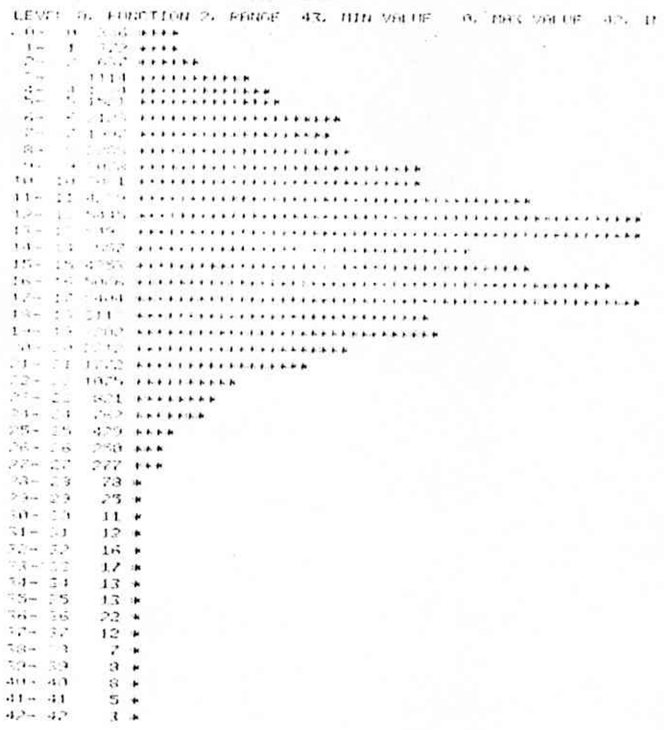


(d) PLAN based on Y, I, and Q

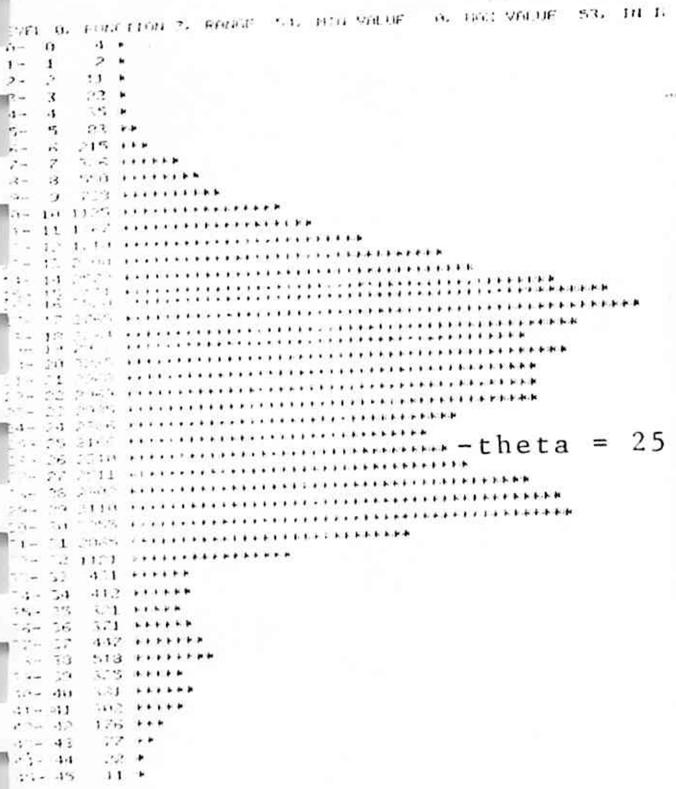
Figure 12: Bear Scene PLAN



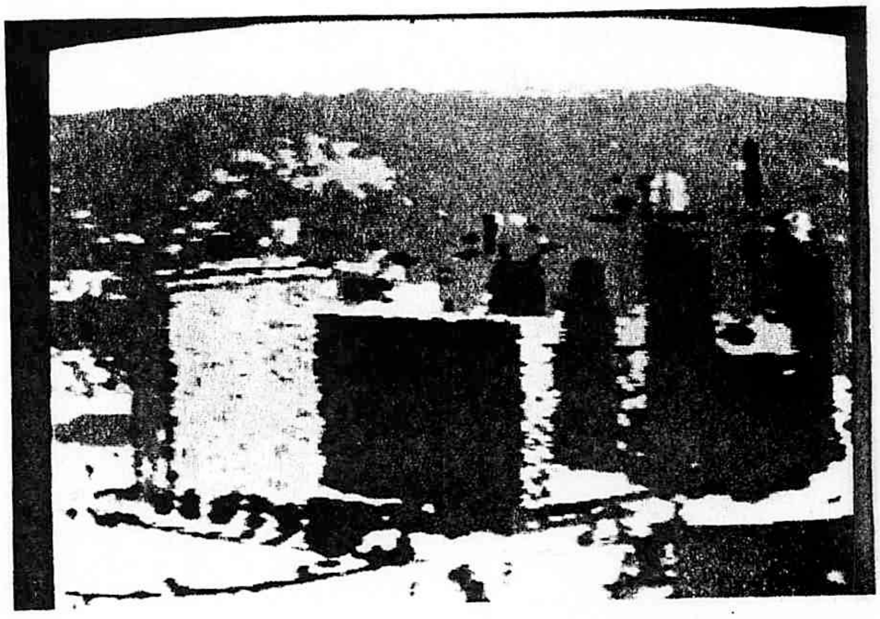
(a) "Y" Histogram



(b) "I" Histogram



(c) "Q" Histogram



(d) PLAN based on Y, I, and Q

Figure 13: Pittsburgh Scene PLAN

III.2 Planning Algorithms II: Spatial Compression

A second method of data reduction involves the spatial compression of an image. This transformation is carried out in the processing cone. Briefly stated, the cone is a simulation of a parallel computational data structure which facilitates the transformation and reduction of large amounts of data in a layered fashion. Features of the input data can be extracted during the reduction process so that each cell in a higher layer of the cone will contain features with a more global view of the data (see Figure 14). Here, the goal of the processing cone structure is to aid in the segmentation of scenes by allowing extraction of features that effectively collapse the textural variation in regions.

As an example of the utility of the cone, let us consider a textured region where the distribution is strongly bimodal ("salt and pepper"). A simple region grower applied here would probably be ineffective due to the local variations in the data and would generate a severely fragmented segmentation. Further, the histogram-guided algorithm would find two cluster types for the pixels; the region would be fragmented by this approach also. On the other hand, by averaging the information over a local window, the data can be smoothed so that internal variations of the region are greatly reduced. Now the histogram-guided algorithm will find one cluster type for the pixels and the region can be effectively labelled. Notice that other features can be extracted over local areas while reducing the total amount of information. Since the higher levels of the cone do not contain much data, the extraction of features and coarse labelling of regions can be done at a great time savings.

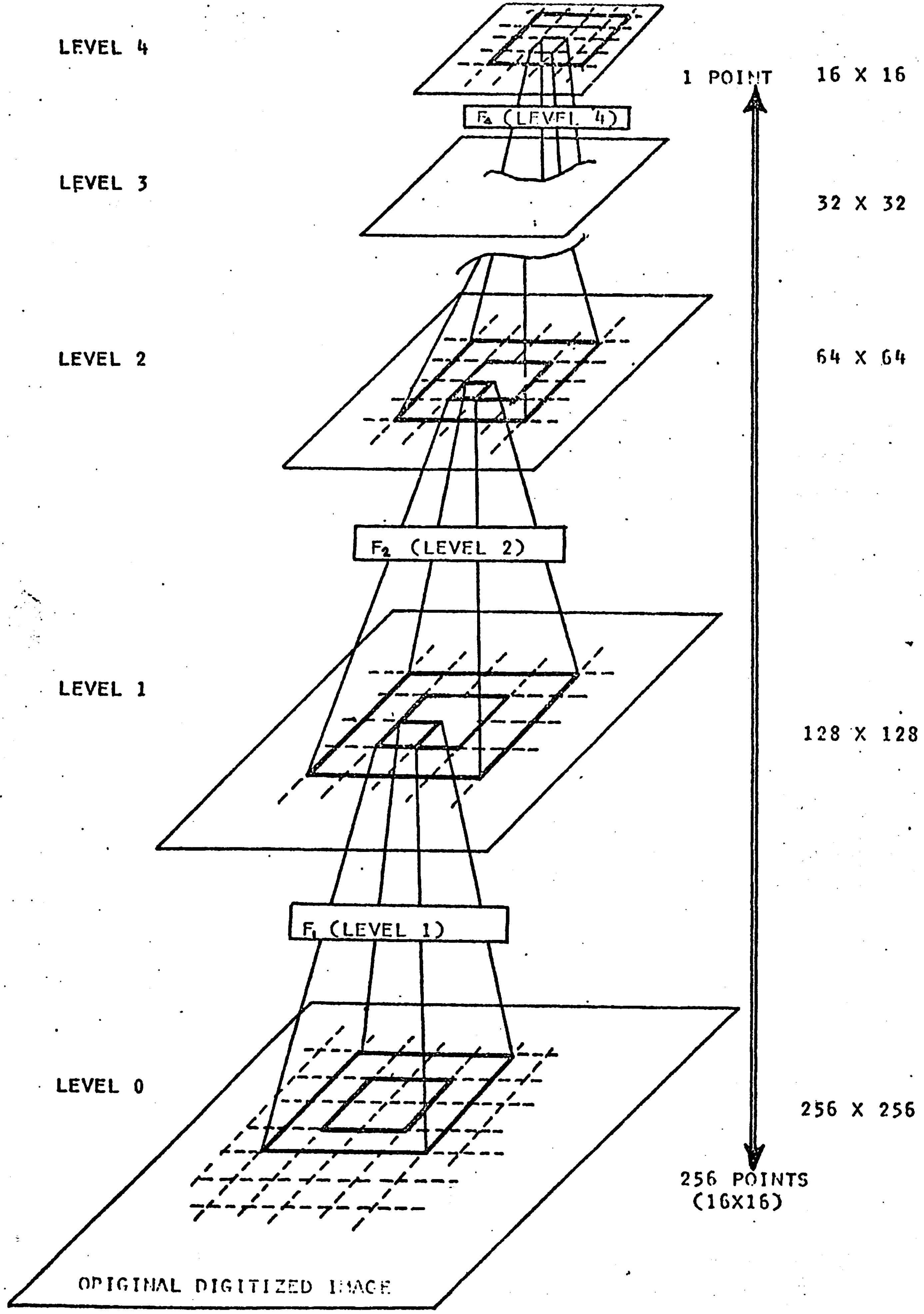


Figure 14 Processing Cone

Now we will examine a somewhat different use of the cone -- the labelling with the same symbolic label of spatially disjoint regions which have the same visual features. This involves the reduction of the symbolic information in the plans formed by the histogram compression algorithm. The plan labels themselves can be "reduced" by passing upward the most frequently occurring label in a 2×2 area. The labels which reside at the top layers of the cone would represent the largest regions of the image.

By maintaining a hierarchy of possible segmentations, a "correct" segmentation may be assembled by a knowledge directed system as a function of the goal of the system or degree of effort to be expended. A knowledge-directed system may determine the correct level of description by fitting interpretations to different levels in the hierarchy or by extracting useful levels depending on the size and properties of regions.

There are problems encountered in the reduction process which should be pointed out. First, each region of the data will be best collapsed at some corresponding cone level. However, it is not obvious a priori what level that will turn out to be. For fine grass texture a relatively low (unreduced) level of the cone might allow a reasonable statistical sample of the textural feature, while a higher level of the cone will be needed in order to encompass a number of the macro-texture elements for the gross shadow and highlight areas of the tree within the receptive field of a single cell. Of course, the optimum level is also a function of the size of the object in the image.

The implication of this discussion is that, at any given cone level, some cells will represent data at the proper level of reduction for producing a meaningful texture measure of a region, while other cells at the same level will represent features of a region extracted from a receptive field that is too small or too large. Therefore, we can expect that "mutant" features will be produced when the reduction window is too large and overlaps two regions with distinct characteristics. In this respect the process is blind and the windows containing adjacent textures will be treated as if they contained a single texture. In addition, small regions will be merged to form single regions which will contain, perhaps, none of the characteristics of the original data.

The problems that have been described are not problems of the cone. Rather they are problems to be faced by a hierarchical view of textural variations without knowledge of the appropriate level of analysis in each area. These difficulties seem to be inherent in the problems of texture [3]. In light of these considerations, the higher, more compressed, layers of the cone must be used cautiously -- but they do provide a coarse organization of the data at little computational expense.

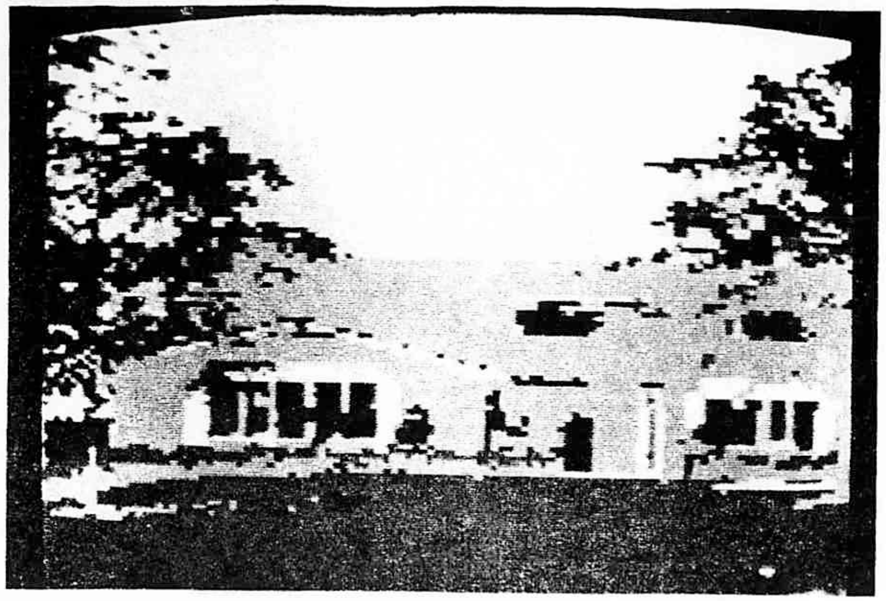
Combining Spatial Reduction and Quantization Compression

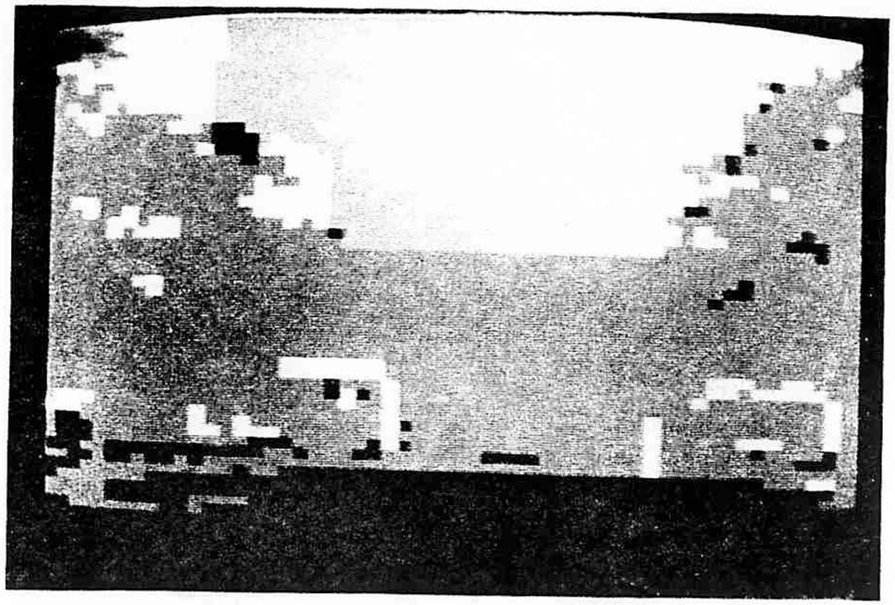
The following two examples indicate two uses of the cone during the planning stage. Figure 15 demonstrates the strategy of spatial reduction followed by quantization compression. The data was obtained as follows:

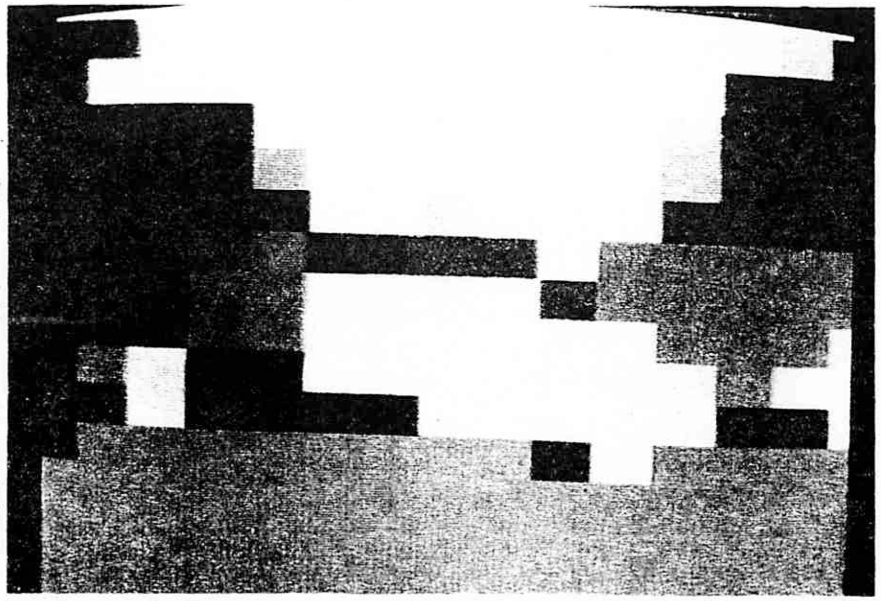
(I) Reduce raw input data (red, green, blue) by averaging 2 × 2 windows. Reduction takes place from level Ø (256 × 256 pixels) through level 4 (16 × 16 pixels);

(II) At each level 1-4 perform the following steps:

(a) compute Y, I, and Q;
(b) cluster each feature via quantization compression to form a binary image;
(c) intersect the binary images to form an image with at most 8 labels;
(d) clean up the resulting symbolic image in the following manner: change the label of a pixel to that of the most frequently occurring frequency, i.e., > 80% of the window.


(a) Cone level 1 (128x128)


(b) Cone level 2 (64x64)


(d) Cone level 4 (16x16)

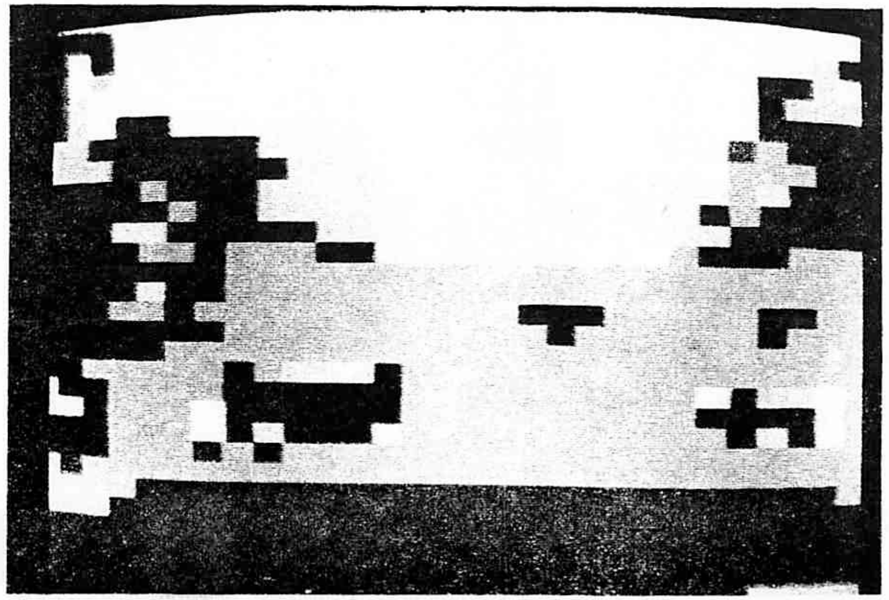

(c) Cone level 3 (32x32)

Figure 15: Spatial Reduction followed by Quantization Compression

This figure clearly illustrates the window problem mentioned earlier. As the window size increases (corresponding to higher cone levels), we see small regions merging into larger surrounding regions (e.g., trim around windows) and large regions becoming fuzzy at their boundaries. Textural variations are greatly smoothed over, leaving a small set of relatively homogeneous regions at level 4 of the cone (16 × 16 pixels). The poor quality of the segmentation at level 2 was the worst result in all the experiments run. Obviously, very little has been gained by this plan; the low-level system can recover from this, though, during the refinement stage where a more detailed and careful analysis can be performed.

Notice that the coarseness of the sampling leads to false contours in the images presented. This is demonstrated in the area of the right tree, for example, where the object has been split into two separate regions. Of course, from another point of view the system is distinguishing textured tree with sky showing through from textured tree with roof showing through. In general, these four segmentations illustrate the observation made by Rosenfeld [15]:

> "In a slowly changing scene it is important to have fine quantization, but the sampling can be coarse, while in a scene with a large amount of detail, it is necessary to sample finely, but quantization can be coarse."

A more useful analysis is shown in Figure 16 which demonstrates the strategy of quantization compression followed by spatial reduction. The following steps were performed:

(I) compute Y, I, Q from raw input data (red, green, blue) at level Ø (256 × 256 pixels);

(II) cluster each feature via quantization compression to form a binary image;

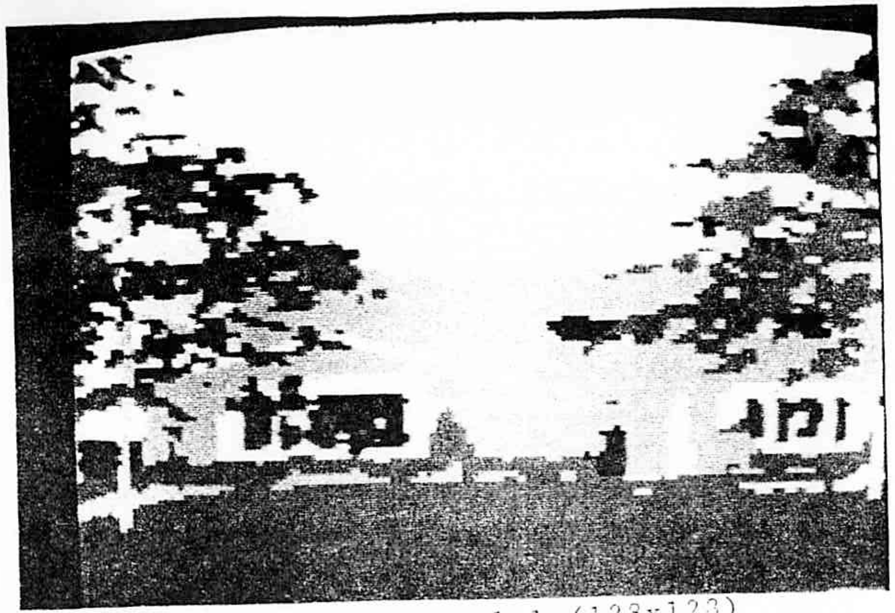
(III) intersect the binary images to form a plan with 8 labels;

(IV) reduce the plan from level Ø through level 4 by passing upward the symbolic region label which is the majority in the 2 × 2 windows.

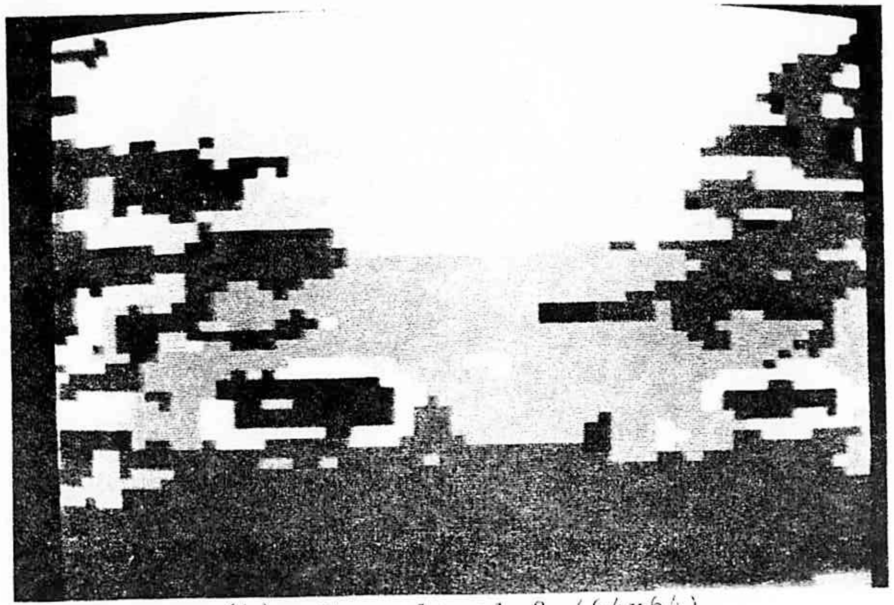
Figure 16 presents a set of plans which are more desirable than those in Figure 15. The plan from which these are derived (Figure 10) was generally good but benefitted by some cleaning up in the tree region. By selecting for large regions, the reduction of the plan ignores small local subpatterns while maintaining the most important information of the original plan; namely that the picture consists of five major regions, corresponding to sky, grass, house, and the two trees. This is shown in Figure 16d. Distinct region labels formed by region growth on the small amount of data a high level can be projected down to level Ø. Then disconnected portions of the tree will be given the same label. This can be done without the necessity of a complex and time-consuming process performed on the mass of data of the original image.

III.3 Refinement of Plans

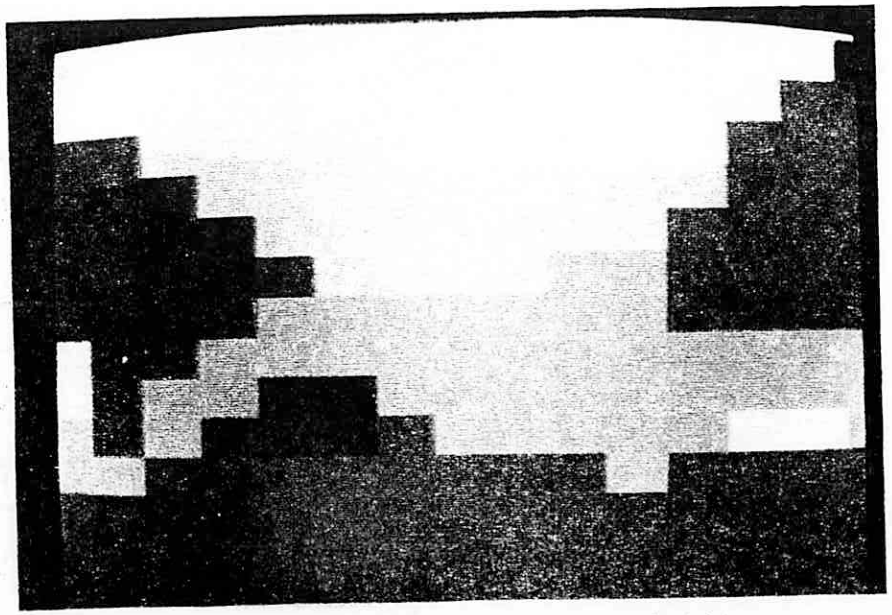
The coarseness of the planning stage will yield, in most cases, a segmentation which contains grossly overmerged regions. In order to effect an accurate segmentation, the system will have the ability to recursively apply the histogram-guided compression technique to a region of the plan at each lower level of resolution in the cone. This may not have to be performed on all regions since these algorithms are eventually to be embedded in a process with feedback from semantic high-level processes. For example, the refinement process may be guided by object verification strategies so that some portions of the hierarchical plan representation are analyzed to a greater level of detail.



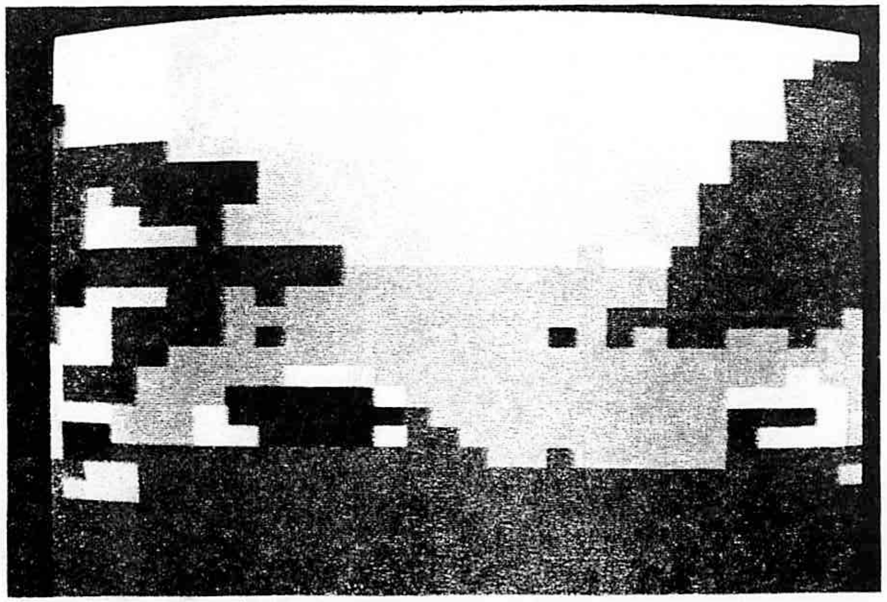
(a) Cone level 1 (128x128)



(b) Cone level 2 (64x64)



(d) Cone level 4 (16x16)



(c) Cone level 3 (32x32)

Figure 16: Quantization Compression at Level 0
followed by Symbolic Reduction of the PLAN

One region of the plan can be analyzed with a 2D histogram based on different features. However this recursive histogramming of regions and subregions can lead to serious computational problems, because the number of regions in textured scenes can increase exponentially as finer textural detail is extracted.

Once the texture components have been assigned unique labels, the actual texture is difficult to describe, particularly if statistical descriptors are desired. On the other hand, structural properties of the texture elements can be extracted. It is possible to avoid fragmentation of textured regions in those cases where many small disconnected regions are nearby and ought to have the same region identifier. Here the cone can be employed be extracting the majority region label from a local area and forming a labelled image at a coarser level of resolution (at a higher level in the cone). Then, a region in the coarser resolution plan could represent a set of disconnected areas at levels below. It is an inexpensive process to map this particular label to many regions at finer levels of resolution. In the processing cone, simulated parallel hardware performs such operations. However, it is even more important to avoid histogramming regions of micro-texture. Thus, Price [10] used planning to reduce his data and the impact of the problem. This leads to a need for hierarchical region description and some way of determining that the system has moved down to the level of "micro-texture" (which of course is a relative term).

It is hoped that the use of the higher dimensional segmentation will limit this potentially recursive analysis to just one refinement step. We outlined in a previous paper [12] a technique for two-dimensional clustering. Briefly, the algorithm involves placing the histogram into the cone and treating it as a pseudo-image. The histogram is blurred up the cone by averaging, and it is scaled so that valleys between clusters disappear. Thus, clustering becomes a problem analogous to region isolation except that from a textural point of view the histogram data is usually much simpler than the original image data. It is highly unlikely, therefore, to see a "salt and pepper distribution;" rather, a more probable description for a two-dimensional plot would be a "continuous gradient" in the histogram (not necessarily correlated at all to spatial gradients in the image). A gradient detector [16] could be used to isolate cluster centers, and region growing could be used to add on peripheral layers to the degree desired (refer to conservative labelling in Section II.2).

Figures 17 and 18 illustrate the refinement process using two-dimensional histograms. Here, for convenience, clustering was done by hand simply by thresholding the histogram so that the very low magnitudes which link the cluster centers were turned off. This was not automated because we wanted to see the limit in effectiveness of this process without errors introduced via unsophisticated clustering methods. Region growing was then applied to the isolated cluster centers to yield the results shown.

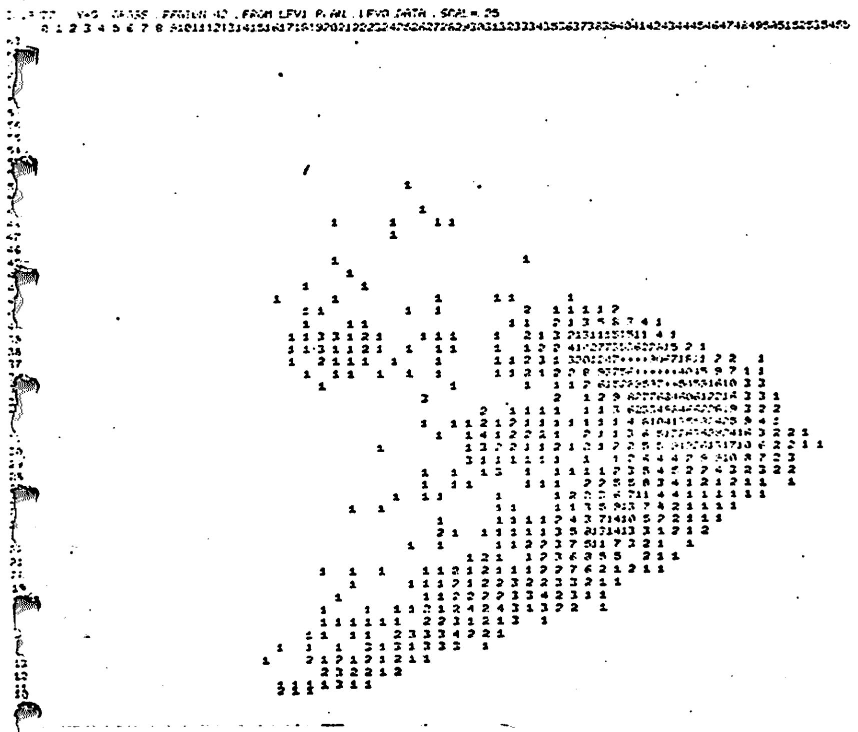
Figures 17a and 17b show the Y and Q distributions for the grass/shrub region of Figure 1. These histograms would be considered uni-modal (i.e., no threshold point is apparent) by the clustering algorithm outlined in Figures 7-9 and thus would not offer any new segmentation information for this region. On the other hand, the two-dimensional histogram shown in Figure 17c reveals two distinct subclusters (corresponding to the (1) grass and (2) shadow regions). These clusters are isolated and labelled as shown in Figure 17d. Note that clustering was done interactively by the following steps:

```
3/19/77. . Y . GRASS + SHRUBS. . LEV1 PLAN. . LEV0 DATA
VAL MAGNITUDE      TOTAL NUMBER OF POINTS = 16380

  9     2    *
 10    13    *
 11    34    **
 12    46    **
 13    54    **
 14    81    ***
 15    96    ***
 16    80    ***
 17   125    ***
 18   106    ***
 19   102    ***
 20   133    ****
 21   150    ****
 22   182    *****
 23   270    ******
 24   217    *****
 25   203    *****
 26   195    *****
 27   174    ****
 28   193    *****
 29   300    ********
 30   502    ************
 31   723    *****************
 32   928    *********************
 33  1329    ******************************
 34  1517    **********************************
 35  2101    ***********************************************
 36  2544    *********************************************************
 37  2200    **************************************************
 38  1171    ***************************
 39   362    *********
 40   122    ***
 41    44    **
 42    22    *
 43    13    *
 44     4    *
 45     9    *
 46     3    *
 47     8    *
 48     9    *
 49     5    *
 50     3    *
 51     4    *
 52     1    *
```

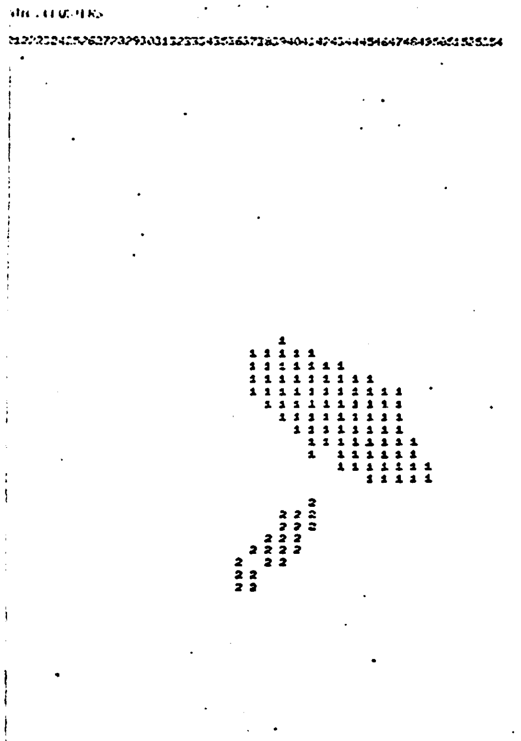
(a) "Y" Histogram of Grass

```
3/19/77. . Q . GRASS + SHRUBS . LEV1 PLAN . LEV0 DATA
VAL MAGNITUDE      TOTAL NUMBER OF POINTS = 16380

  0     1    *
  1     0    *
  2     0    *
  3     0    *
  4     0    *
  5     1    *
  6     0    *
  7     0    *
  8     0    *
  9     0    *
 10     0    *
 11     1    *
 12     0    *
 13     3    *
 14     0    *
 15     3    *
 16    19    *
 17    27    **
 18    44    **
 19    65    **
 20    80    ***
 21    54    **
 22    77    ***
 23    37    **
 24    61    **
 25    54    **
 26    90    ***
 27    84    ***
 28    94    ***
 29    60    ***
 30   106    ***
 31   103    ***
 32   107    ***
 33   106    ***
 34   117    ***
 35   157    ****
 36   225    ******
 37   432    **********
 38   501    ***********
 39  1095    ***********************
 40  1469    *******************************
 41  2304    ************************************************
 42  1998    ******************************************
 43  2442    ***************************************************
 44  1206    **************************
 45  1500    ********************************
 46   630    **************
 47   538    ************
 48   224    ******
 49   142    ****
 50    44    **
 51    44    **
 52     9    *
 53     6    *
```

(b) "Q" Histogram of Grass



(c) Y * Q Histogram of Grass



(d) Conservative labelling of strong clusters

Figure 17: Refinement of Grass PLAN-Region by 2-D Histogram

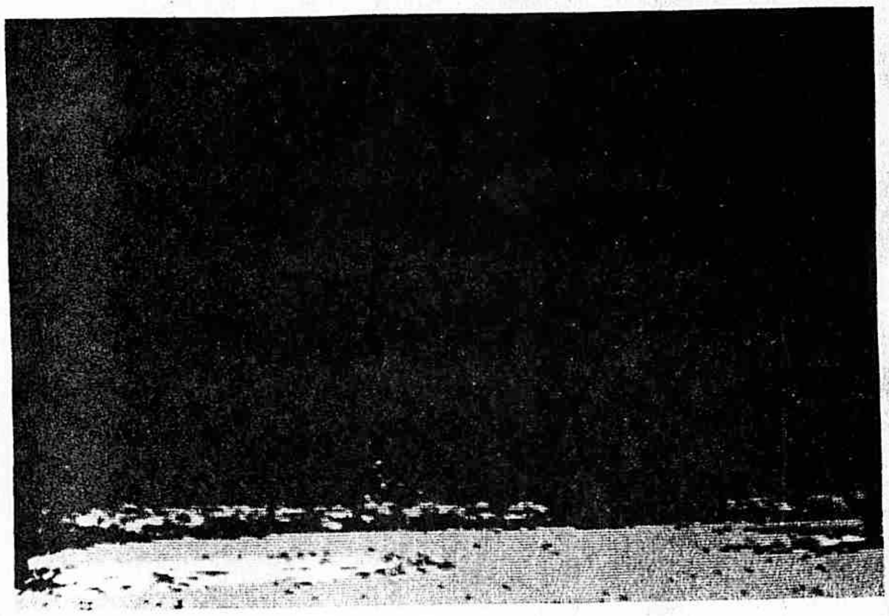


Figure 17e: Map clusters found in Figure 17d back to corresponding image pixels. Note appearance of shadow region (bright area).

(1) place the histogram into the cone at level 2 (64 × 64 pixels);

(2) suppress (i.e., set to 0) all points below a user-supplied threshold;

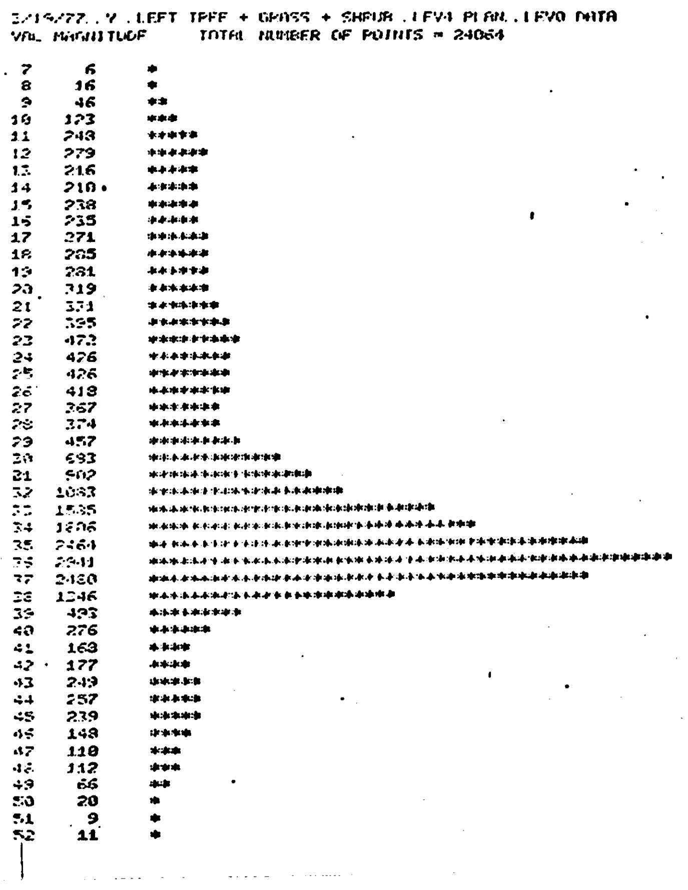
(3) region grow (join adjacent non-zero points) to label all connected points in a cluster with a unique symbolic label.

Figure 17e shows the region cores which resulted from mapping the clusters which were labelled (Figure 17d) to the corresponding pixels in the image. The holes which appear in these regions result from the use of the conservative clustering algorithm which labels only the very well-defined cluster centers. In most cases, these holes can be filled in with a reasonable heuristic; namely, a hole which is mostly surrounded by a core region takes on the label of that region.

Figures 18a and 18b show the Y and Q distribution for the grass/shrub and left tree "region" (the image at level 4 of Figure 16 indicates how this large area of the image could be labelled as a single region). The two-dimensional histogram in Figure 18c and cluster labels in Figure 18d serve as a further illustration of the power of this higher-level representation -- clusters which would have required a recursive clustering analysis in the one-dimensional case are found in one step here.

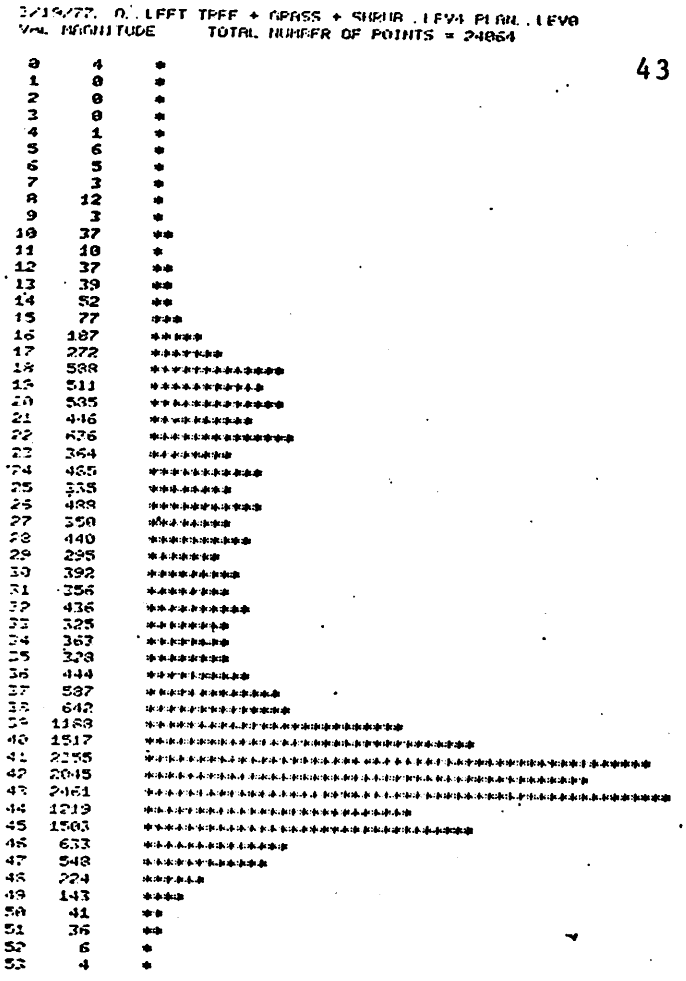
III.4 Second Order Analysis of Texture

Finally, we would like to outline a second order analysis of texture through the use of spatial adjacency matrices. We wish to characterize the various levels of textural detail of a region by analyzing distributions of local feature activity of adjacent cells. Information which clusters around the main diagonal indicates relatively small differences in activity. Conversely, information on which is off the main diagonal indicates relative heterogeneity of the image data.



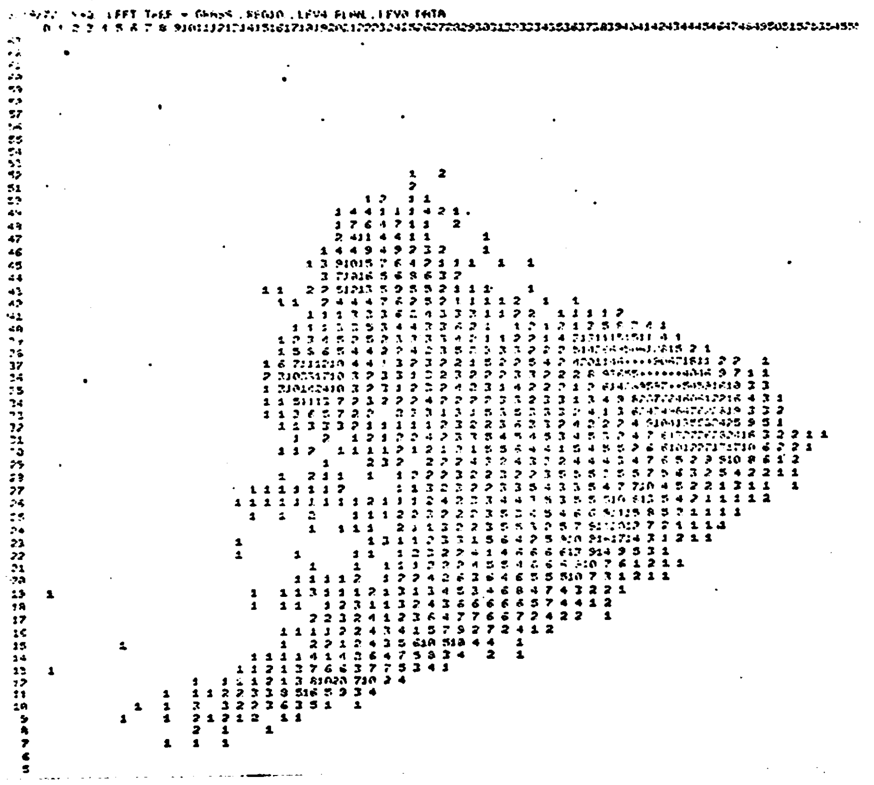
```
3/19/77. Y .LEFT TREE + GRASS + SHRUB .LEV4 PLAN.LEV0 DATA
VAL MAGNITUDE     TOTAL NUMBER OF POINTS = 24064

 7     6   *
 8    16   *
 9    46   **
10   123   ***
11   243   *****
12   279   ******
13   216   *****
14   210   *****
15   238   *****
16   235   *****
17   271   ******
18   285   ******
19   281   ******
20   319   ******
21   371   *******
22   395   ********
23   473   *********
24   426   ********
25   426   ********
26   418   ********
27   367   *******
28   374   *******
29   457   *********
30   693   **************
31   902   ******************
32  1083   **********************
33  1535   *******************************
34  1806   ************************************
35  2464   *************************************************
36  2941   ***********************************************************
37  2430   *************************************************
38  1246   *************************
39   493   **********
40   276   ******
41   168   ****
42   177   ****
43   249   *****
44   257   *****
45   239   *****
46   148   ****
47   110   ***
48   112   ***
49    66   **
50    20   *
51     9   *
52    11   *
```

(a) "Y" Histogram of Grass/Tree

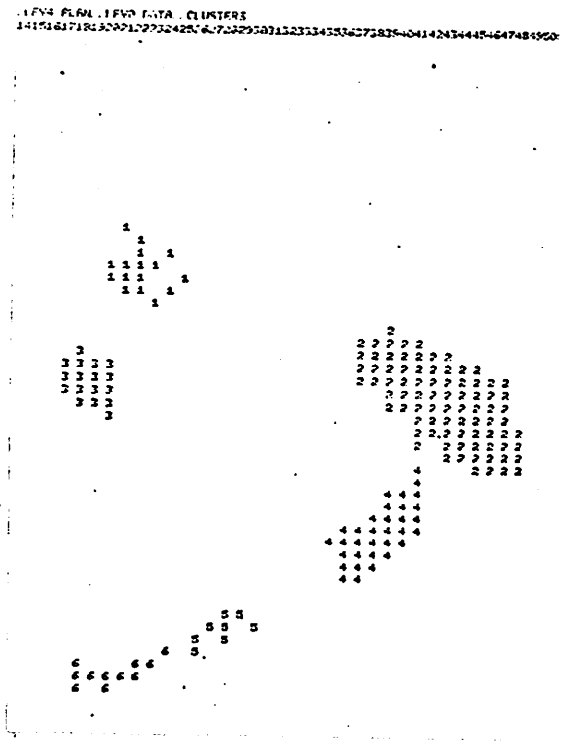


```
3/19/77. Q..LEFT TREE + GRASS + SHRUB .LEV4 PLAN.LEV0
VAL MAGNITUDE     TOTAL NUMBER OF POINTS = 24064

 0     4   *
 1     0   *
 2     0   *
 3     0   *
 4     1   *
 5     6   *
 6     5   *
 7     3   *
 8    12   *
 9     3   *
10    37   **
11    10   *
12    37   **
13    39   **
14    52   **
15    77   ***
16   187   *****
17   272   *******
18   598   **************
19   511   ************
20   535   **************
21   446   **********
22   676   ***************
23   364   *********
24   485   ************
25   335   *********
26   488   ************
27   350   *********
28   440   **********
29   295   *******
30   392   *********
31   356   ********
32   436   **********
33   325   ********
34   363   ********
35   328   ********
36   444   *********
37   587   **************
38   642   ***************
39  1188   ***************************
40  1517   ***********************************
41  2255   ************************************************************
42  2045   ***********************************************
43  2461   *****************************************************************
44  1219   ****************************
45  1503   ***********************************
46   633   ***************
47   548   ************
48   224   ******
49   143   ****
50    41   **
51    36   **
52     6   *
53     4   *
```

(b) "Q" Histogram of Grass/Tree



(c) Y * Q Histogram of Grass/Tree



(d) Conservative labelling of strong clusters

Figure 18: Refinement of Grass/Tree PLAN-Region by 2-D Histogram

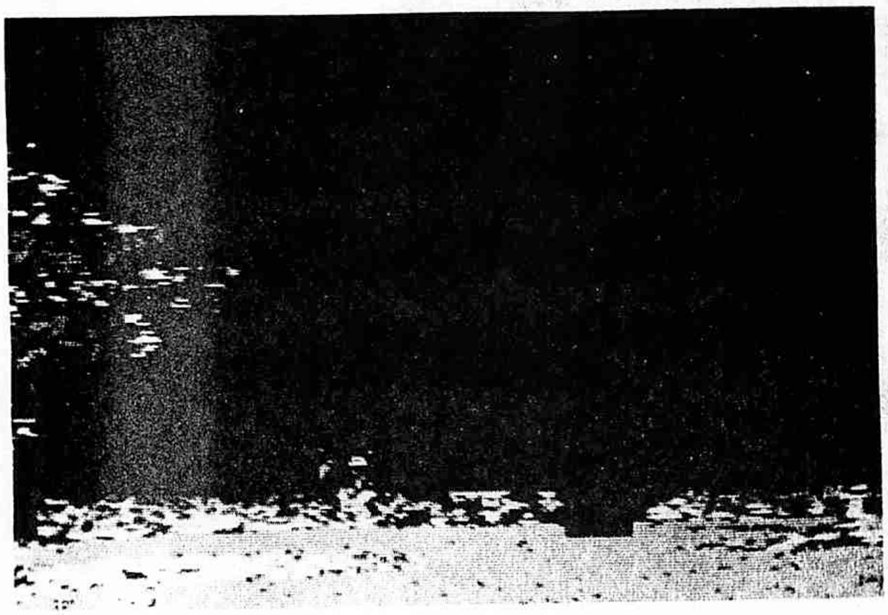


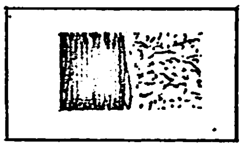
Figure 18e: Map clusters shown in Figure 18d back to corresponding image pixels. Note texturing of cluster labels in tree.

For our purposes, the usefulness of this representation lies in its ability to refine overmerged PLAN regions. The histogram analysis is insensitive to the structure of the textural properties of the data, and therefore the following type of situation is likely to occur. Suppose that a portion of an image contains a relatively homogeneous dark region which is adjacent to a speckled texture region (Figure 19). The histogram analysis will assign one region label to all connected (8-adjacent) dark pixels and a different label to all connected bright pixels (shown in Stages I and II). These regions can then be symbolically reduced in the cone in order to retain only the coarsest amount of detail in the image, e.g., the most frequently occurring labels (Stage III). By collapsing all textural variations, a region mask can be produced from a set of adjacent identical labels that have been brought up from below. However, each label in a particular mask could have been produced by many different variants of the data, i.e., there is a many-one mapping and the underlying data may not be as uniform as the plan suggests. Now the high-level mask can be used as a guide to further analysis. The adjacency matrix of the data which lies within the receptive field of the mask contains two clusters, corresponding to dark vs. dark and dark vs. bright (Stage IV). Finally, a cluster labelling algorithm can be applied and the corresponding region-labelling will yield a segmentation (Stage V).

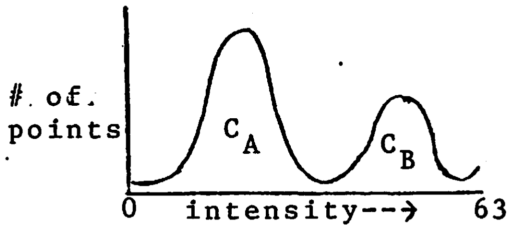
Notice that once the high-level mask has been created, the adjacency matrix can be measured across either the full-resolution, full-quantization image data or the quantization compressed plan data. In the latter case the adjacency analysis would take into account the first-order (pointwise

Analysis of Textural Properties of Regions

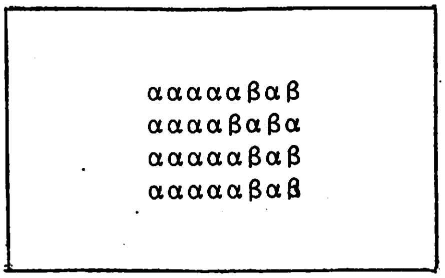
Example sub-image



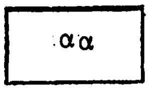
Stage I: Histogram and Label



Stage II: Map Labels back to Image and Region Grow to form PLAN.

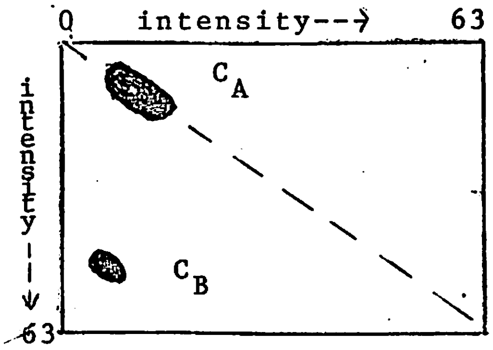


Stage III: Reduce PLAN Regions by selecting most frequent Region Label in 4x4 Window.



cone level i, region mask α

Stage IV: Plot Adjacency Matrix of Image Data using High-Level Reduced PLAN as a Mask.



Stage V: Project Clusters onto Image Data.

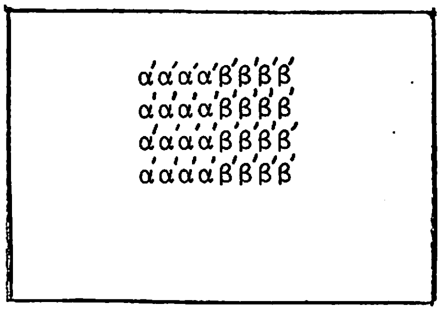


Figure 19

feature) clustering of the data as obtained from the initial analysis. By using the PLAN region labels as input to the adjacency matrix, this representation will be much clearer in that it will not be obscured by very small (noise) differences.

In addition to texture discrimination, the adjacency matrix can be used for texture description. Features such as angular second moment difference (heterogeneity) and entropy (information) have been used by Haralick [11] for classification. However, texture description is more difficult. Scalar features of the adjacency matrix can be computed and used in a descriptor list which could be associated with each region. We expect though, that this representation will not be sufficient because much of the information is being thrown away. Research into this problem is now in progress.

APPENDIX - PREPROCESSING

IV.1- Removing Mixed Pixels and Use of Conditional Blurring

The images used by the low-level system are 256 x 256 arrays of pixels digitized to 6 bits per pixel. There are three planes of data obtained by scanning the image through red, green, and blue filters. The total amount of data which must be processed is approximately 1.2 million bits.

The input data is preprocessed in two stages. First, an operator is applied which will remove gradients (introduced by digitization effects) at boundaries. In the initial scanning process the beam will sometimes measure a light intensity which falls across a boundary and therefore will record an average brightness value between the values on either side of the boundary. The data which results from this process are called mixed pixels and can be identified with a local detector [17]. The algorithm will unmix when exactly a two-pixel gradient is detected in all three input planes (red, green, and blue). Once found, a mixed pixel can be assigned the value of the adjacent pixel which is closest to it in the three features and lies along the gradient. In the sample window of data shown below, the center point would be detected and unmixed as indicated.

	(35,40,20)	
	(20,30,15)	
	(2,10,9)	

MIXED

Two-pixel gradient around center pixel. Pixels shown as red, green, and blue triples.

$\longrightarrow$

	(35,40,20)	
	(35,40,20)	
	(2,10,9)	

UNMIXED

De-blur center pixel to point it is closest to in red, green, and blue planes.

A second preprocessing stage is employed whereby data is averaged only if the total amount of local pairwise difference over a window is small. This <u>conditional blurring</u> operation [18], which is similar to an algorithm presented in [19], will have the effect of smoothing finely textured areas while leaving strong boundaries intact. The algorithm can be described as follows:

n_1	n_2	n_3
n_4	n_0	n_5
n_6	n_7	n_8

$$n_0 = \frac{1}{n_s} \sum_{n_i \in S} n_i$$

where:

n_0 = output value for n_0

$S = \{n_i \mid |n_i - n_0| < T\}$

T = preset threshold

n_s = # elements in s

IV.2 Color Transformation

Researchers in computer vision are faced with a bewildering set of alternative color spaces which can be used for their many purposes. Various considerations have led us to adopt the YIQ color space for initial segmentation. The computation of YIQ is linear and it appears to be a simple approximation of the <u>opponent color process</u> which some theorists believe takes place in the eye [20]. The Y component essentially measures intensity, while the I and Q opponents, which respectively measure cyan-orange and magenta-green, transform the image in such a way as to heighten color contrast. This is a particularly desirable effect for histogram-guided operations in that there is a tendency to force bi-modality in the distributions.

The formula for YIQ is as follows:

$$\begin{array}{c} Y \\ I \\ Q \end{array} = \begin{bmatrix} .509 & 1.000 & .194 \\ 1.000 & -.460 & -.540 \\ .403 & -1.000 & .597 \end{bmatrix} * \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Since the I and Q opponents can take on negative values, these features may need to be shifted into a positive range. In addition, when working with integer data, it has been found to be useful to spread the distribution by some small linear scale (i.e., multiply by 1.5 before truncation). The necessity for scaling arises from the fact that natural scenes tend to have low saturation (white-washed) and thus the range of I and Q is quite compressed. Much of the information which would be lost in the process of integer truncation can be recovered by scaling the distribution beforehand.

Note that in the histograms presented in this paper we have changed the sign of the Q component. This is done so that it will more or less correspond to the green gun of our color monitor display.

A full discussion of the behavior of YIQ, as well as other color transformations, is given in a recent paper by Kender [21]. He shows that non-linear transformations such as hue and normalized red, green, and blue, are unreliable as features due to the essential singularities in their distributions, i.e. small perturbations of the input data can cause arbitrarily large changes in the output of these features.

V. References

[1] A. Hanson and E. Riseman, "The Design of a Semantically Directed Vision Processor (Revised and Updated)," Computer and Information Science Department Technical Report 75C-1, University of Massachusetts, February 1975.

[2] A Hanson and E. Riseman, "A Progress Report on VISIONS: Representation and Control in the Construction of Visual Models," Computer and Information Science Department Technical Report 76-9, University of Massachusetts, July 1976.

[3] E. Riseman and M. Arbib, "Computational Techniques in the Visual Segmentation of Static Scenes," to appear in IEEE Transactions on Computers, June 1977.

[4] J.A. Feldman and Y. Yakimovsky, "Decision Theory and Artificial Intelligence: I. A Semantics Based Region Analyzer," Artificial Intelligence, Volume 5, Number 4, 1974.

[5] J.M. Tenenbaum and H.G. Barrow, "Experiments in Interpretation-Guided Segmentation," SRI Technical Note, AI Center, Stanford Research Institute, March 1976.

[6] D. Marr, "Early Processing of Visual Information," AI Memo 340, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, 1975.

[7] E.C. Freuder, "Affinity: A Relative Approach to Region Finding," Computer Graphics and Image Processing , 5, 254-264, 1976.

[8] A Hanson and E. Riseman, "Preprocessing Cones: A Computational Structure for Scene Analysis," Computer and Information Science Department Technical Report 74C-7, University of Massachusetts, September 1974.

[9] M.D. Kelley, "Edge Detection in Pictures by Computer Using Planning," Machine Intelligence 6, 1971.

[10] K. Price, "Change Detection and Analysis in Multispectral Images," Ph.D. Thesis, Carnegie-Mellon University, Pittsburgh, Pa., December 1976.

[11] R. Haralick, K Shanmugan and I. Dinstein, "Textured Features for Image Classification," IEEE Trans. on Systems, Man, and Cybernetics, SMC-4, September 1974.

[12] A. Rosenfeld and E.B. Troy, "Visual Texture Analysis," IEEE Conference on Feature Extraction and Selection in Pattern Recognition, 1970.

[13] R. Uhlander, "Analysis of Natural Scenes," Ph.D. Thesis, Carnegie-Mellon University, Pittsburgh, Pa., April 1975.

[14] A. Hanson, E. Riseman and P. Nagin, "Region Growing in Textured Outdoor Scenes," Computer and Information Science Department Technical Report 75C-3, University of Massachusetts, February 1975.

[15] A. Rosenfeld and A.C. Kak, Digital Picture Processing, Academic Press, 1976.

[16] R.A. Kirsch, "Computer Determination of the Constituent Structure of Biological Images," Computers and Biomedical Research, 4, 1971.

[17] J. Prager, "Extracting and Labelling Boundary Segments in Nature Scenes," Computer and Information Science Department Technical Report 77-7, University of Massachusetts, May 1977.

[18] Tom Williams, personal communication.

[19] A. Rosenfeld, Technical Report, Computer Science Center, 1977.

[20] National Bureau of Standards, NTSC Standard.

[21] J. Kender, "Saturation, Hue, and Normalized Color; Calculation, Digitization Effects and Use," Carnegie-Mellon University, Pittsburgh, Pa., November 1976.

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER COINS TR 77-8	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) REGION EXTRACTION AND DESCRIPTION THROUGH PLANNING		5. TYPE OF REPORT & PERIOD COVERED INTERIM
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Paul A. Nagin Allen R. Hanson Edward M. Riseman		8. CONTRACT OR GRANT NUMBER(s) ONR N00014-75-C-0459 NSF DCR75-16098
9. PERFORMING ORGANIZATION NAME AND ADDRESS Computer and Information Science University of Massachusetts Amherst, Massachusetts 01003		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research Arlington, Virginia 22217		12. REPORT DATE 5/77
		13. NUMBER OF PAGES 52
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this document is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Scene analysis, hierarchical segmentation, planning, feature extraction, texture analysis, preprocessing cone, histogramming, region growing, spatial adjacency matrix

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This paper examines several image segmentation algorithms which have been explored in the development of the VISIONS system. Each of these algorithms can be viewed as a variation on a basic theme: the clustering of activity in feature space via histogram analysis, mapping these clusters back onto the image, and then isolating regions by analysis of the spatial relationships of the cluster labels. It is shown that the interaction between these two representations of data (global feature

20. Abstract, continued.

information and spatial information) provides a view that is lacking in either.

The scene segmentation algorithms contain the following stages:

(1) PLAN: reduce the amount of detail in the scene to a bare minimum by performing a fast simple segmentation into primary areas using spatial and/or quantization compression.

(2) REFINE: resegment the scene with careful attention directed to the textural complexities of each region.

The primitive transformations which are used include histogram clustering, region growing, data reduction by narrowing the quantization range, and/or data reduction by spatially collapsing the data while extracting features. These algorithms have been implemented using a parallel, hierarchical computational structure. Comparisons of performance on several images are given.