Using Pyramids to Define Local Thresholds for Blob Detection

M. SHNEIER

***Abstract*–A method of detecting blobs in images is described. The method involves building a succession of lower resolution images and looking for spots in these images. A spot in a low resolution image corresponds to a distinguished compact region in a known position in the original image. Further, it is possible to calculate thresholds in the low resolution image, using very simple methods, and to apply those thresholds to the region of the original image corresponding to the spot. Examples are shown in which variations of the technique are applied to several images.**

***Index Terms*–Image processing, pattern recognition, segmentation, spot detection, thresholding.**

I. Introduction

The most common way to extract objects from a picture is to threshold the picture. Many different techniques have been used to select good thresholds for this purpose [10]. Threshold selection involves choosing a gray level t such that all gray levels greater than t are mapped into the "object" label, while all other gray levels are mapped into the "background" label. In its simplest form, a single threshold is chosen for the whole image. This does not usually give good results because of variations in lighting, or because there are several objects in the picture with different gray level characteristics. For better results, several local thresholds can be extracted from various parts of the picture, and can be applied only in those regions.

This paper describes a method of identifying parts of a picture on which to apply a threshold, and a means of calculating a local threshold for each of these parts. The method involves constructing a "pyramid" of images, each of lower resolution than its predecessor [2], [8], [9]. At some level of the pyramid, it is to be expected that any blob-like object should become spot-like. Thus, by running a spot detector over the low resolution images, the interesting regions in the picture can be discovered, and only those regions need be thresholded. In addition, the characteristics of the local regions (or the spots) can be used to calculate a good local threshold. The relationship of the current work to earlier work, such as that of Kelly [3], is discussed in Section IV. Examples are given of the application of the method to several images. In all cases the results are quite good, and highlight the usefulness of the method.

II. The Algorithm

The algorithm has two main tasks. The first is to find parts of the picture that differ significantly from the background (likely objects), while the second is to calculate a local threshold for each of these parts and apply it in the neighborhood of the parts. Both tasks make use of the pyramid of low resolution images.

1) If the whole pyramid has been constructed, stop. Otherwise, read in the previous pyramid level (the picture if this is the first iteration).

2) Build a new level (see below).

3) Apply a spot detector to the new level.

4) Evaluate the spots resulting from step 3 and find "good" spots (see below). If there are too many good spots, go to step 1.

5) For each good spot,

a) calculate a threshold (see below);

b) apply the threshold to the region in the original picture corresponding to the spot and write the results to the output picture.

6) Go to step 1.

The original image forms the base of the pyramid. Each level is constructed on top of its predecessor, and is processed before its successor is constructed. This means that only one level need be maintained at any time, in addition to the original picture and the partially constructed thresholded picture.

There are a number of different ways of constructing pyramids, using neighborhoods of various sizes, and having either disjoint or overlapping neighborhoods. Various nonoverlapped 2×2 operators were implemented, as well as overlapped 4×4 and 5×5 operators. In all cases, the rate of tapering of the pyramid was held constant, so that each level was one quarter the area of its predecessor.

The 2×2 operators included an average of the four points, their median, and their maximum. The 4×4 operator found the median of the 16 points in its neighborhood, and used that as the value for the next level, while the 5×5 operator used weighted averaging to approximate a Gaussian distribution centered on the midpoint of the neighborhood.

In addition to varying the size of the pyramid operator, the size of the spot detector was also subject to change. Two detectors were used, one of size 3×3, and one of size 5×5. Both were Laplacian operators, looking for a central peak surrounded by a valley. The sensitivity of the operators could be controlled by thresholding the relative heights of the peaks and valleys. For the 3×3 operator the threshold was fixed, while for the 5×5 operator it was a function of the mean and maximum gray levels in the image.

The blob detection process is invoked with parameters specifying the kind of pyramid operator and the spot detector to be used. The same operators are used at all levels of the pyramid, even though it could be argued that small operators should be used at low levels where there is still a lot of detail, while larger operators should be used higher up, where averaging has merged the objects and the background.

The pyramid operator is applied to the current level, producing the next level which is one quarter the area of its predecessor. The chosen spot detector is applied at each point of the new level, looking for points that differ from their neighbors, and ranking them according to how much they differ. The result of running the spot detector is a new image with high values where there are spots, and low values elsewhere.

The spot detectors are very conservative, so another process is run to find a subset of "good" spots. Good spots are spots that are isolated. At low levels of the pyramid (high resolution), spots that are close together are deleted because they can be expected to merge into single spots higher up in the pyramid. At higher levels of the pyramid, this is not such a good idea because single points represent large regions in the original picture. Thus, the definition of "good" is weighted by the level in the pyramid. A spot is good if the number of its neighbors that also responded positively to the spot detector is less than a level-dependent threshold.

Manuscript received December 6, 1979; revised August 12, 1982. This work was supported by the Defense Advanced Research Projects Agency and the U.S. Army Night Vision Laboratory under Contract DAAG-53-76C-0138, DARPA Order 3206.

The author was with the Computer Science Center, University of Maryland, College Park, MD 20742. He is now with the U.S. Department of Commerce, National Bureau of Standards, Washington, DC 20234.

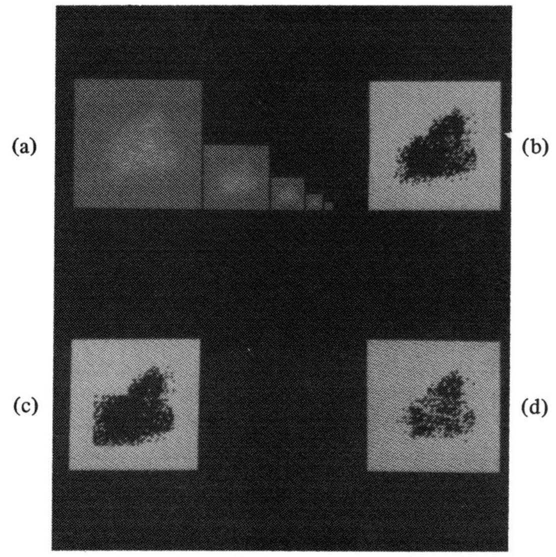

Fig. 1. (a) A FLIR image of a tank, and the pyramid constructed from it. (b) Thresholded image using the average of center and surrounding spots. (c) Thresholded image using surrounding spots only. (d) Thresholded image using center spot only.

Each spot in the low resolution image corresponds to a region in the picture. If there are too many spots, then large parts of the picture will be covered. If there is indeed an object in the picture, it should coalesce into a smaller number of spots higher up in the pyramid. If there is no object, then all the spots represent noise. In either case, the picture is too busy. A maximum number of good spots is allowed at each level. If this number is exceeded, no further processing is performed, and a new pyramid level is constructed.

When a small enough number of good spots is discovered at a given level in the pyramid, the thresholding can be performed. Notice that it need only be applied to the regions in the picture corresponding to the spots in the pyramid. All the other regions are ignored.

Many threshold selection techniques are applicable at this stage. There are the standard techniques [10] which may be applied to the picture itself in the region corresponding to a spot. In addition, it is possible to make use of the information in the low resolution image to calculate a threshold. Both approaches were followed for the examples to be discussed here. Using the low resolution image has the advantage that simple operations on the low resolution image correspond to complex operations involving much larger numbers of points in the picture.

The simplest threshold that can be extracted from the low resolution image is simply the gray level of the spot. This threshold is equivalent to the average gray level of the region in the picture corresponding to the spot. Usually, this threshold does not extract the whole object because the object gray levels bias the threshold, and there are very few nonobject points in the region to provide an opposite bias [Fig. 1(d)].

An alternative threshold is obtained by ignoring the spot itself, and averaging the surrounding points in the low resolution picture. This suffers from the opposite problem. Now, too many nonobject points reduce the threshold, and so parts of the background are classified as belonging to the object [Fig. 1(c)].

A compromise between the two methods gives very good results. The outputs from the above two threshold selection processes are averaged, and the result is used as the threshold [Fig. 1(b)].

The threshold is applied to a region slightly larger than that corresponding to the spot. This is to ensure that parts of the object that were averaged into different points in the low resolution image still may be classified, provided that they are not too far away from the spot center. If, indeed, the object extends a significant distance from the spot center, the spot detector should have found several spots in the neighborhood, each of which would be processed separately (or they would all be merged into a larger spot at the next level).

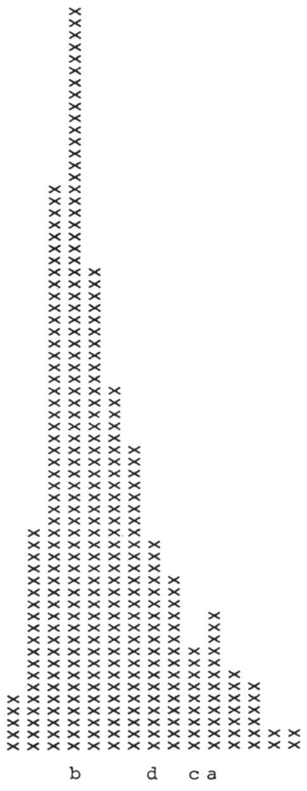

Fig. 2. An example of a histogram used for threshold selection. Point *a* is the initial point chosen for thresholding (see text). Point *b* is the highest peak in the direction of the background. Point *c* is the point chosen as the final threshold. Point *d* is the threshold chosen by the method of averaging the center and background points in the low-resolution image. The histogram is for a spot in the bottom left picture of Fig. 5. The small size of the spot results in a very low peak in the histogram (at *a*).

Another method of calculating a local threshold was also implemented. The method involves computing a histogram of the gray levels in the region of the original picture that corresponds to each spot. For each spot, a histogram is constructed for a region slightly larger than the projection of the spot onto the picture. The histogram is then examined, and a threshold is selected. The process of selection is complicated by the shape of the histogram, which tends either to be unimodal, or to have no significant peaks (Fig. 2). The method that was used to find a threshold involves making an initial estimate, and refining the estimate on the basis of the shape of a part of the histogram.

The initial guess that was used was one of the naive thresholds mentioned above. The gray level corresponding to the spot in the pyramid provides an estimate of the gray level in the center of the object. Usually, the estimate needs to be modified to take account of parts of the object close to the background. To accomplish this, the histogram is examined, starting at the initial estimate, and moving in the direction of the background gray levels. The highest peak in the histogram in this direction is discovered, and the final threshold is chosen at the deepest valley between this peak and the initial estimate. This usually results in a good threshold, in most

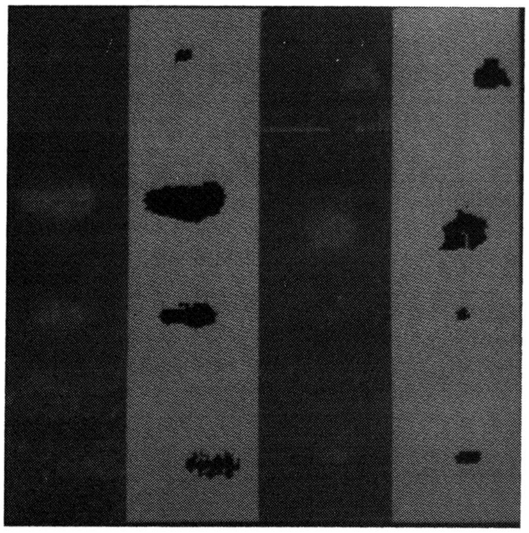
Fig. 3. Eight FLIR images and their threshold outputs. The pyramids were built by averaging in these examples and the threshold was selected as the average of the center and surrounding points in the low resolution image.

cases in one very similar to the averaging of the center and surround points in the pyramid discussed earlier (Fig. 2).

The output picture is initially blank. The only regions of the picture that are changed are those that correspond to positive responses to the spot detector at some level in the pyramid. As a result, very little background noise appears in the output.

III. Examples

The method was applied to 196 FLIR images and to a picture of part of a handwritten signature. Some of the results are shown in Figs. 3–6. The examples are divided into three categories.

The first set of pictures (Fig. 3) was processed using a simple averaging scheme for building the pyramids. The threshold was selected from the low resolution image by taking the average of the center (spot) gray level, and the average surrounding level.

Sometimes, when the contrast between the object and the background is small, the averaging process may cause the object to merge into the background. For FLIR imagery, it was found that it is often better to use the median instead of the average in building the pyramids, and to use larger neighborhoods. Fig. 4 shows a set of examples where this was done. The threshold selection used the same method as for Fig. 3. In fact, for most of the 196 FLIR images, the 5×5 Gaussian pyramid operator, together with the 5×5 spot detector, produced better results than the other methods. It appeared that using larger regions both for building the pyramids and for detecting spots gave better results than using smaller neighborhoods. Thus, the 4×4 pyramid operator was better than the 2×2 operators, but not quite as good as the 5×5 operator.

The alternative method of selecting a threshold by examining the histogram is illustrated in Figs. 5 and 6. Fig. 5 shows four FLIR images and the results of thresholding them. The pyramids for these images were constructed by averaging, and the thresholds were selected by examining a histogram of a region in the image slightly larger than that corresponding to the spot.

Fig. 6 illustrates the difference between selecting the threshold using only the low resolution image, and making use of the histogram as well. For the signature in Fig. 6, the histogram method results in a much cleaner thresholded image.

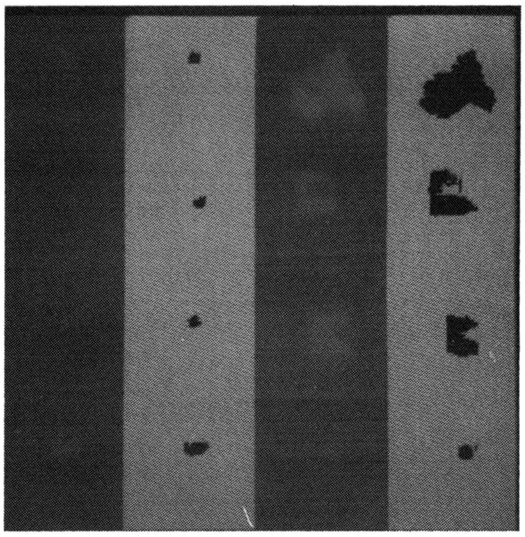
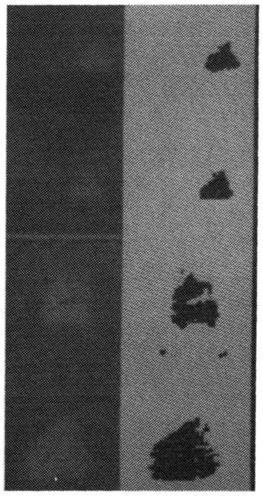
Fig. 4. Twelve FLIR images and their thresholded outputs. The pyramid was built using the median and the threshold was selected as the average of the center and surrounding points in the low-resolution image.

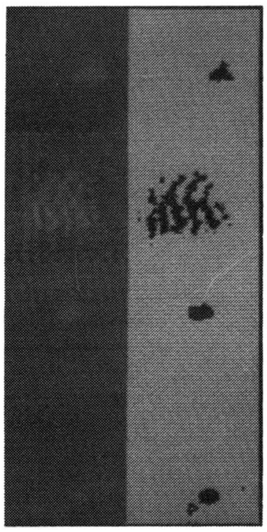
Fig. 5. Four FLIR images and their thresholded outputs. The pyramids in these examples were constructed by averaging, and the threshold was selected by examining the histograms of local regions corresponding to spots.

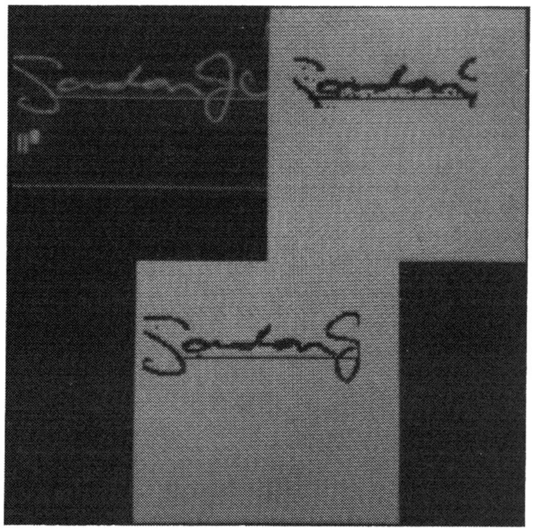

Fig. 6. (a) A picture of part of a handwritten signature. (b) The threshold output using the average of the center and surrounding low-resolution points. (c) The result of calculating a threshold by examining the histogram.

IV. Discussion

The blob-detection system described here provides a good threshold-detection technique, with the ability to isolate significant regions of the picture based on size and compactness. This results in cleaner thresholded images because regions are only thresholded if they meet the criteria imposed by the spot detector. Also, because local image statistics are used to calculate the threshold value, a good segmentation can be obtained, regardless of conditions elsewhere in the image.

While thresholding the image produces good results in many cases, it is rather a crude way of extracting the objects of interest. Given that the locations of the objects are known, some more sophisticated techniques could be applied in those regions. For instance, the Superslice algorithm (Milgram [5]) could be used to provide edge-region coincidence data, or the powerful pyramid linking processes of Burt *et al.* [1], could be used to extract the fine object shape, in a manner consistent with the hierarchical blob detection system.

Other questions arise concerning the operators used to construct the pyramid, and to search for objects with specific shapes. For images in which the objects are fairly distinct from the local background, the nonoverlapped pyramid operators and the 3×3 spot-detector were powerful enough to produce good segmentations. It usually made little difference which of the max, mean, or median pyramid operators was used, although over all the images, the median operator came out best.

As soon as the local contrast deteriorates, however, the larger overlapped pyramid and shape detection operators become necessary. Because they have a larger neighborhood over which to make their calculations, they tend to detect finer variations. These can be stored at the next level in the pyramid by the construction operators, or used to extract objects that contrast only slightly with their backgrounds. Another advantage of the larger operators is that nothing "falls through the cracks" when the pyramid is constructed, because of the overlapped neighborhoods. Of course, larger operators are more expensive to compute, although the extra operations can be minimized by applying box filtering techniques (McDonnell [4]). The extra sensitivity of the larger operators is a significant advantage in the low contrast typical of FLIR imagery. The Laplacian spot detectors used in these experiments can also be replaced by more sophisticated shape detectors, such as spoke filters (Minor and Sklansky [6]).

One of the features of this method is that it makes it possible to extract directly only those objects having a specified shape. In conventional thresholding schemes, the shape of an object can only be discovered after the object has been extracted, resulting in costly postprocessing. Some experiments were carried out using detectors that were sensitive to elongated objects (Shneier [7]). To extract elongated objects, a line or streak detector was applied instead of the spot detector. It was found that linear features such as roads, rivers, and runways, could be extracted using an extension of the thresholding techniques described here. This leads to the possibility of looking for vehicles on roads, for instance, by first finding the roads using the linear feature detectors, and then using the spot detectors only within those regions to find the vehicles.

While there are similarities between the work described here and that of Kelly [3], there are also fundamental differences. The most obvious difference between the two systems is that Kelly was finding edges whereas the current system is concerned with the region extraction. In addition, Kelly used only two images, the second of which was smaller than the first by an arbitrary amount that depended on the size of the objects being extracted. This meant that objects of only one size could be extracted, and that the size had to be prespecified. The current system can deal with objects of any size (as is apparent from the examples). It uses a hierarchy of images instead of only two, and is able to find the best size for extracting each object. A third difference concerns the way in which shape information is handled. The current system uses shape information in deciding which objects to consider significant (that is, in the feature detection itself). Kelly's system, however, first extracted edges in the smaller picture, and then had to decide which of those edges were best suited to describing the outline of the shape in the larger picture. It is this property of segmentation based on shape that is considered the major contribution of this paper.

V. Conclusions

A new method of detecting blobs in a picture by spot detection and local thresholding has been presented. The examples show how simple threshold-detection calculations on low resolution images can lead to a good segmentation of a picture based on object shape. It was found that using larger, overlapped operators for constructing pyramids and for feature detection resulted in improved performance over smaller operators, and that the results for a class of FLIR images were for the most part reasonably good.

References

[1] 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. 802-809, 1981.

[2] A. R. Hanson and E. M. Riseman, "Segmentation of natural scenes," in *Computer Vision Systems*, Hanson and Riseman, Eds. New York: Academic, 1978, pp. 129-163.

[3] M. D. Kelly, "Edge detection in pictures by computer using planning," in *Machine Intelligence VI*, B. Meltzer and D. Michie, Eds. Edinburgh: Edinburgh Univ. Press, 1971, pp. 397-409.

[4] M. J. McDonnell, "Box filtering techniques," *Comput. Graphics Image Processing*, vol. 17, pp. 65-70, 1981.

[5] D. L. Milgram, "Region extraction using convergent evidence," *Comput. Graphics Image Processing*, vol. 11, pp. 1-12, 1979.

[6] L. G. Minor and J. Sklansky, "The detection and segmentation of blobs in infrared images," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-11, 1981.

[7] M. Shneier, "Extracting linear features from images using pyramids," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-12, 1982.

[8] S. L. Tanimoto, "Regular hierarchical image and processing structures in machine vision," in *Computer Vision Systems*, Hanson and Riseman, Eds. New York: Academic, 1978, pp. 129-163.

[9] L. Uhr, "Recognition cones, and some test results: The imminent arrival of well-structured parallel-serial computers; positions, and positions on positions," in *Computer Vision Systems*, Hanson and Riseman, Eds. New York: Academic, 1978, pp. 363-368.

[10] J. S. Weszka, "A survey of threshold selection techniques," *Comput. Graphics Image Processing*, vol. 7, pp. 259-265, 1978.

An Iterative Approach to Region Growing Using Associative Memories

WESLEY E. SNYDER AND ALAN COWART

Abstract—The postulate is made that "any computation which can be performed recursively can be performed easily and efficiently by iteration coupled with association." The "easily and efficiently" part of that postulate is nontrivial to prove, and is shown by examples in this paper. The use of association leads directly to potential implementation by content-addressable memories.

The example addressed is region growing, often given as a classical example of the use of recursive control structures in image processing. Recursive control structures, however, are somewhat awkward to build in hardware, where the intent is to segment an image at raster scan rates. This paper describes an algorithm and hardware structure capable of performing region labeling iteratively at scan rates. Every pixel is individually labeled with an identifier signifying to which region it belongs.

The difficulties which often justify recursion ("U"- and "N"-shaped regions, etc.) are handled by maintaining an equivalence table in hardware, transparent to the computer, which reads the labeled pixels. The mechanism for updating the region map is explained in detail.

Furthermore, simulation of the associative memory has been demonstrated to be an effective implementation of region growing in a serial computer.

Index Terms—Associative memories, image processing, parallel processing, region growing, segmentation.

I. Introduction

In this paper we present an alternative algorithm to recursive region growing. This algorithm is functionally identical to traditional region growing in that it returns a set of labeled pixels, meeting the adjacency and similarity criteria. The usefulness of such labelings as part of segmentation systems is well established [1]-[3] and will not be discussed here.

While this algorithm is functionally identical to traditional region growing, it is fundamentally different in concept and potential implementation.

Our objective was to find a way to achieve the results of region growing, but to do so "on the fly" with a single pass over the data. We have achieved this result by using the concept of a content-addressable memory. This memory may be a physical piece of hardware or a lookup-table-driven access method in simulation software.

Since the VLSI revolution is spurring the development of special-purpose signal processors, it is hoped that this paper will inspire its readers to look at both traditional and newly conceived image analysis algorithms in a new way.

Manuscript received October 6, 1980; revised November 29, 1982. This work was supported primarily by NASA under Grant NSG 1535 and in part by DFVLR (Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt e. V.).

The authors are with the Department of Electrical Engineering, North Carolina State University, Raleigh, NC 27650.

II. The Traditional Approach to Region Growing

One central problem arising in scene analysis is the partitioning of an image into a set of meaningful regions. These regions are composed of all pixels that have similar attributes (such as gray level) and that are connected to each other. One of the most powerful approaches to this task is "region growing" [3]. This technique is initiated by choosing a pixel which meets the criteria for inclusion in a region. It then proceeds by examining all adjacent pixels meeting an acceptance criterion. This acceptance criterion is based upon the similarity between the pixel and the neighbor in question. Typical measures of similarity include the magnitude of the neighboring pixel's gray level or the relative contrast between the pixel and its neighbor under consideration for inclusion in the region. This process is repeated recursively for all newly accepted pixels until no new pixels can be added to the region.

The recursion may be accomplished by either writing a recursive program or an iterative program which explicitly uses a pushdown stack. The latter algorithm is detailed below.

1) Begin with the center pixel at (i, j).

2) Consider the pixel at $(i - 1, j)$. Is that pixel "similar" to the center pixel? If yes, push $(i - 1, j)$ onto the stack.

3) Repeat 2) for $(i, j - 1)$, $(i + 1, j)$, and $(i, j + 1)$.

4) Mark the center pixel so it will not be considered again.

5) Pop the stack to choose a new center pixel, thus redefining i and j.

6) If the stack is empty, stop. The set of marked pixels constitutes the region.

7) If the stack is not empty, go to 2).

After this process has run, a new start pixel will be chosen and a new region will be grown. When all pixels have been assigned to a region, the algorithm terminates.

Since the region-growing technique always results in closed regions, this technique is sometimes preferable to other techniques which are based on edge detection or line fitting. Numerous variations and applications of the basic region-growing technique have been proposed in the past [1], [2], [8], [11]. Although region growing has proved to be an integral part of scene analysis, its use can quickly become computationally prohibitive particularly for high resolution images. This has prompted us to consider alternative methods of region partitioning.

III. An Algorithm for Region Partitioning

The algorithm for region partitioning presented in this section is a region-growing technique based on the concept of equivalence relationships between the pixels of an image. Our definition of equivalency is as follows. Two pixels a and b are defined to be equivalent [designated $R(a, b)$] if they belong to the same region of an image. This relationship can be shown to be reflexive $(R(a, a))$, symmetric $(R(a, b) \Rightarrow R(b, a))$, and transitive $(R(a, b)$ AND $R(b, c) \Rightarrow R(a, c))$.

The transitive property enables all pixels in a region to be determined by considering only local adjacency properties. In this algorithm, each pixel will be compared to each adjacent pixel in a left-to-right, top-to-bottom raster scan fashion. The assignment of a region label to a pixel results from this comparison operation. Fig. 1 demonstrates the situation than can arise as result of this comparison. Pixels in a simple binary image are being labeled in raster scan order. The region labeling pro-