A MULTIRESOLUTION APPROACH TO BINARY TREE REPRESENTATIONS OF IMAGES

Hayder Radha **, Riccardo Leonardi*, and Martin Vetterli***

* AT&T Bell Laboratories, Holmdel, New Jersey 07733

** Center for Telecom. Research and Dept. of Elec. Eng., Columbia University
New York, New York 10027

ABSTRACT

The Binary Space Partitioning (BSP) tree representation of images provides a good segmentation and a simple data structure (binary tree). Our BSP tree approach partitions the image by straight lines that pass through the boundaries of the objects in the scene. The result of this partitioning is a segmented image consisting of a set of convex regions where the pixel intensities are homogeneous within each region. In this paper we introduce a multiresolution method for constructing a BSP tree. This approach derives a hierarchy (pyramid) of scale-space images from the original image. In this hierarchy, a BSP tree of an image is built from other trees representing low resolution images of the pyramid. A low-resolution-image BSP tree serves as an initial guess to construct a higher-resolution-image tree. Due to filtering when constructing the pyramid, details are discarded. As a result, we obtain a more robust segmentation. Moreover a significant computational advantage is achieved.

1. INTRODUCTION

Binary Space Partitioning (BSP) trees have been used in computer graphics applications as an efficient representation of polyhedra in d-dimensional space [Thibault]. The BSP tree approach partitions the space (that surrounds the desired object to be presented) by hyperplanes (straight lines for 2-D space) passing through the boundary of the object. The result of this binary partitioning is a set of convex (unpartitioned) regions which are represented by the leaf nodes of the tree. The partitioning lines are assigned to the nonleaf nodes as shown in Figure 1.

Using straight lines to partition an image that consists of several objects (of unknown shapes and sizes) requires segmenting the image into these objects. Without performing the difficult segmentation process, one can base the partitioning on the contour (edge) information in the image. In recent work [Radha90] we have introduced a Hough transform-based method for generating BSP tree representations of images. Two steps were used in [Radha90] for building a BSP tree of an arbitrary image: (1) extract the boundary locations of the objects in the image through an edge detection process, and (2) determine the linear characteristics of these boundaries through a series of Hough transforms, in order to match them (the edge points) with a minimum set of straight lines. Our BSP tree generation method starts by applying a modified Hough transform (HT) scheme to all edge points in the image, and selects the HT cell with the maximum number of votes (corresponding to the line that passes through the maximum

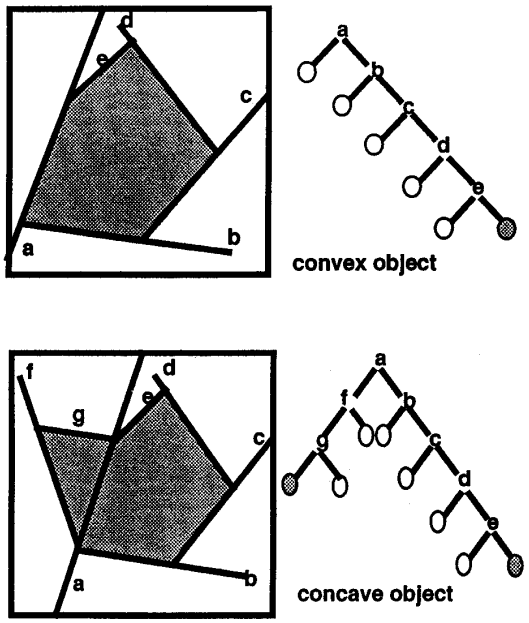


Figure 1. BSP tree representations of objects.

number of connected edge points). The resulting straight line partitions the image into two sub-images. The same process is repeated on each sub-image. In other words, the HT of one of the two sub-images (halfplanes) is computed, and a search for the HT cell with the maximum number of votes is performed. Now, the line corresponding to this cell will partition the selected sub-image. The same thing is done to the other sub-image. This procedure is repeated, recursively, till a termination criterion is reached.

This approach provides a good segmentation performance (as shown in Figure 2), and a simple data structure (binary tree). We have shown the potential of this representation for high compression image coding [Radha90]. However, this HT-based method is computationally intensive. A hierarchical implementation of the Hough transform can be a solution to this problem as suggested in [Wang]. Another solution is to use a *multiresolution* method when building the desired BSP tree. In this paper, we introduce a multiresolution-based, hierarchical method for generating the BSP tree from several scale-space images derived from an original image. As explained in the next

CH2977-7/91/0000-2653 $1.00 © 1991 IEEE

section, the multiresolution approach takes advantage of computations that take place at a coarse resolution level when performing similar computations at a finer (higher) resolution level. This new approach includes the introduction of a *line focusing* algorithm used to derive a tree representing an image (at a given resolution) from another tree representing a coarser image. The computational gain is shown in Section 3. Section 4 presents some simulation results while Section 5 concludes this work.

2. THE MULTIRESOLUTION APPROACH

The multiresolution approach derives a hierarchy (pyramid) of signals from the original signal [Mallat] as shown in Figure 3. We assign negative numbers to the coarse resolution levels since the highest resolution (original) signal starts at level zero.

A derived signal at a given resolution level L in the hierarchy contains the (original) signal's information of that level L and all other resolutions *lower (or coarser)* than L. For example, the image at level -1 in Figure 3 contains the information of levels -1, -2, and -3. However, it does not contain the higher resolution (level 0) information. One can (for example) detect the boundaries of large objects (coarse details) found in a scene by performing an edge detection process on a low resolution image of that scene. This low resolution boundary information can be precisely refined using a high resolution image of the same scene [Bergholm]. This procedure is more computationally efficient.

A similar multiresolution approach can be used to extract a BSP tree representation of an image. We first derive a hierarchy of images from the original image. Then, we build the BSP tree of the lowest (most coarse) resolution image in the pyramid using the HT-based method (described in [Radha90]). We use this low resolution tree as an initial guess to build another tree representing the next higher resolution image in the hierarchy.

Figure 2. BSP tree representation of an image. a b c d

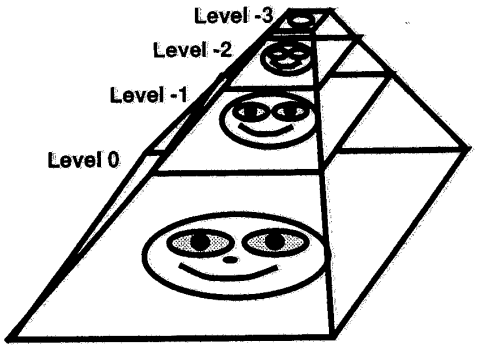


Figure 3. A multiresolution pyramid

The same process (applied from one level to the next) is repeated until we generate the BSP tree that represents the highest resolution (original) image. From here on, we consider only the two top levels of the pyramid.

Throughout this document we use the (θ, ρ) representation of a straight line h, where θ is the direction (with respect to the x axis) of the vector normal to h, and ρ is the distance between the origin and h.

2.1 From a Coarse Tree to a Finer Tree

Let I_0 be the original image, and E_0 be the edge image of I_0. From I_0, we derive the image I_m by (1) convolving I_0 with a 2-D Gaussian filter $g_{\sigma_m}(x, y)$ of variance σ_m^2, and (2) subsampling the filtered image using a decimation factor $M = 2^{-m}$ in both the x and y directions, where $m \leq 0$ is the resolution level. An edge image E_m is then derived from I_m using a maximum-gradient edge detection process. By employing the HT-based method explained in [Radha90], a BSP tree T_m is generated from the edge image E_m. Every leaf node in T_m represents an unpartitioned, convex region of I_m. Every nonleaf node μ_m, which represents a convex region in I_m, is associated with a line (θ_m, ρ_m) that partitions the region.

In the previous work [Radha90], we derived a BSP tree T_0' (which represents the original image I_0) from the edge image E_0 only. It should be noted that E_0 conceptually contains the boundary information of I_0 at all resolution levels $L \leq 0$. The previous approach, however, is computationally intensive. The objective of this work is to reduce the computational burden of the Hough transform-based method by deriving a BSP tree T_0 (which represents the original image I_0) from:

1. the tree T_m which represents the image I_m, and contains the boundary information of the original image I_0 at all resolution levels $L \leq m$;
2. the boundary information of the resolution levels $m < L \leq 0$ contained within the edge image E_0.

It is important to note that both T_0 and T_0' represents I_0 at all levels $L \leq 0$. In general, however, $T_0' \neq T_0$. The quality of the two images reconstructed from both trees are comparable.

We derive T_0 in two stages: (1) a BSP tree T_{m0} is derived using a preorder tree traversal of T_m and a line focusing algorithm (explained below), and (2) T_0 is then generated by replacing the leaf nodes of T_{m0} with new BSP sub-trees which represent the additional boundary information at the resolutions $m < L \leq 0$.

It should be clear that T_{m0} has the same shape, structure, and number of nodes as T_m. Moreover, and under ideal conditions, a line h_m associated with a node μ_m in T_m will have the same parametric representation (θ_m, ρ_m) as the line h_0 associated with the corresponding node μ_{m0} (which is derived from the node μ_m). Due to filtering, however, the edges of I_0 shifts from their original positions. It was shown [Bergholm] that this shift is proportional to the scale-space parameter σ_m. We used this result to design a line focusing algorithm [Radha91].

Due to lack of space, we only state the main result of this algorithm. Given a line $h_m = (\theta_m, \rho_m)$ detected at level m of a multiresolution hierarchy, the corresponding line $h_0 = (\theta_0, \rho_0)$ (at the higher resolution 0) can be detected by performing the Hough transform on all edge points (x_e, y_e) that satisfy the following condition:

$$\rho_m - \Delta\rho_{max} \leq \rho \leq \rho_m + \Delta\rho_{max}, \qquad (2.1)$$

where $\rho = x_e\cos(\theta_m) + y_e\sin(\theta_m)$, and $\Delta\rho_{max} = 2\sigma_m(\cos\theta_m + \sin\theta_m)$.

It should be clear that the smaller $\Delta\rho_{max}$ is, the less computation is required to detect the line h_0. Consequently, given $h_m = (\theta_m, \rho_m)$, one can reduce $\Delta\rho_{max}$ by selecting smaller values for σ_m. However, using small σ_m increases the amount of aliasing energy in the decimated image I_m. This represents a tradeoff between the amount of: (1) computation required to detect h_0, and (2) aliasing energy one can tolerate in the decimated image. In [Radha91] we derive an expression for $\Delta\rho_{max}$ in terms of a measure of the Gaussian filter's aliasing energy. Using this expression, one can quantify the relationship between the minimum computation area and the amount of aliasing.

3. COMPUTATIONAL ADVANTAGE

In this section we derive an expression for the computational advantage of using the multiresolution approach (i.e., building T_0 from T_m and E_0) versus applying the HT-based method directly on E_0 (i.e., building T_0'). First we develop a model for the computational complexity of building a binary tree T from an edge image E.

The time required to build T from E depends mainly on (1) the number of edge points P_E in E, and (2) the topology (i.e., the tree structure) of T [Radha91]. Due to the large number of possible topologies of a binary tree T with n ($>>1$) nodes, here we only consider two extreme cases of binary-tree structures: (a) the complete (fully symmetric), and (b) fully asymmetric topologies. Figure 4 shows the fully symmetric and asymmetric binary trees for $n = 15$. These two cases are representative of the lower and upper bounds for the computational complexity of building T, respectively. We measure the computational complexity of both cases as the *total* number of edge points P_T needed to be processed in order to build the desired tree T. One can estimate the time t_T required to build a BSP tree T of an image as: $t_T = t_{ht}\, P_T$, where t_{ht} is the time required to compute the HT of an edge point.

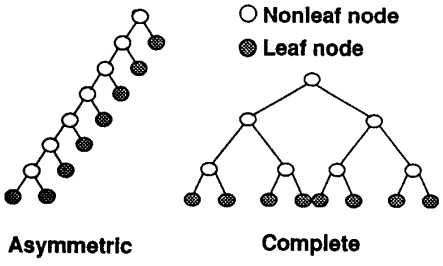


Figure 4. Asymmetric and complete binary trees.

Here we denote N to the number of partitioning lines, and $\bar{p}$ to the average number of edge pixels that lie on a line h (detected from E). Therefore, $P_E = N\,\bar{p}$.

A *complete binary* tree T_c of depth d has all of its leaves at level d [Tenenbaum]. The total number of nonleaf nodes in T_c is $2^d - 1$. Therefore, (in our case) the number of straight lines $N = 2^d - 1$, or $d = \log_2(N+1)$. After detecting the root-node line (which requires processing all P_E edge points in E), one has to process (on the average) $P_E - \bar{p}$ edge points to detect the two lines (at level one of T) that are associated with the right and left children of the root node. Similarly, it can be shown [Radha91] that $P_E - (2^j - 1)\,\bar{p}$ edge points have to be processed to detect the partitioning lines at level j of a complete binary tree.

Therefore, the computational complexity of generating T_c can be expressed as follows:

$$P_T = \sum_{j=0}^{d-1} [\, P_E - (2^j - 1)\,\bar{p}\,] \qquad (3.1)$$

A *fully asymmetric* binary tree T_a of depth d has d ($= N$) nonleaf nodes. There is one nonleaf node for each level between zero (the root node) and level $d-1$. The nonleaf node at level $d-1$ has two leaf children, whereas each of the remaining $d-1$ nonleaf nodes has one leaf child and one nonleaf child. In order to detect the straight line associated with the nonleaf node at level j of a fully asymmetric binary tree T_a, it is required (on the average) to process $P_E - j\,\bar{p}$ edge points. Therefore, the computational complexity of generating T_a can be expressed as follows:

$$P_T = \sum_{j=0}^{d-1} [\, P_E - j\,\bar{p}\,] \qquad (3.2)$$

It can be shown that the time t_T required to generate a BSP tree T from an edge image E satisfies following inequality:

$$[\, d\,(N+1) - N\,] \leq \frac{t_T}{t_{ht}\,\bar{p}} \leq [\, N\,(N+1)/2\,] \qquad (3.3)$$

We define the computational advantage ratio as: $R_{CA} = t_{T_0}' / t_{T_0}$, where t_{T_0} and t_{T_0}' are the times required to build T_0 (from T_m and E_0) and T_0', respectively.

To simplify the derivation we assume that every straight line h_0 detected from E_0 (when building T_0'), has a corresponding line h_m in T_m. In other words, E_0 contains boundary information at resolution levels $L \leq m$ only. Therefore, the number of lines N_m detected from the edge image E_m is equal to the number of lines detected from E_0.

Using simplifying assumptions, it can be shown [Radha91] that:

$$\frac{M^2 \log_2 [\,(N_m + 1)/2\,]}{M^2 + \log_2 [\,(N_m + 1)/2\,]} \le R_{CA} \le \frac{M^2 [\,(N_m + 1)/2\,]}{M^2 + [\,(N_m + 1)/2\,]} \quad (3.4)$$

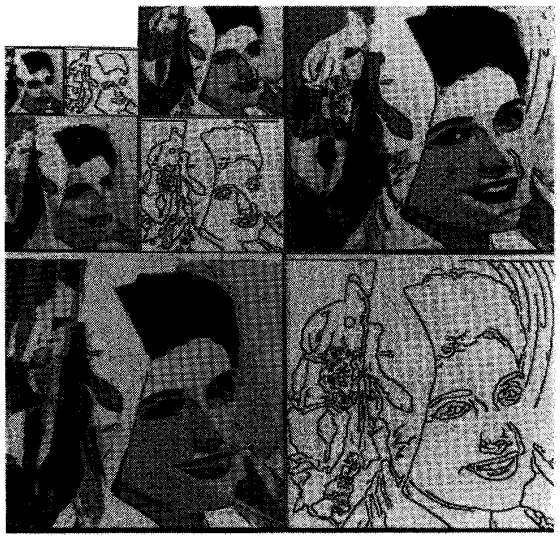
Since M^2 and $\log_2 [\,(N_m + 1)/2\,]$ are always larger than one and two, respectively, the lower bound of R_{CA} are *always* larger than one. Therefore, using the multiresolution approach will always provide a computational advantage versus the direct application of the HT-based method. Moreover, the upper bound of R_{CA} converges to M^2 when $(N_m + 1)/2 >> M^2$ which is the case for most images.

4. SIMULATION RESULTS

The multiresolution approach described above was tested on the 256x256, gray-scale image shown in Figure 2a. This image represents, in our case, the resolution level $m = 0$. Two scale-space images (I_{-1} and I_{-2}) at resolutions $m = -1$ ($M = 2$) and $m = -2$ ($M = 4$) were derived using $\sigma_m = 1$. Three edge images E_0 (Figure 5g), E_{-1} (Figure 5d), and E_{-2} (Figure 5b) were derived from I_0, I_{-1}, and I_{-2}, respectively, using a maximum-gradient edge detector.

In this section, we show the results as mean-value images, where each unpartitioned region (which represents a leaf node of the tree) has been filled with the average value of the pixel intensities within that region. We refer to these images as the BSP-tree images.

First the HT-based method was applied on the edge image E_{-2} in Figure 5b (of size 64x64). The resulting BSP-tree image is shown in Figure 5a.



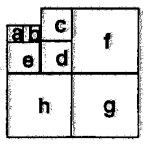


Figure 5. BSP tree and edge images.

The corresponding tree T_{-2} and the edge image E_{-1} were used to derive the BSP-tree image shown in Figure 5e. By traversing T_{-2} and using the line focusing algorithm, the line equations at the resolution level $m = -1$ were determined. Therefore, Figure 5e shows the details of resolution level $m = -2$ displayed at resolution $m = -1$. Figure 5c shows the result of expanding the leaves of T_{-2} using the boundary information of resolution $m = -1$ found in the edge image E_{-1}.

Using the BSP tree T_{-1} (whose image is shown in Figure 5c), we repeat the same process: (1) derive the BSP-tree image of Figure 5h which shows the details of resolution $m = -1$, and (2) expand the leaves of T_{-1} with new subtrees (representing the details at resolution $m = 0$) using boundary information within E_0 (Figure 5g). The result of step (2) is shown in Figure 5f.

By applying the HT-based method on E_0 directly, one gets the BSP-tree image shown in Figure 2d. By comparing this image with the BSP-tree image of Figure 5f, we can notice that a better segmentation can be achieved using the multiresolution approach. This is an added value to the main speed advantage one can gain when employing this new approach. For example, using a Sun 4 workstation, it takes about 50 minutes to generate the image of Figure 2d, whereas the corresponding image of Figure 4f was generated (using the multiresolution approach) in about 10 minutes on the same machine.

5. CONCLUSION

In this paper we have introduced a multiresolution approach for binary tree representation of images. We also introduced a line focusing algorithm used to combine the different trees representing several scale-space images. This new approach provided an improved performance in segmentation and speed as compared with a direct employment of the HT-based method proposed in our earlier work [Radha90].

References

[Bergholm] F. Bergholm, "Edge Focusing," *IEEE trans. Pattern Anal. Machine Intell.*, Vol. PAMI-9, pp. 726-741, Nov. 1987.

[Mallat] S. G. Mallat, "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation," *IEEE trans. Pattern Anal. Machine Intell.*, Vol. PAMI-11, pp. 674-693, July 1989.

[Radha90] H. Radha, R. Leonardi, B. Naylor, and M. Vetterli, "Image Representation Using Binary Space Partitioning (BSP) Trees," *Visual Communications and Image Processing V Proceedings*, October 1990, Vol. 1360, Part One, pp. 639-650.

[Radha91] H. Radha, R. Leonardi, B. Naylor, and M. Vetterli, "Binary Tree Representation of Images," AT&T Bell Laboratories Technical Memorandum, January 1991.

[Thibault] W. C. Thibault and B. F. Naylor, "Set Operations on Polyhedra Using Binary Space Partitioning Trees", *SIGGRAPH*, July 1987, pp. 153-162.

[Tenenbaum] A. M. Tenenbaum, Y. Langsam, and M. J. Augenstein, *Data Structures Using C*, Prentice Hall, 1990.

[Wang] Y. Wang, R. Leonardi, and S. K. Mitra, "The Connectivity Hough Transform and its Fast Implementation," *ICIP Proceedings*, September 1989, pp. 548-552.