REPORT NO. UIUCDCS-R-88-1409

# FAST TEXTURE RECOGNITION USING INFORMATION TREES

by

Darrell R. Hougen and Stephen M. Omohundro

February 1988

DEPARTMENT OF COMPUTER SCIENCE
1304 W. SPRINGFIELD AVENUE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
URBANA, Il 61801

# Fast Texture Recognition using Information Trees

**Darrell R. Hougen**
**Stephen M. Omohundro**
*Department of Computer Science and*
*Center for Complex Systems Research,*
*University of Illinois at Urbana-Champaign,*
*508 South Sixth Street, Champaign, IL 61820, USA.*

January 14, 1988

**Abstract.**

In this paper we approach the problems of texture recognition and discrimination using information trees, an information theoretic technique for efficient category learning in continuous domains. Texture recognition is important in the early stages of visual information processing because it provides a method for identifying materials from their visible surface properties. In addition, it allows images to be segmented into meaningful regions prior to object recognition.

Small, $n \times n$ patches are extracted from video images and used as training examples. Each patch is associated with a texture category by a teacher. Real valued features representing morphological properties of the patch are computed at a variety of scales and orientations. The computed values identify the patch with a point in a visual feature space. The most relevant dimensions for categorization are extracted by the information tree building algorithm. The feature space is recursively decomposed into hyper-rectangular regions representing textures classes. The resulting hierarchical decomposition is stored in a *k-d* tree—a binary tree containing both a discrimination value and dimension at each node. The splitting of a parent hyper-rectangle into its two children is determined by a universal information theoretic criterion. Information trees generalize by assigning a single category label to an entire leaf region.

## 1. Salience of Texture Recognition and Discrimination

Texture recognition and discrimination form one component of the complex task of processing visual data. For a variety of reasons we believe that it must occur relatively early in this process. Researchers on biological vision systems affirm this important role for textures [5]. The traditional approach to texture identification starts with an a *priori* model (such as Fourier descriptors or statistical moments) and attempts to show that it accounts for the desired discrimination abilities. We would like the system to learn which features are actually salient for the categorization task. Our system is one step in a larger program for developing a vision system which obtains its internal models by learning rather than by having them built in from the start. We hope to thereby achieve a system which is more robust, better tuned to its environment, and ultimately easier to build than that obtained by using the traditional approach. We have therefore tried to develop our algorithmic components so that they may be applied to a variety of visual problems.

There are two major classes of subtask involved in the overall task of visual processing: the extraction of continuous geometric and motion parameters and the assignment of objects to categories. This second task typically requires the labeling of distinct regions within the image. If the geometry and motion components of the scene can be inferred, they can be used to aid in the labeling of regions. Unfortunately, such geometric information is often difficult to obtain directly. It is also not required as is clear from the fact that people can interpret two-dimensional images such as paintings which lack such information. People are often able to partition an image into meaningful regions without using three dimensional geometry or motion information. Three dimensional objects take a variety of forms and orienta-

tions usually precluding any kind of direct template matching for recognition. Therefore one would like to decompose an image into potentially meaningful regions before knowing what those regions represent. Texture recognition is one technique for accomplishing this.

Particular materials, such as grass or asphalt, tend to give rise to relatively homogeneous patches of image. As such, they can be recognized from any sufficiently large subregion. The size of the region necessary for sufficiently accurate classification of a texture depends on the largest scale of significant variation. We would like to segment images into regions which correspond to objects composed of a single material. Because any textured region is as good as any other for identifying a material, texture is often useful for segmentation. The segmentation proceeds by first labeling small overlapping subregions of the image according to their texture and then clustering regions with the same label. We have implemented a texture classification system based on a data structure we call an "information tree" and have achieved a classification accuracy of up to 89% on 25 textures. We have used this module to build a rudimentary image segmenter and preliminary tests are quite promising.

## 2. Information Trees

Information trees map points in a real $k$-dimensional space into a set of categories in an empirically determined way. In addition, they allow for extremely efficient evaluation of this mapping. In the process of their construction they automatically identify relevant and irrelevant feature dimensions. Similar ideas may be used to efficiently learn and implement continuous mappings between two real spaces[6]. Similar algorithms have recently been examined

by a number of researchers, though mostly in discrete domains [1,2,4,7,8].

Information trees are designed to attack the general problem of learning an unknown function from a set of data points. Let $I$ be the input space and $O$ be the output. Let $F$ be a mapping from $I$ to $O$. Since $F$ is known only for a finite collection of points in $I$, the known part of $F$ may be represented as a set of input/output pairs. We would like to be able to evaluate $F$ for an arbitrary point $x$ in $I$. If $O$ is a continuous space, we would like to approximate the coordinates of the point $x$ in $O$ as closely as possible. In the case that $O$ is a discrete set, we must choose the correct element from $O$.

A classical approach to classification assigns a point to the category of its nearest neighbor in the set of samples. [3] showed that asymptotically, this approach has a probability of error which is less than twice that of any algorithm. In the continuous case, the $n$ nearest neighbors to $x$ can be used to estimate the output by interpolation. The $n$ nearest neighbors of a query point may be found quickly and efficiently if the points are stored in a *k-d* tree [4,6]; $k$ is the dimensionality of the input space. The *k-d* tree is a binary tree with both a discrimination value and dimension stored at each node. Nodes in the tree correspond to hyper-rectangular regions of the feature space. The root node represents the entire space. The set of all regions at a given level in the tree form a partition of the feature space. The two children of a parent node are formed by splitting the parent hyper-rectangle along one of its dimensions at a given value. The left child contains points for which the corresponding coordinate is less than this value, the right contains points for which it is greater.

A multi-dimensional tree of this type may be characterized by the method for choosing the splitting dimension and value at each stage. In the original

$k$-$d$ tree, the most spread out dimension was split and the splitting value was chosen to separate the parent sample points into two equal sets. Using a branch and bound technique, this structure allows the nearest neighbors of a point to be found in an expected time which is only logarithmic in the number of sample points. Unfortunately, if there are dimensions in the feature space which are irrelevant for the categorization task, an enormous number of samples may be required for adequate categorization.

We have therefore been using a variant structure which we call an information tree. The choice of which dimension to cut and where to cut it is based on an information theoretic criterion. If we knew the exact probability distributions over the input space for each category, we could precisely measure the amount of information about the output category contained in the knowledge that a point is contained in a given hyper-rectangle. According to Shannon, we merely sum the quantity $-p_i \log p_i$ over the output categories where $p_i$ is the probability that a point in the given hyper-rectangle is of category $i$. We may approximate this quantity by estimating the distribution using the empirical data. Let $N$ be the number of samples labeled by category numbers from 1 to $C$ in a given hyper-rectangle with $N_i$ samples from the $i$th category. The information may be estimated as

$$I = \sum_{i=1}^{C} \frac{N_i}{N} \log \frac{N_i}{N}.$$

Let $I$ be the category information of the parent node and $I_l$ and $I_r$ be the category information contained in its two children. If the left child contains $N_l$ points and the right child contains $N_r$ points, then the information about the output category which is gained by making a given cut is

$$\frac{N_l}{N} I_l + \frac{N_r}{N} I_r - I.$$

An information tree is built by choosing the cuts to maximize this quantity at every stage. This procedure always splits a hyper-rectangle on the most relevant dimension available. The tree building is terminated when the leaves are sufficiently homogeneous. We terminate splitting when the information gained in making a particular cut falls below some fraction of the information gained by splitting the root. This fraction is chosen experimentally.

Given a new point we may estimate its category extremely quickly using an information tree. We start at the root and proceed downward toward the leaves. We choose the branch which contains the given point by comparing its value along the discrimination dimension with the discrimination value stored at the node. If the tree is balanced, the whole process takes only a logarithmic number of comparisons.

## 3. Computing Features

Our implementation of information trees maps a vector of double precision numbers representing features into a set of categories represented by integers. The features are computed from small patches of an image. Therefore, each feature computing routine must take a small image patch as input and output a double precision number.

The set of numbers output by the feature computing routines defines all the knowledge that the information tree has available to it about an image patch. Therefore, a good set of feature computing routines must accurately represent all of the useful information contained in an image patch. Because the tree building procedure extracts only the most relevant features, we can begin with a wide variety of choices and the less useful ones will not be included in the final tree.

Most of the features used compared the average gray levels of adjacent regions in an image patch and measured the absolute amount by which they differed or the square of that difference. The regions were of varying sizes, orientations, and aspect ratios. The results over the entire image patch were combined by a number of techniques including averaging, taking the average of the top $n$ results, taking the maximum value over a subregion and then averaging over the entire patch and using the results to compute a variance. A total of 45 features were used in all. Different features were sensitive to different textures. Some of the textures were very accurately identified while others were not.

## 4. Building an Information Tree

An information tree is built by repeatedly choosing a dimension to cut along and a discrimination value. In a single stage, a $k$-dimensional region is split by a single $(k-1)$-dimensional hyperplane perpendicular to one of the axes. Fig. 1 is a tree diagram depicting a tree formed by choosing fifty points from each of five categories. The tree has been projected onto the 2-d plane defined by feature dimensions 0 and 1. In Fig. 2, the data have been cut along feature dimension 2. In this figure, the horizontal axis corresponds dimension 2 and the vertical axis to dimension 0 on both sides of the cut. In Fig. 3, the region to the left of the original cut has again been cut along dimension 2 while the area to right has been cut along dimension 34. The data to the left of the original cut have been plotted along dimension 2 in both the horizontal and vertical directions resulting in the apparent correlation between the points. The data to the right of the original cut have been plotted along dimension 2 in the horizontal direction and dimension 34 in the vertical direction. The

boxes have been independently scaled. We have found this representation technique to be quite useful for representing complex multi-dimensional data. In the final diagram, the upper right-hand box has been split twice more, first along dimension 17 and then along dimension 1.

Notice that the small box along the top edge of the plot contains points from more than one category and could be cut again. However, the box contains very few points. Since the information to be gained in cutting a box is proportional to the probability of landing in that box, it was not considered worthwhile to continue cutting. We have defined the minimum information gain necesssary to justify a cut as a fraction of the information gained in making the initial cut. In this case, the cutoff was set at 0.001.

## 5. Experimental Setup

We recently completed several experiments in which information trees were built to recognize 25 categories of textures. Samples patches were drawn at random from 25 350x350 pixel gray-level images of textures.

The purpose of the first experiment was to determine an appropriate information gain cutoff value. The experiment used trees built from 100 20x20 patches from each of the 25 texture categories. Cutoff values of 0.1, 0.05, 0.01, 0.002, and 0.001 were used. The results of this experiment indicated that cutoff values of 0.01 to 0.002 were appropriate. For values of the cutoff greater than 0.01, some of the categories did not recieve seperate leaf nodes resulting in misclassification of test patches from those categories. For lower cutoff values, the classification accuracy did not increase appreciably.

The second experiment forms a benchmark for the technique as it currently exists. Batches of 100 and 200 training patches from each texture

# Building an Information Tree

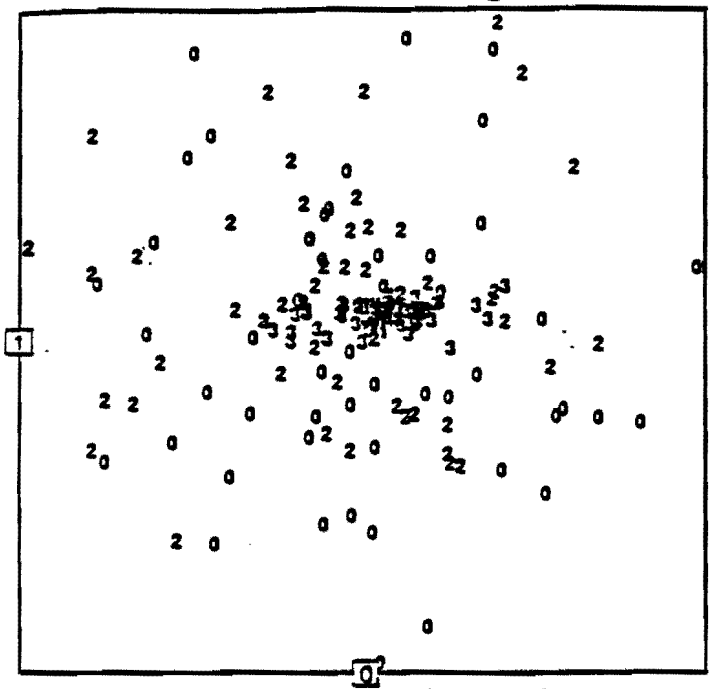

Figure 1: The uncut data plotted along dimensions 0 and 1.

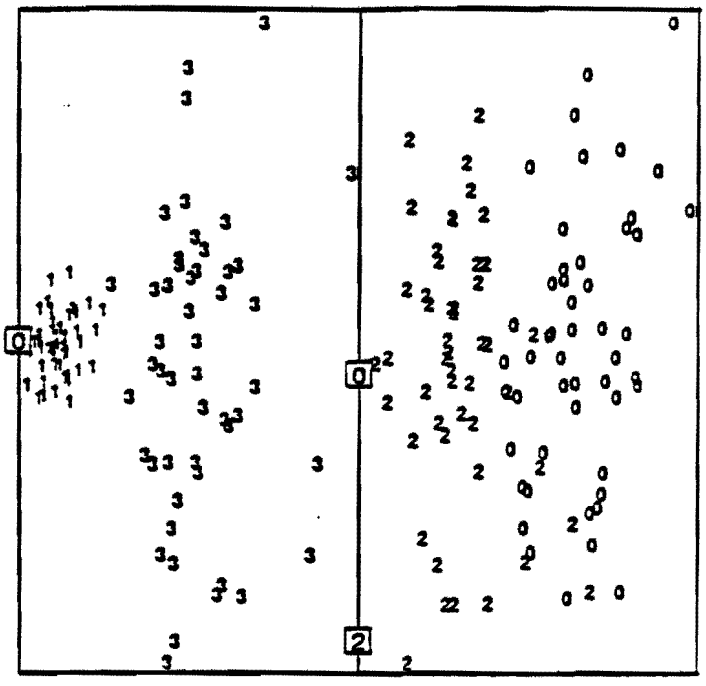

Figure 2: The same data cut along dimension 2.

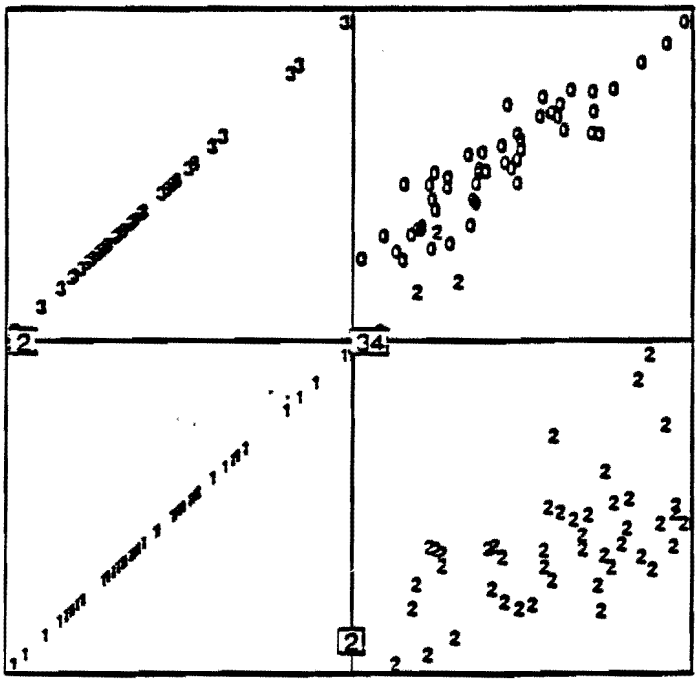

Figure 3: The left side cut along dimension 2 and the right side cut along dimension 34.

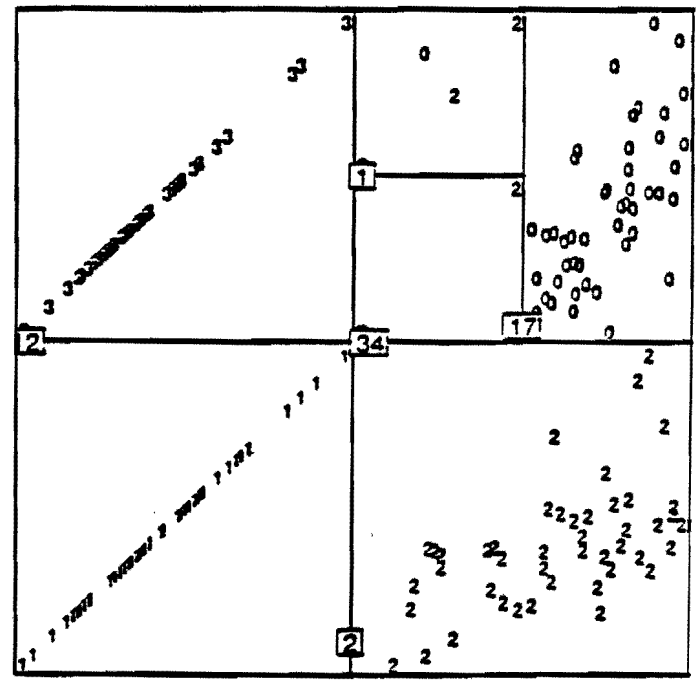

Figure 4: The final tree.

| *Examples* | *Cutoff* | *Region Size* | | | | |
|---|---|---|---|---|---|---|
| | | 20x20 | 25x25 | 30x30 | 35x35 | 40x40 |
| 100 | 0.01 | 71 | 77 | 81 | 85 | 85 |
| 100 | 0.002 | 72 | 78 | 82 | 87 | 87 |
| 200 | 0.01 | 72 | 78 | 83 | 85 | 87 |
| 200 | 0.002 | 73 | 83 | 87 | 88 | 89 |

Table 1: Overall Classification Accuracy.

were used at cutoff values of 0.01 and 0.002 for image patches ranging in size from 20x20 to 40x40 in increments of five. 100 test samples were drawn at random from the same images as the training samples. The test patches were the same size as the training patches.

## 6. Experimental Results

Table 1 depicts the overall classification accuracy of the information trees. Each tree is characterized by the number of examples drawn from each category, the information gain cutoff value and the patch size. It is easy to see that the overall classification rates are much higher than the 4% chance level performance.

The only pronounced effect is the increase in accuracy with patch size. This may be the result of several causes. Textures exhibit variation at a number of spatial scales. Beyond some scale, there is no significant change. This allows a textured area to be identified from any subregion of sufficient size. It may be that some of textures were not sufficiently represented by very small patches. Indeed, inspection of magnified patches by human observers showed that some of the smaller ones were hard to identify.

In addition, averaging the feature values over larger regions seems to result in values that lie in a narrower range. The computed values appeared to be more accurate. Spurious noise was averaged out. These effects could be seen in "cleaner" tree diagrams.

The effects of cutoff value and number of examples appear to be minimal. However, these numbers had been chosen by inspecting the results of preliminary experiments. The results of those experiments showed that a reduction in the number of examples or increase in the cutoff value resulted in significant performance degradation.

## 7. Differences Between Textures

A closer examination of the data showed that some categories were consistently identified more accurately than others. Table 2 depicts the classification accuracy for a set of trees built using 100 examples of each category and a cutoff value of 0.01. The test examples were drawn from the same images as the training examples. Also listed are the average performance, the number of categories that scored above 90% and the number of categories that scored above 80%.

It is clear that some of the categories were consistently identified very accurately. The big winner was an almost featureless bed sheet. However, the slightly more featured mailing envelope, two shirts and a piece of wood also consistently did very well. Their features were detected well enough to consistently discriminate them from each other and every other texture.

The losers are interesting for the ways in which they lost. For example, the system frequently confused grass samples with samples from the pile of long flat conifer needles (referred to as pine needles), a very human-like

| *Category* | *Region Size* | | | | |
|---|---|---|---|---|---|
| | 20x20 | 25x25 | 30x30 | 35x35 | 40x40 |
| Asphalt | 52 | 53 | 69 | 72 | 75 |
| Blanket 1 | 62 | 64 | 85 | 81 | 84 |
| Blanket 2 | 83 | 89 | 94 | 95 | 90 |
| Carpet 1 | 72 | 56 | 54 | 84 | 78 |
| Carpet 2 | 53 | 34 | 56 | 60 | 63 |
| Carpet 3 | 82 | 78 | 87 | 85 | 89 |
| Concrete | 63 | 69 | 85 | 72 | 89 |
| Curtain | 54 | 75 | 77 | 92 | 90 |
| Jeans | 88 | 88 | 85 | 94 | 93 |
| News Paper | 83 | 89 | 92 | 75 | 76 |
| Envelope | 93 | 100 | 100 | 100 | 100 |
| Fake Wood | 83 | 90 | 89 | 94 | 82 |
| Grass | 35 | 55 | 53 | 70 | 88 |
| Pine Needles | 31 | 46 | 41 | 66 | 42 |
| Sand | 52 | 59 | 55 | 68 | 69 |
| Bed Sheet | 100 | 100 | 100 | 100 | 100 |
| Shirt 1 | 98 | 99 | 99 | 96 | 96 |
| Shirt 2 | 99 | 100 | 98 | 99 | 99 |
| Shirt 3 | 52 | 59 | 76 | 73 | 69 |
| Shirt 4 | 73 | 86 | 83 | 88 | 95 |
| Sweater 1 | 90 | 85 | 94 | 99 | 91 |
| Sweater 2 | 73 | 83 | 76 | 86 | 95 |
| Towel | 80 | 90 | 97 | 95 | 100 |
| Wall | 58 | 85 | 87 | 91 | 90 |
| Wood | 90 | 94 | 96 | 99 | 99 |
| Average | 71 | 77 | 81 | 85 | 85 |
| 90+ | 6 | 7 | 10 | 12 | 13 |
| 80+ | 12 | 14 | 17 | 17 | 18 |

Table 2: Classification Accuracy. 100 Examples. Cutoff 0.01.

error. It also mixed up asphalt, concrete, sand, the curtain and one of the carpet images. In general, when the system misclassified samples, it had a tendency to associate them with categories that are considered similar by human standards. It rarely associated grass with sand or concrete with pine needles for example.

## 8. Evaluation, Improvements and Future Directions

It is clear that the system is capable of identifying some textures with high accuracy. It would be nice to know why other textures were not so accurately identified and what can be done to improve the overall performance.

The weak link in the system is probably the feature computing routines. No methodology has been developed to ensure that all of the relevant information is extracted from an image patch. When that information was present, the information tree demonstrated the capacity to use it very efficiently. However, there is no way to compensate for a lack of information.

Hence, one of the future directions of our research will be to explore methods for generating sets of features that cover the data. We will also explore the way in which features change under different lighting conditions. We will also look at how changes in other viewing conditions affect the recognition problem. In addition, we will look at ways in which knowledge about texture can help to solve other problems in computer vision. We have already alluded to the fact that texture recognition and discrimination can aid in image segmentation. We believe that this and similar techniques will also aid in stereopsis, motion, and object recognition.

## 9. Acknowledgements

We would like to thank Henry Cejtin for suggesting that we base our splitting criterion on information theoretic grounds, Dave Ballman for writing the camera interface and Doug Knights for help with some of the image manipulation code.

## References

[1] L. Brieman, J.H. Friedman, R.A. Olshen, C.J. Stone, "Classification and Regression Trees," Wadsworth, Belmont, California (1984).

[2] Ronald Christensen, "Entropy Minimax Multivariate Statistical Modeling–I: Theory", Int. J. of General Systems, 11 (1985) 231–276.

[3] T. M. Cover and P. E. Hart, "Nearest Neighbor Pattern Classification," *IEEE Transactions on Information Theory*, IT-13:1 (1967) 21–27.

[4] Jerome H. Friedman, Jon Louis Bently, and Raphael Ari Finkel, "An Algorithm for Finding Best Matches in Logarithmic Expected Time," ACM Transactions on Mathematical Software, 3:3 (1977) 209-226.

[5] Bela Julesz, "Toward and Axiomatic Theory of Preattentive Vision", in *Dynamic Aspects of Neocortical Function*, ed. by G. Edelman, *et. al.*, John Wiley and Sons, New York (1984) 585–612.

[6] Stephen M. Omohundro, "Efficient Algorithms with Neural Network Behavior," Complex Systems 1:2 (1987) 273-347.

[7] J.R. Quinlan "Learning efficient classification procedures and their application to chess end games," in Machine Learning: An Artificial Intelligence Approach, R. Michalski, T.M. Mitchell, J.G. Carbonell eds., Tioga (1983) 463-482.

[8] Larry Rendell, "A new basis for state-space learning systems and a successful implementation," Artificial Intelligence 20:4 (1983) 369-392.

| BIBLIOGRAPHIC DATA SHEET | 1. Report No. UIUCDCS-R-88-1409 | 2. | 3. Recipient's Accession No. |
|---|---|---|---|

4. Title and Subtitle

FAST TEXTURE RECOGNITION USING INFORMATION TREES

5. Report Date: February 1988

6.


7. Author(s): Darrell R. Hougen and Stephen M. Omohundro

9. Performing Organization Name and Address

Department of Computer Science
1304 W. Springfield
Urbana, Illinois 61801


8. Performing Organization Rept. No.

10. Project/Task/Work Unit No.

11. Contract/Grant No.

12. Sponsoring Organization Name and Address

13. Type of Report & Period Covered: Technical

14.

15. Supplementary Notes


16. Abstracts

In this paper we approach the problems of texture recognition and discrimination using information trees, an information theoretic technique for efficient category learning in continuous domains. Texture recognition is important in the early stages of visual information processing because it provides a method for identifying materials from their visible surface properties. In addition, it allows images to be segmented into meaningful regions prior to object recognition.

Small, $n \times n$ patches are extracted from video images and used as training examples. Each patch is associated with a texture category by a teacher. Real valued features representing morphological properties of the patch are computed at a variety of scales and orientations. The computed values identify the patch with a point in a visual feature space. The most relevant dimensions for categorization are extracted by the information tree building algorithm. The feature space is recursively decomposed into hyper-rectangular regions representing textures classes. The resulting hierarchical decomposition is stored in a $k$-$d$ tree—a binary tree containing both a discrimination value and dimension at each node. The splitting of a parent hyper-rectangle into its two children is determined by a universal information theoretic criterion. Information trees generalize by assigning a single category label to an entire leaf region.



17. Key Words and Document Analysis. 17a. Descriptors

visual texture
machine learning
information tree
k-d tree


17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group

| 18. Availability Statement | 19. Security Class (This Report) | 21. No. of Pages |
|---|---|---|
| unlimited | UNCLASSIFIED | 18 |
| | 20. Security Class (This Page) UNCLASSIFIED | 22. Price |