The Adaptive Self Parameterising Texture Region Boundary Tracker

D M Booth *and J E W Mayhew †

The Research Initiative in Pattern Recognition,
RSRE, St. Andrews Road,
Malvern, Worcs. WR14 3PS

A texture segmentation algorithm is described. Its approach is to locate the texture boundary within a processing window, and then follow the discontinuity through the image.

The system operates in two modes : bootstrap and feed forward mode. In bootstrap mode a portion of the image is segmented, the texture boundary is located, and all of the parameters necessary for configuring the system are found. These include the resolution of the segmentation, identification of the most effective features for distinguishing between the two textures, and an estimate of the texture's conditional probability density functions (pdfs) given the chosen feature set. In feed forward mode the location of the texture boundary is used to position the next processing window, and this is segmented using parameters supplied by the bootstrapper.

Bootstrapper

The bootstrapper provides answers to the following questions :

- What texture measures should be used?
- Over how small an area can the texture measures be computed with a reasonable level of statistical confidence?
- How is the training set to be derived and modeled?
- What type of clustering algorithm should be employed?
- What will be the source of the prototype texture regions needed for solving the previous problems.

The following sections show (in order of program execution) how each of the questions above have been answered. Progress is illustrated using results gained from processing the image shown in figure 1 (512x512 pixels, 256 grey levels).

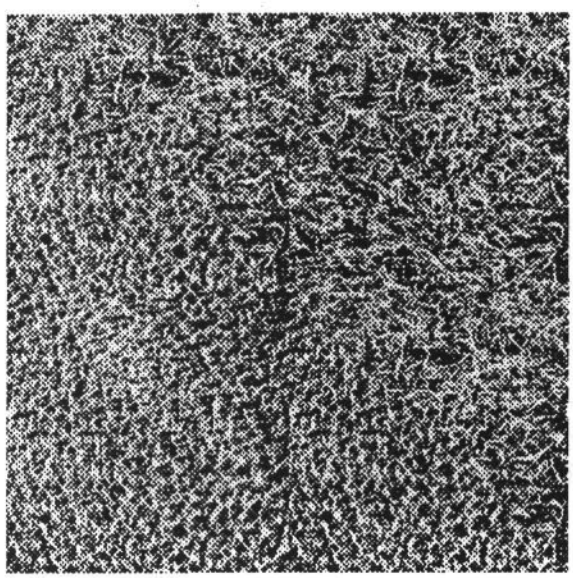

Figure 1. Test image.

*Seconded by The Royal Signals and Radar Establishment.

†AI Vision Research Unit, University of Sheffield, Sheffield, Yorks.

AVC 1989 doi:10.5244/C.3.26

Figure 1 consists of two independently histogram equalised Brodatz textures [1] with a boundary running from the top left hand corner to the bottom right. The top left hand quadrant has been selected as input to the bootstrapper.

Preliminary Segmentation

An example of both textures needs to be made available to the system. The approach adopted here is to use a split and merge algorithm [2] to provide a preliminary segmentation, from which two differently textured regions are identified by means of a statistical test on feature values computed over their sub-blocks.

Split is a recursive procedure which, beginning with the whole subimage, divides nonhomogeneous image blocks into four quadrants. An image block is considered to be nonhomogeneous when the difference in feature values between any two of its quadrants exceeds a user-specified split threshold. Recursion depth is limited by a restriction placed on the minimum size of an image block. The segmentation is represented by a quadratic picture tree, where the root node corresponds to the current subimage. Each non terminal node has 4 children, each of which corresponds to a different quadrant of the parent image block.

Split produces a set of image blocks of varying size. These are input to a routine which assigns a common label to adjacent regions separated by a distance in feature space of less than a grouping threshold.

The split and merge routine employs one texture measure. If more were to be used, the most variable of them would tend to dominate, and effort would be wasted computing features that contribute very little to the discrimination process. It is not possible to standardise the feature values without prior knowledge of their distributions for each texture.

The preliminary segmentation is shown in figure 2. The grey level of an image block is linearly related to the value of the texture statistic computed over the same area.

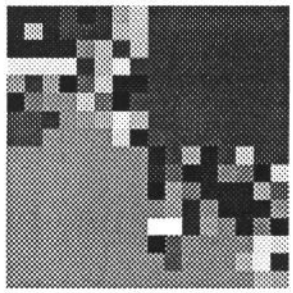

Figure 2. Preliminary segmentation.

A representative of both texture classes is selected from the set of regions produced by the split and merge algorithm. These are identified by first considering the two largest regions. Both are divided into sub-blocks, and texture measures (§2.3) are evaluated for each sub-block. These two samples under go Hotelling's T^2 test [8] for evidence that they originate from the same population, in which case the smaller of the regions is discarded and the next largest region is considered.

Window Size

To achieve a high resolution segmentation texture features must be computed over as small a processing window as possible. However, small windows produce features with reduced statistical confidence. In particular, the variance of the texture measures will increase dramatically when the processing window is smaller than the texture primitive, and as a result classification will become unreliable. Window size is particularly important when the segmentation algorithm does not employ overlapping windows, in which case the window size defines the resolution of the segmentation. Hence it would be desirable to find the size of the texture primitives and set the minimum size of a processing window accordingly. Conners et al. [4] describe a technique for computing texture primitive sizes using inertia statistics derived from cooccurrence matrices. A cooccurrence matrix contains estimated probabilities that two pixels $f(x1, y1)$ and $f(x2, y2)$, which are separated by a distance, δ, and an orientation, θ, have grey levels i and j respectively. Various statistical texture measures can be calculated from such a matrix [4,7], in par-

ticular, the inertia measure is given by

$$I(\delta,\theta) = \sum_{i=0}^{N_G-1} \sum_{j=0}^{N_G-1} (i-j)^2 s(i,j|\delta,\theta)$$

where $s(i,j|\delta,\theta)$ is the (i,j) th element of the matrix (δ,θ) and N_G is the number of grey levels. Inertia can be seen as measuring the dissimilarity between many pairs of pixels separated by (δ, θ). E.g for a 1 dimensional data stream that exactly repeats itself every λ pixels, the inertia measure will have a minimum value of zero when $\delta = \lambda$. Likewise, the size of the smallest texture primitive is given by twice the displacement corresponding to the first inertia peak, for fixed θ. This procedure should be repeated for several orientations, e.g. θ = 0, 45, 90 and 135 degrees. If the feature set is invariant under spatial rotation then the largest primitive size must be used, otherwise one could choose to evaluate features in only those directions where the primitive size is smallest, thus gaining as much resolution as possible.

Feature Selection

The feature employed by the split and merge algorithm is hard wired, and chosen on the basis of its performance with respect to various image products. Any reliable measure would suffice, Chen and Pavlidis [3] used correlation. The feature sets used by the clustering procedure are chosen dynamically from some predetermined group of statistical texture measures. We have implemented seven second order cooccurrence statistics : inertia, energy, local homogeneity, entropy, correlation, cluster shade, and cluster prominence. The single most reliable of the cooccurrence statistics was found, in our experiments with natural textures, to be inertia, and consequently it was incorporated into the split and merge routine. Of the 7 cooccurrence statistics, the number that can actually be employed by a classifier at any one time is limited by the so called "curse of dimensionality". As a rule of thumb there should be at least five training samples per feature for each texture class being considered [6]. Given a feature set of cardinality, D, it is necessary to determine a subset containing d features which can best discriminate between the two texture classes. The only way of guaranteeing an optimal solution when dealing with a restrictive number of training samples is by examining every subset of cardinality d, however, this is unlikely to be computationally feasible. A less intensive strategy that can give reliable results is "plus l take away r" [5], here configured $l = 2$, $r = 1$. In common with all bottom-up search strategies, processing begins with a full set of available features and an empty current feature set. The "plus 2" routine identifies the 2 available features that when combined with the current feature set have greatest discriminatory power, and then moves them to the current feature set. The "take away 1" routine identifies the single feature that when removed from the current feature set has the least effect on its discriminatory power, and moves it back to the set of available measurements. A feature set of cardinality d is constructed by performing $d-1$ iterations of the plus 2 take away 1 routine, and then adding the final feature using "sequential forward selection" (i.e. plus 1). The discriminatory power of a feature set is measured by the Bhattacharayya distance between the two normally distributed texture classes ω_1 and ω_2,

$$B(\omega_1,\omega_2) = \frac{1}{8}(\overline{u_1} - \overline{u_2}) \left[\frac{\sum_1 + \sum_2}{2}\right]^{-1} (\overline{u_1} - \overline{u_2})^T$$

$$+\frac{1}{2} * ln \left[\frac{|\frac{1}{2}(\sum_1 + \sum_2)|}{|\sum_1|^{1/2}|\sum_2|^{1/2}}\right]$$

where $\overline{u_i}$ is the mean feature vector for class i, $\sum_i$ is the covariance matrix for class i, and $|\sum_i|$ is the determinant of $\sum_i$.

Training set

The image blocks in the preliminary segmentation are classified using training samples extracted from the prototype texture regions. Since classification performance usually increases as a function of the area over which the texture measures are computed, the quadtree is pruned in such a way that any node whose children are all of the same texture class takes on the identity of the child nodes, which are subsequently deleted. At each resolution, feature values are computed over windows contained within the two prototype texture regions, and a pair of conditional pdfs $(p(\mathbf{x}|\omega_j))$ are fitted.

Clustering

The quadratic picture tree is parsed. At each terminal node which is not a member of a training set, the corresponding image block is classified as belonging to either one or other of the two texture classes. Each image block is characterised by the feature vector, $\mathbf{x}$, which enables the *a priori* probabilities $P(\omega_j)$ to be converted to *a posteriori* probabilities, $P(\omega_j|\mathbf{x})$, by using the Bayes decision theory

$$P(\omega_j|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_j)P(\omega_j)}{p(\mathbf{x})}$$

where

$$p(\mathbf{x}) = \sum_{j=1}^{2} p(\mathbf{x}|\omega_j)P(\omega_j).$$

Classification is made such that

class ω_1 is chosen if $P(\omega_1|\mathbf{x}) >= 1 - \lambda_r$
or class ω_2 is chosen if $P(\omega_2|\mathbf{x}) >= 1 - \lambda_r$
otherwise ω_0 is chosen.

where λ_r is a reject threshold, and rejected samples are assigned to class ω_0.

An image block is normally classified using the pdfs which correspond to the resolution of that image block. However, if the discriminatory power of the feature set is greater at a higher resolution, then the block is split into 4, and each quadrant is classified independently.

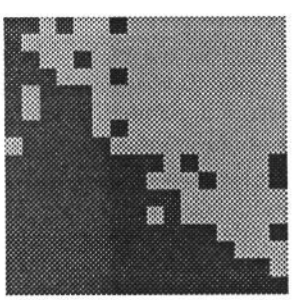

Figure 3. After clustering.

Overview

The bootstrapper is described schematically in figure 4. The squares and rectangles represent processes and parameters, respectively. To summarise, on the first pass of the split, merge and prune routine the minimum block size takes on a "safe" value which is larger than any of the texture primitives. The primitive sizes are evaluated for each of the resulting large image blocks. The split, merge and prune process is then executed a second time using a minimum block size commensurate with the largest primitive size. The training set module identifies the two prototype texture regions, from which it calculates the number of training samples available at each resolution, and hence the number of features that can be employed by the classifier. The best features for distinguishing between the two textures are then determined for each resolution, and the conditional pdfs fitted. The image blocks are classified using a small value of λ_r thus providing a good safeguard against classification errors. The resulting segmentation provides the training set for the next iteration of the algorithm. Since the new training set should be larger than the previous one, the conditional pdfs should have better estimates, and it may be possible to include more features in the classification process, and consequently, a superior segmentation might be achieved. For computational reasons the Bayes classifier is only invoked twice, with just the first of these incorporating a possibly active reject threshold.

Feed Forward

This section describes the procedure for locating and modeling the texture boundary. The algorithm has four parts :

•**Blob Removal**. Regions in the segmented image that are totally surrounded by the other texture class are detected using template matching and then deleted.

•**Boundary Extraction**. The texture boundary is located by scanning the deblobbed image from either the left and right, or top and bottom, depending on the predicted orientation of the boundary. The scan lines stop when they encounter a class transition. If the two opposing scan lines stop at the same place then their point of intersection is assumed to lie on the boundary.

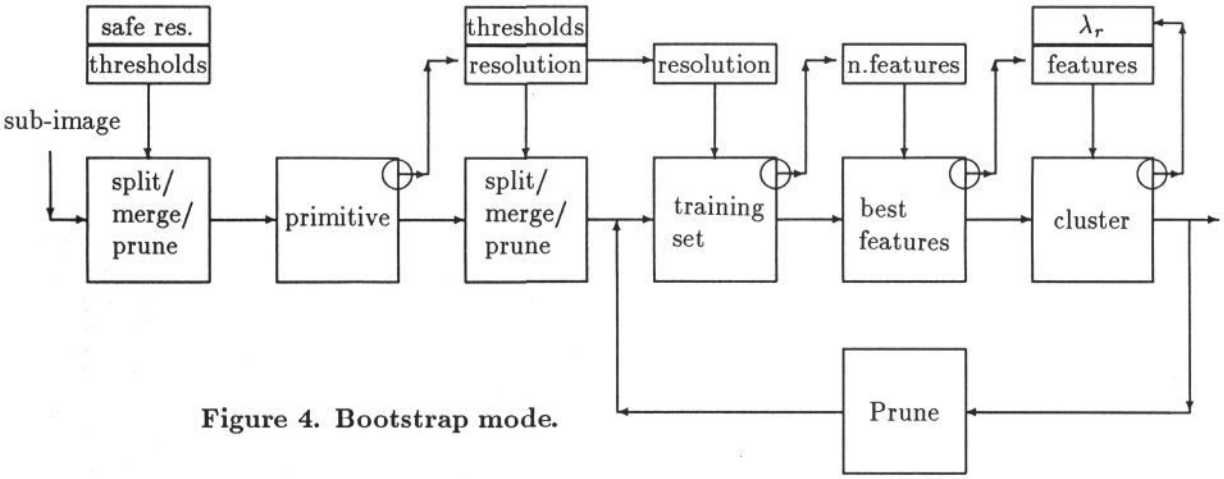

Figure 4. Bootstrap mode.

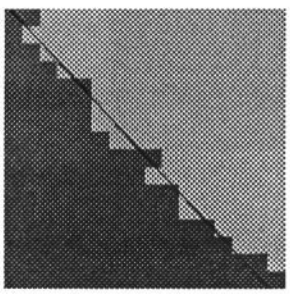

Figure 5. After blob removal and boundary modeling.

•**Boundary Modeling**. A set of boundary points is modeled by a 1st or 2nd order polynomial (see figure 5).

•**Window Positioning**. The centre point of the next processing window is chosen as the intersection of the boundary line with the current processing window.

The parameters obtained from the bootstrapper are fed forward for the segmentation of subsequent processing windows. Since no parameter estimation is required these windows can be smaller (128 pixels square) and the segmentation process is much simplified (figure 6).

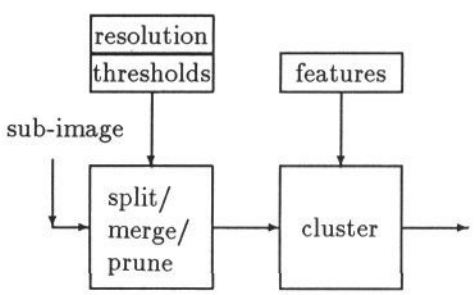

Figure 6. Feed forward mode.

Results

The final segmentation of figure 1 is shown in figure 7. The result is quite favourable, particularly as humans seem to find these two textures difficult to separate. The difference in quality between the preliminary segmentation and the final result demonstrates the benefit of working in a multidimensional feature space, at least where cooccurrence statistics are concerned. Substantial computational savings are possible using the tracking approach as oppose to segmenting the whole image.

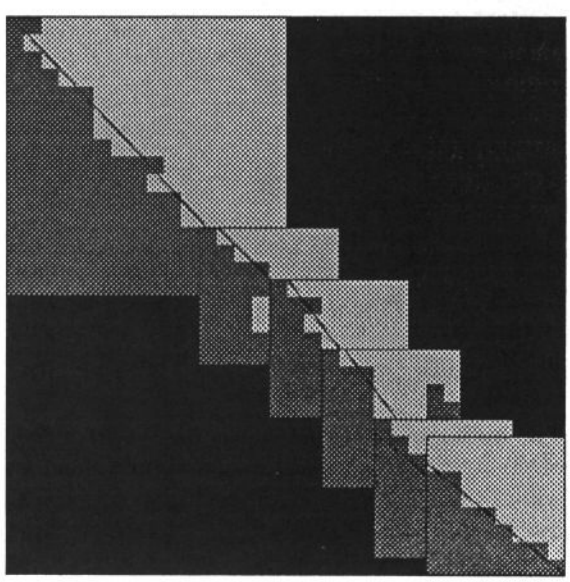

Figure 7. Final segmentation.

Current issues being addressed include :

• The automatic selection of an effective split and merge feature.

• Automatic derivation of split and merge thresholds.

• The addition of a performance monitoring process to run in parallel with the tracking software. As viewing conditions and textural properties change the monitoring process will either update the relevant parameters or instruct the tracking module to perform another bootstrap.

• Increasing the segmentation resolution by overlapping those texture measure windows lying close to the boundary (see figures 8 and 9) and also by fitting a surface to the *a posteriori* probabilities and interpolating for the boundary position.

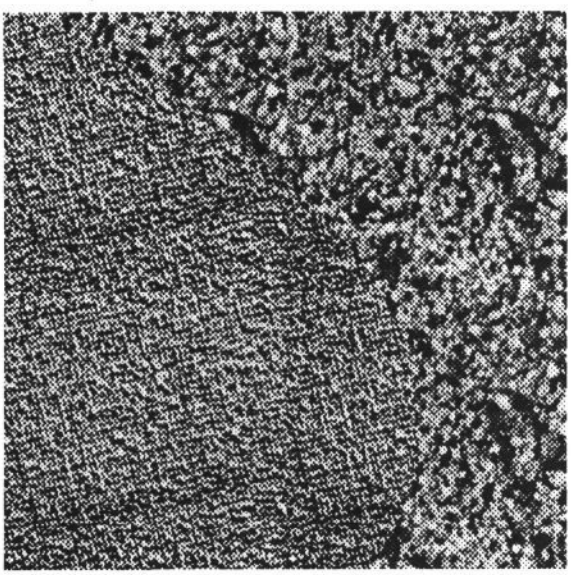

Figure 8. Input image.

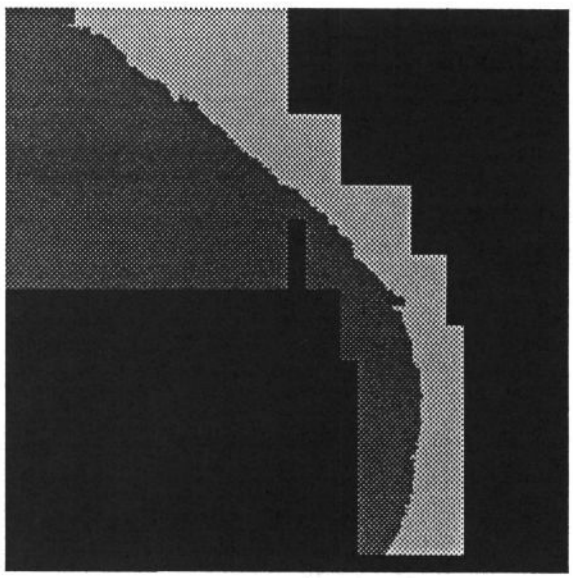

Figure 9. Output image.

Acknowledgement

The authors should like to thank Mr John Sabey of the Royal Signals and Radar Establishment with whom they had several informative and enjoyable discussions.

References

1. **Brodatz, P.** *Texture : photographic album for artists and designers*, New York, Dover (1966).

2. **Chen, P.C. and Pavlidis, T.** "Segmentation by texture using cooccurrence matrix and a split-and-merge algorithm," *CGIP*, 10, (1979) pp.172-182.

3. **Chen, P.C. and Pavlidis, T.** "Segmentation by texture using correlation," *IEEE Trans.* vol. PAMI-5, No.1, (1983) pp.64-69.

4. **Conners, R.W. et al.** "Segmentation of a High-Resolution Urban Scene Using Texture Operators," *CVGIP* 25, (1984) pp.283-310.

5. **Devijver, P.A. and Kittler, J.** *Pattern Recognition : A statistical approach*, Prentice/Hall International (1982).

6. **Foley. D.H.** "Considerations of sample and feature size," *IEEE Trans*, vol. IT-18, (1972) pp618-626.

7. **Haralick, R.M. et al.** "Textural features for image classification," *IEEE Trans*, vol. SMC-3, no.6, (1973) pp.610-621.

8. **Press, W. et al.** *Numerical Recipes, the art of scientific computing," Cambridge University Press* (1986).