A New Data Structure For The Implementation Of Unsupervised Texture Segmentation

Padmapriya N, Pradeep P.P and Whelan P.F

Vision Systems Laboratory
School of Electronic Engineering
Dublin City University
Dublin 9, Ireland

E-mail: {padmapri,pradeepp,paul.whelan}@eeng.dcu.ie

Abstract — **In this paper, we propose a novel technique to implement an algorithm for unsupervised texture segmentation. Local Binary Patterns (LBP) have been demonstrated to provide a robust and efficient framework for texture segmentation. The distribution of local binary pattern and the contrast measure is used to evaluate the similarity between the adjacent image regions during the segmentation operation. The algorithm uses a new data structure. The existing data structures such as trees and graphs are modified to construct a quadtree and mergegraph. Quadtree is used for splitting the image into blocks of roughly uniform textures and mergegraph is used for merging similar adjacent regions. The algorithm has been tested using mosaic images consisting of various textures, in addition to natural scenes. This approach proved to yield computationally efficient segmentation of images.**

Keywords — **Local Binary Pattern, Quadtree, Splittree, Mergegraph, Segmentation algorithm.**

I Introduction

Texture analysis has been an active field of research for the past three decades and has useful applications in industry, biomedical surface inspection, remote sensing, document analysis and many more areas of computer vision. There are numerous techniques for the extraction of texture features and various methods for segmentation. Most of the methods are developed for grey scale textures. Implementation of these methods using hierarchical splitting and agglomerative merging is computationally intensive. We have developed a strategy to implement an algorithm for unsupervised texture segmentation. In the existing texture segmentation methods one of the many data structures is used. This paper is based on a novel data structure using splittree and mergegraph for unsupervised texture segmentation. The performance of the algorithm is tested on grey scale and colour images.

Ojala *et al* [1] developed *Local Binary Pattern* (LBP) and a method for unsupervised texture segmentation and found excellent results using Brodatz textures [2], hence our analysis is focussed on this technique. The major steps in this algorithm are texture feature extraction, image splitting and merging. This algorithm is implemented using a quadtree, to split the image into blocks of roughly uniform texture, followed by agglomerative merging of similar adjacent regions using a mergegraph. This paper is organised as follows. Section 2 describes the feature extraction technique, LBP and the texture description using G-Statistic. Section 3 details the implementation of the split and merge algorithm. Section 4 presents the experimental results and finally section 5 concludes this paper.

II Feature extraction technique

a) Local binary pattern

Wang and He [3] introduced the Texture Spectrum method which is a novel approach for feature extraction and they proposed the usefulness of the texture spectrum for texture classification. A texture image can be decomposed into a set of small textural units, called *Texture units* (TU). A Texture unit is represented by 8 elements, each of which has one possible value (0, 1, 2) obtained from a neighbourhood of 3×3 pixels. The neighbourhood is represented by $V = \{V_0, V_1, V_2, ...V_8\}$, which generates 3^8 standard texture units. The occurrence distribution of texture units is called the *Texture Spectrum*. Wang and He [3] proved that the Texture Spectrum method is useful in discriminating different types of textures and obtained promising results using four Brodatz's natural images. They carried out a comparative classification experiment between a set of texture spectrum measures and a set of co-occurrence measures and found that both methods performed equally. Ojala *et al.* [1] proved that a simple two level version of this method performed equally well. In this version, LBP is described with $2^8 = 256$ possible texture units. The texture unit $TU = \{E_1, E_2, ..., E_8\}$ is obtained by applying the threshold operation using the following rule :

$$E_i = \begin{cases} 0 & V_i < V_0 \\ 1 & V_i \geq V_0, \end{cases}$$

where V_0 is the center pixel. The LBP is determined as follows,

$$LBP = \sum_{i=1}^{8} E_i * 2^{i-1} \tag{1}$$

LBP does not take into account the contrast of texture which is the measure of local variations present in an image and is important in the description of some textures. Hence the LBP is combined with a simple contrast measure C, which is the difference between the average grey levels of pixels with value 1 and pixels with value 0 contained in the texture unit.

b) Description of textures using G-statistic

Texture is classified based on the feature distributions. The texture regions are defined by two feature distributions, the LBP distribution and the contrast distribution, since one texture measure is inadequate to describe the spatial structure of the local texture. The discrimination between the two distributions is performed using G-Statistic [4]. The LBP/C distribution is approximated by 2D distribution of size $256 \times b$, where b is the number of bins for C. As mentioned in [4], the best value for this variable, b is 8 or 16. The value of G-Statistic indicates the probability that two sample distributions come from the same population. If the value of G is high, the probability that the two samples are drawn from the same population is low. The similarity of the two histograms are measured with a two-way test of interaction or heterogeneity.

$$\begin{aligned} G = 2\Bigg\{ & \Bigg[\sum_{s,m} \sum_{i=1}^{n} f_i \log f_i \Bigg] \\ & - \Bigg[\sum_{s,m} \Big(\sum_{i=1}^{n} f_i \Big) \log \Big(\sum_{i=1}^{n} f_i \Big) \Bigg] \\ & - \Bigg[\sum_{i=1}^{n} \Big(\sum_{s,m} f_i \Big) \log \Big(\sum_{s,m} f_i \Big) \Bigg] \\ & + \Bigg[\Big(\sum_{s,m} \sum_{i=1}^{n} f_i \Big) \log \Big(\sum_{s,m} \sum_{i=1}^{n} f_i \Big) \Bigg] \Bigg\} \end{aligned} \tag{2}$$

where f_i is the frequency at bin i; s and m are the two sample histograms; and n is the number of bins.

III Algorithm development

Segmentation of an image entails the division or separation of the image into regions of similar attributes. If the number of possible textures is too large, or if no assumptions can be made about the type of textures, then an unsupervised texture segmentation is used. In the case of unsupervised texture segmentation, statistical analysis is performed on the entire distribution of vectors, and the aim is to recognise clusters in the distribution and assign the same label to them all. This method follows hierarchical splitting and agglomerative merging procedure. The proposed algorithm implements the splitting using a quadtree structure and merging generates a forest structure with each tree of forest representing a merged area in the image.

a) Implementation of hierarchical splitting

The hierarchical splitting divides the image into blocks of roughly uniform texture. A block is split into 4 subblocks based on a uniformity test. The six pairwise G values between the LBP/C histogram of the 4 subblocks are calculated. The

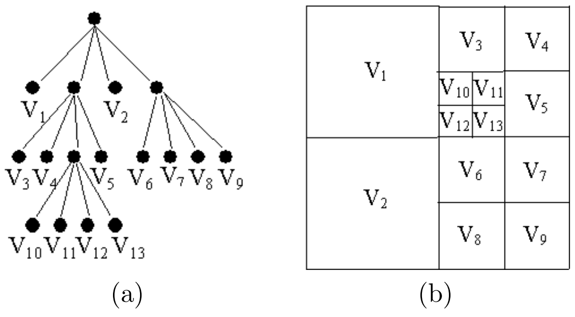

Fig. 1: A typical example of the splittree and the corresponding image representation

maximum G value is represented by G_{max} and the minimum G value by G_{min}. The uniformity of the region was tested with $R = \frac{G_{max}}{G_{min}} > X$ where X is a threshold value. If the relation $R > X$ is true, then the block is considered to be non-uniform and is split further into four blocks. The procedure of splitting the block was repeated recursively on each subblock until a block size S_{min} is reached. The value for S_{min} is 16×16 or 4×4 [4]. This algorithm is implemented using a quadtree structure. A quadtree structure is called a splittree (Fig. 1(a)) with every parent node having 4 child nodes and the corresponding image representation is shown in Fig. 1(b). This algorithm generates a splittree by using an iterative procedure. The tree is initialised with one node referring the whole image. The iterative procedure traverses through the leaves of the tree. If the image area corresponding to a particular leaf is nonhomogeneous then it is split into 4 child nodes. This iteration stops when there are no more leaves to be split or the image size of an unsplit leaf is S_{min}. The leaves of the splittree at the end of the iteration procedure refers to a homogeneous region in the image. These leaves are denoted as $\mathbf{V} = \{\mathrm{V}_1, \mathrm{V}_2, \mathrm{V}_3, ..., \mathrm{V}_{N_s}\}$ where each V_i is a leaf and the i^{th} index is selected arbitrarily and N_s is the number of homogenous regions s. Also, the union of all V_i equals the whole image. Pseudo code for the splittree approach is given below:

```
NodeType Head ;
// Image is assigned to Head of Tree
Head.Image = Image;
while(1)
{// Start Iteration
int No_of_leaves ;
//Scan through All the leaves
NodeType *Leaf = GetLeaves(Head,&No_of_leaves);
FLAG = 0 ;
for(i = 0 ; i < No_of_leaves ; i++)
{
//Check for homogeneity
 if(!Homogeneous(Leaves[i].Image))
  {
//split the image of ith leaf into four.
   split(Leaf[i])
   FLAG = 1 ;
  }//end if
}//end for
if(FLAG ==0)//FLAG = 0 for Exit Condition
break;
}//end while
```

b) Implementation of agglomerative merging

An agglomerative merging procedure was applied to the image which has been split into blocks of roughly uniform texture. This procedure merges similar adjacent regions until a stopping rule is satisfied. The pair of adjacent segments which has the smallest *Merger Importance* (MI) value are merged. MI was calculated from $MI = p \times G$, where p is the smallest number of pixels among the two regions and G is the value of G-Statistic. After merging, the two respective LBP/C histograms are summed to be the histogram of the new image region, which alters the segmented image. The G distribution between the new region and all adjacent regions are then computed. The merging procedure is adopted until the stopping rule is satisfied. $MIR = \frac{MI_{curr}}{MI_{max}} > Y$, where MI_{curr} is the Merger Importance of the current merge, and $MI_{max}$is the largest merger importance of all preceding merges. If the ratio MIR exceeds the threshold value Y, the merging procedure is halted. Theoretically, in the initial merges, the adjacent regions with identical LBP/C histograms, have a zero MI value which will lead to termination of merging prematurely. Hence for the first few merges, the stopping rule was not verified [4]. The threshold value depends on the image and is tested for each image experimentally.

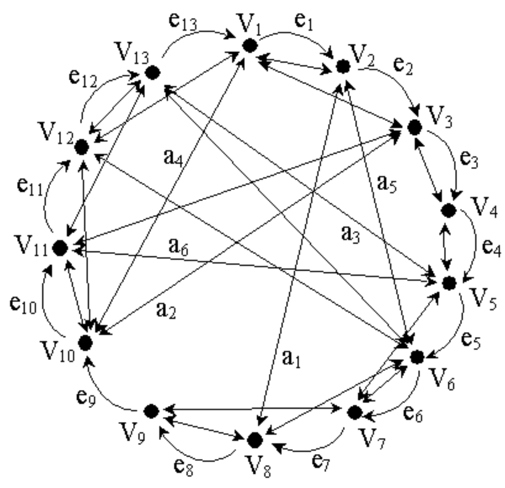

Fig. 2: A typical example of mergegraph

The implementation of agglomerative merging is done by creating a mergegraph. A mergegraph is a triplet $G = (\mathbf{V}, A, E)$ where $\mathbf{V}$ is the set of nodes and A and E defines two types of links which connects these nodes.

- $\mathbf{V} = \{\mathrm{V}_1, \mathrm{V}_2, \mathrm{V}_3, ..., \mathrm{V}_{N_s}\}$, where V_i is the set of leaves of the splittree.
- A=$\{a_1, a_2, a_3, ..., a_n\}$, where a_i is the set of adjacent links between V_i and V_j.
- E=$\{e_1, e_2, e_3, ..., e_n\}$, where e_i is the unidirectional link between V_i and $\mathrm{V}_{(i+1)modn}$. This forms the circular link list of nodes.

Fig. 2 shows a typical mergegraph for the splittree and the image illustrated in Fig. 1. The mergenode procedure traverses through the circular link list. The merger importance value for all the adjacent nodes are found and the region with smallest merger importance value is merged with the current node. All the newly merged nodes are delinked from the circular link list and the merging information and adjacency information for the current node is updated. This procedure is repeated until a stopping rule is satisfied. The pseudo code outlined below shows one such iteration.

```
//Head points to the first node of Mergegraph
 NodeType cnode;
//if current node is not equal to head
 for(cnode = Head;cnode!=Head;cnode=cnode->next)
 {
//Merge adjacent node of smallest merger
  importance with the current node
   Merge_Adj(cnode);
//update adjacent nodes for the new cnode
   update_Adj_Information(cnode);
//update merge Information of cnode
   update_merge_Information(cnode)
 }//end for
 if(cnode(mergecondition))
 {
   AddForest(cnode)
 }//end if
```

IV Results and Discussion

The images used for colour texture analysis are from the VisTex image database [5]. Segmentation results for four texture mosaics and two natural scenes are presented. The mosaics are also created using the VisTex [5] image database. For testing the algorithm, four images of 256 × 256 pixels were constructed with various combinations of the VisTex texture images of 128 × 128 pixels. The texture mosaics in Fig. 3(a) and Fig. 3(b) are constructed using four different textures whereas in Fig. 3(c) and Fig. 3(d), the texture mosaics are constructed with two textures and four segments. The later is done to verify whether the segmentation is based on the blocks or texture. The result shows that the segmentation depends on the texture and not on the blocks. The natural images consist of a bird in sea, of size 256×256 and rocks in sea of size 384×384 are also considered for testing. The results discussed here are presented for grey scale images, R, G, and B planes. The split threshold value lie between 0.9-1.2 and the merge threshold value lie between 1.2-1.4 depending on the mosaic and natural scene images. These values are obtained after repeated trails. Figure 3 depicts the segmented grey scale images. Fig. 4 and Fig. 5 show the results in R, G and B planes respectively. The segmented results in an individual plane depends on the colour components of the image. Fig. 4 indicates that the stopping rule used in these experiments allows the merging of the small components into a single region. Different values for the stopping rule parameter will result in the termination of segmentation before these small regions are merged. Hence the LBP method is more sensitive to the splitting and merging threshold values. Fig. 5 has associated a large number of small features and hence several small blocks remain unmerged after agglomerative merging procedure [6]. The result in Fig. 5 illustrates that the segmentation is better in B plane as compared to other planes, since the blue component is dominant in the image. The method applied is a good representation of a split image and the graph derived from the splittree forms the initial point for the merge procedure. On iterative merging procedure the mergegraph breaks up into independent trees which is referred as a forest data structure.

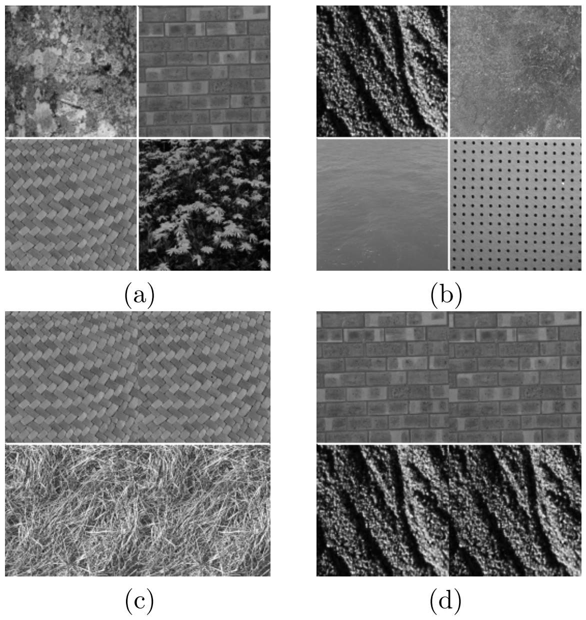

Fig. 3: (a),(b) Segmented texture mosaics with four different textures. (c),(d) Segmented texture mosaics with two different textures

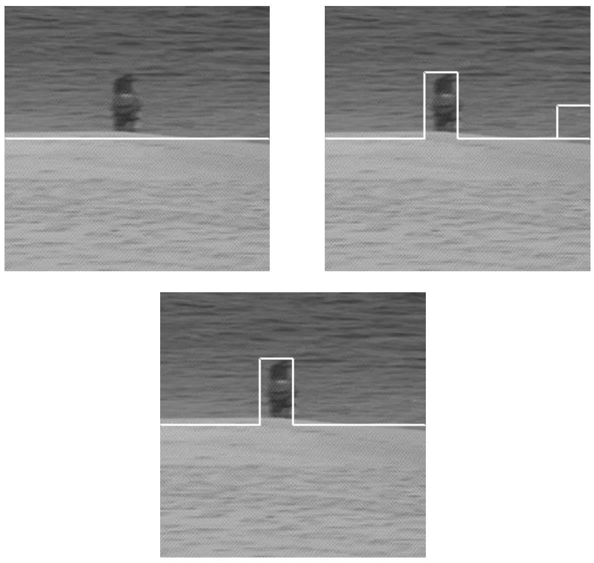

Fig. 4: Segmented image of a natural scene in R, G and B plane respectively

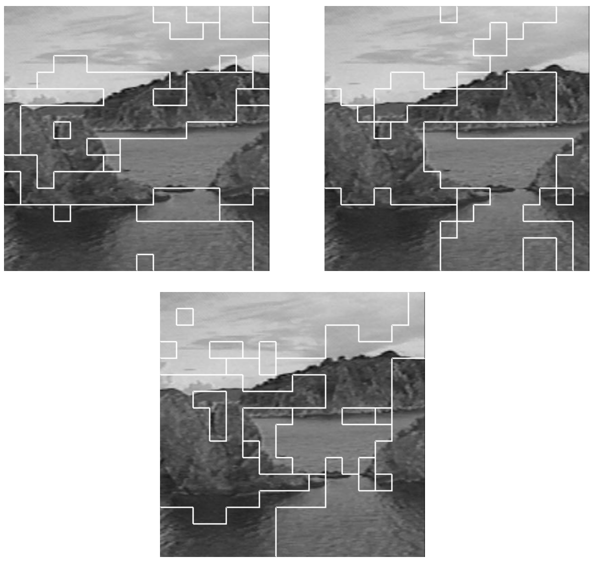

Fig. 5: Segmented image of natural scene in R, G and B plane respectively

Each tree in this forest represents a merged area. A split region is represented by the coordinates of the square block. Adopting this representation reduces the computational complexity. The efficiency of the algorithm is tested based on the processing time requirement for a given input size on a specific machine. The time taken to split a 256×256 image is 0.581 seconds, and 0.691 seconds for merging the same image. For a Pentium 3 processor, with 933MHz speed, the algorithm needs 1.272 seconds to perform the required task. The average processing time required for this image is 1.28 seconds irrespective of the planes from which the features are extracted. For an image size of 384×384, the computational time for splitting the image is 2.163 seconds and merging the image is 6.089 seconds. The average time needed for this image is 7.757 seconds. From the above results of the geometrical and natural images it is evident that the colour has an important contribution to the discriminative power of the features [7]. Also the inclusion of colour can increase the segmentation results without significantly complicating the feature extraction algorithms. Extending these experiments in different colour spaces will result in good segmentation.

V Conclusions

In this paper we outlined an unsupervised texture segmentation algorithm based on a split and merge technique. The algorithm is based on a quadtree data representation where a square image segment is broken into four quadrants, if the original image segment is nonuniform in attribute. Adjacent pairs of regions are merged using the mergegraph. Since the algorithm is more structured, it is easy to implement unsupervised texture segmentation, leading to consistent results.

References

[1] T. Ojala , M. Pietikainen and D. Harwood. "A comparative study of texture measures with classification based on feature distributions". *Pattern Recognition*, 29(1):51-59, 1996.

[2] P.Brodatz. "Texture-A photographic Album for Artists and Designers". *Reinfold, NewYork.*

[3] L. Wang and D.He. "Texture classification using texture spectrum". *Pattern Recognition*, 23(8):905-910, 1990.

[4] T. Ojala and M. Pietikainen. "Unsupervised texture segmentation using feature distributions". *Pattern Recognition*, 32:477-486, 1999.

[5] VisTex. Colour Image Database. *http: //www-white.media.mit.edu/ vismod/ imagery/ VisionTexture/ vistex.html.*,2000.

[6] D.K.Panjwani and G.Healey. "Markov Random Field models for unsupervised segmentation of textured colour images". *IEEE Transactions on Pattern analysis and Machine Intelligence*, 17(10):939-954, 1995.

[7] A.Drimbarean and P.F.Whelan. "Experiments in colour texture analysis". *Pattern Recognition Letters*, 22:1161-1167, 2001.