SCS&E Report 9316
December, 1993

Real-Time Colour Image Segmentation

Mehdi N. Fesharaki and Graham R. Hellestrand

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
THE UNIVERSITY OF NEW SOUTH WALES

Abstract

This paper describes a new method for colour image segmentation. The algorithm is based on testing the homogeneity of pixels around a center pixel by using statistical inference techniques. A 5 by 5 window around each pixel is partitioned into two sub-samples in different orientations. Then the cumulative distribution function of two sub-samples are compared with each other. Based on the Kolmogorov-Smirnov statistic, the homogeneity of two sub-samples is verified. If all pixels within the window are homogeneous, therefore, the computed statistic for all different partitionings must verify the homogeneity; otherwise, the homogeneity is rejected. As well, the computed statistic is combined with the intensity uniformity of two adjacent pixels to prevent oversegmented and/or undersegmented results. Moreover, we consider how the algorithms can be effectively implemented as a real-time hardware design.

Contact: mehdi@vast.unsw.edu.au
hell@vast.unsw.edu.au

1 Introduction

Segmentation is a critical element in image analysis and pattern recognition. The function of segmentation is to identify the homogeneous regions in an image based on properties such as intensity, colour, texture, etc. Many different approaches to this problem exist, and may be categorized as follows[1, 32]:

1. edge detection algorithms, where the focus is on the dissimilarities between two regions.

2. grouping or forming algorithms, which are concerned with the similarities between regions as a basis for merging.

This paper is primarily involved with region forming. There are different approaches for region forming such as spatial clustering, region growing, and split and merge schemes. For a complete survey of segmentation algorithms, the reader is referred to Haralick and Shapiro[11, 32], In this paper, we consider algorithms suited to pixel data, sequentially digitized at video rate. Among popular methods for image segmentation, region growing techniques are best suited to pixel stream oriented processing. In our algorithm, small regions in the image are merged using local information. This is done by computing similarity measures in adjacent regions. When this value lies below a given threshold, the regions are said to be sufficiently similar and are merged.

Two problems which exist in using this method are determining which properties are most useful as a basis for segmentation and selecting a suitable threshold. Setting the threshold inaccurately will result in "virtual" regions if it is set too low, or leakage between regions if it is set too high. Such errors may compromise the further stages in an image analysis system. Therefore, it is very important to set thresholds appropriately in a dynamic way[32].

Accurately setting thresholds relies on using the properties involved in measuring similarities between two regions. For this purpose, two different methods are used. In the first method, thresholds are set by measuring some probabilistic features of regions such as mean and variance. For example, in Gupta et al's algorithm[8], regions are grown based on two statistical tests. For this purpose, the variance and mean of a small pixel group is compared with the variance and mean of already formed neighbour regions. If the comparison satisfies a particular criterion, the candidate region is grown and the mean and variance of that region updated. Similarly, Haralick and Shapiro[11] propose a statistical t-test for comparing each single pixel with its adjacent regions in a raster fashion. If the computed statistic indicates, the new pixel is added to the region, and the mean and variance of the region is updated. A similar idea, but in reverse, is used by Levin and Shaheen[17] who define a statistic to avoid an increase in intensity dispersion in a previously formed region when a candidate pixel is added to it. In these region growing techniques, storage capacity is needed to update region properties during region growing. This technique is less suitable for a real-time image segmentation. In addition, if the light intensity varies linearly within a region R and we insist that the intensity be approximately constant within the region, then there will be artificial boundaries formed within R[29].

Due to the essential identity of edge detection and region growing[9], using edge information for region growing is another approach. For example, Yakimovsky[33] employs the likelihood function to test the strength of the edge occurring in a local area. For this

purpose, the edge information is transferred into the data structure of regions, and then by heuristic methods, regions are grown. This method also needs to store region properties in memory, and moreover, no distribution is presented for the calculated statistic. Gradient information has also been used by Haralick and Dinstein[10] for growing the regions. Generally, gradient based operators are known to be sensitive to noise.

Based on gradient relaxation techniques, Bhanu et al[4, 6] developed a segmentation algorithm, and then fabricated it in VLSI[5]. The gradient method provides control over the relaxation process by choosing some parameters which can be tuned to obtain the desired segmentation results at a faster rate. Relaxation techniques may be characterised as parallel iterative algorithms, which support the overlapped computation for each iteration, and which allow the algorithm to be implemented using a real-time, parallel pipeline design methodology. The chip fabricated by Bhanu et al classifies grey level input images with 512×250 resolution into two classes at 30 frames/second. Bhanu et al state that they believe that a chip can be developed for colour images, but they do not show how.

In this paper, we propose an image segmentation algorithm based on the region growing, and suitable for implementing in real time hardware. Due to the pixel stream oriented approach of this algorithm, only local information is used for image segmentation. It is known that ignoring global information about regions may affect the segmented output strongly, but this effect may be alleviated by exploiting local edge information in order to grow regions. The way in which the local information is interpreted affects the performance of the segmentation algorithm. In our algorithm, using the edge information in the image, regions are grown and labeled based on the probability that regions are uniform. It is shown that by using feedback from edge information in the image to set an adaptive threshold, leakage problems, which are a common disadvantage of region growing approaches[32, 11], are improved. This algorithm produces a segmented image with a unique label assigned to each homogeneous region, and uses a real-time, parallel pipelined architecture.

Colour is a primary analytical property of an image. Adding colour to the list of region properties increases the computation cost but facilitates greater insight into the homogeneity of regions. Various attributes of colour have been used in similar and different segmentation approaches[7, 3, 23, 17, 28]. However, Ohta et al[28] and Ohta[27] show that no significant difference is observed in results obtained when different sets of colour features are used. In our algorithm, the three components of colour are used as different spectrums of signal information, which helps to achieve a more accurate classification.

2 Region homogeneity

A commonly used definition of image segmentation[14] states that if I is the set of all image pixels, a segmentation of the image is a set of all connected subsets or regions $R_1,\ R_2,\ ...,\ R_N$ such that:

$$\bigcup_{i=1}^{N} R_i = I \quad where\ R_i \cap R_j = \emptyset \qquad \forall i \neq j$$

$$P(R_i) = TRUE \quad \forall i$$

$$P(R_i \cup R_j) = FALSE \quad \forall R_i \; adjacent to R_j$$

where $P(\cdot)$ is the homogeneity predicate.

On the other hand, region growing is dependent upon identifying the factors which enable the determination of the boundaries between groups of pixels which have similar characteristics. Ideally, regions in an image should not require pre-processing in order for them to be identified. Practically, however, since the illumination, the view conditions, and the spectral reflectance function vary from point to point in a region, there is no uniform distribution of properties throughout the image. Each of these factors may result in similar property measures occurring in adjacent but separate regions, thereby creating major problems for image segmentation algorithms. This means that the probability that a measurement falls within a certain interval is a useful notion in segmentation. .P Many techniques have been used to resolve the ambiguities inherent in classifying pixels with overlapping properties. One method involves a functional description of the homogeneous regions together with the use of regression functions to fit an optimal surface to the region. The error associated with the surface fit and the correlation between adjacent surfaces measures the homogeneity of that area[9, 30, 2, 18].

In our method, we look at homogeneity from a different perspective. Instead of functional characterization of homogeneous regions, we classify pixels by examining the statistics of other pixels in their neighbourhood. Specifically, we formulate the following hypotheses about pixels in a given neighbourhood:

H_0 : All pixels in the given neighbourhood belong to one homogeneous region, or
H_1 : they belong to at least two different homogeneous regions.

In order to accept or reject the null hypothesis, we spatially partition the neighbourhood of interest into two sub-regions in different orientations, and compare the cumulative distribution functions, *(cdf),* the intensities of pixels, in each pair of pixels belonging to a sub-region, and so accept or reject, with a confidence level, α, the truth, otherwise, of the null hypothesis. If the null hypothesis is accepted, the center pixel in the window is considered to be part of a homogeneous region. Figure 1 shows four such *(cdf),*'s with which homogeneity of the regions can be inferred. Compared to functional description techniques, this technique has the advantage that it does not require characterization of the regions being compared.

To compare the *cdf* of two adjacent sub-samples, the Kolmogorov-Smirnov test is used. Among non-parametric statistical methods, the Kolmogorov-Smirnov test is more suited for comparing the *cdf's* of small samples of populations[34], and has already been used for region growing by Muerle and Allen[22]. The method which they use in this test is not suitable for a pixel stream oriented algorithm, since they first segment the whole image into small cells with sizes of $2 \times 2, 4 \times 4$, or 8×8, and compute the *cdf* of each cell. Beginning with the first cell in the upper left-hand corner, the *cdf* of each cell is compared with its adjacent cells, and any neighbouring cell having a statistical distribution sufficiently similar to that of the initial cell is merged with the initial cell, and a fragment is formed. When no further adjacent cells can be found whose *cdf's* are sufficiently similar to be merged with the fragment, the fragment is complete and is defined as a region. The criterion of similarity used, is the absolute difference between

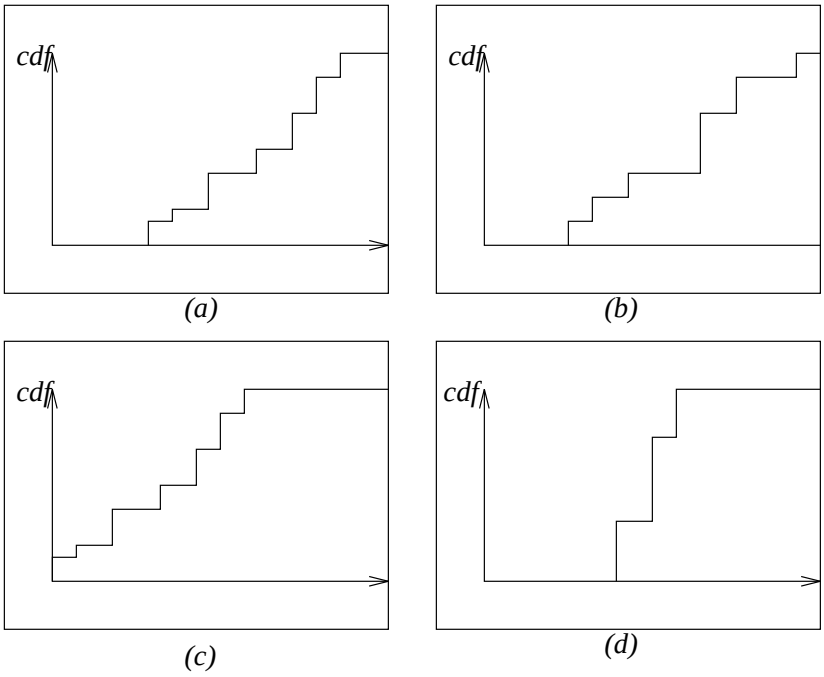


Figure 1: *cdf* in *(a)* and *(b)* come from the same population. The *cdf* in it (c) seems to be similar to that in *(a)* and *(b)*, but it has a lower intensity range and is thus from a different population. The *cdf* in *(d)* is again different from any of the others.

two distributions. If the difference between the two *cdf* is less than some threshold, the adjacent cells are merged and the *cdf* is updated. This new cell is grown to the point that none of its neighbouring cells can merge with it. This procedure is repeated for all unprocessed cells until all cells are processed. Another advantage of the Kolmogorv-Smirnov test is its ability to discriminate between some kinds of textured regions. This has been discussed by Muerle[21].

3 Testing for Homogeneity

To test the null hypothesis (which determines homogeneity), a small neighbourhood around the pixel (a 5×5 window) is spatially partitioned into two parts in four different orientations as shown in Figure 2. An edge is identified when the null hypothesis is rejected.

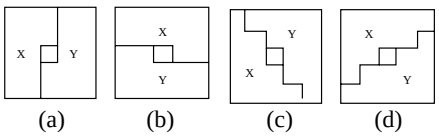


Figure 2: Partitioning the window in four different ways

In each window, let the number of pixels in the sub-regions X and Y, be m and n, respectively. Thus we obtain $N = m + n$ observations, being $X_1, ..., X_m$ and $Y_1, ..., Y_n$.

Assuming that X and Y are mutually independent and come from populations Π_1 and Π_2 respectively, the null hypothesis may be stated as follows:

$$H_0 : P(X \leq a) = P(Y \leq a), \qquad \text{for all } a \ .$$

To verify this hypothesis, the cumulative distribution function, *cdf*, of two samples are compared. To achieve this, the N observations from the two samples are ordered to form the set $Z_{(i)} : i = 1, ..., N$, where $Z_{(1)} \leq Z_{(2)} \leq .... \leq Z_{(n)}$. The statistic J is defined as follows[13]:

$$J = \frac{mn}{d} \max_{i=1,...,N} \left\{ |F_m Z_{(i)} - G_n Z_{(i)}| \right\} \tag{1}$$

where m and n are as before, d is the greatest common divisor of m and n, and

$$F_m(a) = \frac{number\ of\ X's\ \leq a}{m}$$

and

$$G_n(a) = \frac{number\ of\ Y's\ \leq a}{n}$$

are considered to be the empirical cumulative distribution functions for two random samples of sizes m and n from two distributions with *cdf F(x)* and *G(y)*.

The correctness of Equation (1) is by the virtue of Theorem 1 and an argument similar to the proof of Theorem 4.2.18 in[31].

Theorem 1*: If $F_n(x)$ is the empirical cdf of a random sample of size n from a distribution with cdf F(x), then:*

$$P(\lim_{n \to \infty} \{ \max_{-\infty < x < \infty} |F_n(x) - F(x)| \} = 0) = 1.$$

Proof: See[19].

We can simplify the calculation of J, and eliminate sorting by partitioning the range of the N observed data into ρ equal sub-ranges with length τ, as follows:

$$\tau = \frac{Z_{(N)} - Z_{(1)}}{\rho}$$

Supposing $\rho = 4$, the following set can be defined:

$$\Theta = \left\{ Z_{(1)} + \tau, Z_{(1)} + 2\tau, Z_{(1)} + 3\tau \right\} \tag{2}$$

With a 5×5 window divided into two equal sub-regions, and excluding the center pixel, we get $m = n = 12$. Now Equation 1 may be rewritten as follows:

$$J = \max_{\theta_k \in \Theta} \left\{ |F_m Z_{(\theta_k)} - G_n Z_{(\theta_k)}| \right\} \tag{3}$$

where $k = 0, 1, 2$, and $F_m(\theta_k)$ and $G_n(\theta_k)$ are defined as follows:

$$F_m(\theta_k) = number\ of\ X's\ \leq \theta_k \tag{4}$$

$$G_n(\theta_k) = number\ of\ Y's\ \leq \theta_k \tag{5}$$

From Figure 2, if the neighbourhood around the pixel belongs to a homogeneous region, all J's calculated for different orientations show a relatively small value; otherwise, there is a boundary perpendicular to the direction of the partition delivering the greatest J at that pixel. Therefore, to select and test the greatest J, all J's are ordered to form the set $\{J_{(i)} : i = 1, ..., 4\}$, where $J_{(1)} \leq ... \leq J_{(4)}$.

The next Point is the difference between edge detection and region segmentation. In region segmentation, declaration of edges even with a confidence level (α) of 1% is not enough. This is due to the leakage problem which might happen through pixels which are not declared as edge elements. In fact, the confidence level, α, shows the high probability of an edge occurring, but the reverse is not true. This does not mean that all edge elements are detected. Therefore, the interpretation of this value is important. Moreover, there are some cases where the Kolmogorov-Smirnov test incorrectly shows a high value of difference between two sub-samples. For example, where the homogeneous region has a sloped surface, one of the partitions shown in Figure 2 (the one which is perpendicular to the slope direction) may produce a large value for J. As well, this problem may arise due to noise in the image.

To resolve these problems, the statistic J_4 may be combined with an estimate of intensity uniformity around the pixel. For this purpose, the desired confidence level for homogeneity around the pixel is set less than or equal to 25%. Based on this confidence level, from Table A.23 in [13], if $J_4 \leq 5$, the center pixel of the window, x, is considered to be a part of a homogeneous region. Otherwise, each pixel is compared with the pixels to its left and above, as shown in Figure 3, and if the absolute difference in the comparison metric is less than a user set threshold, δ, they are considered homogeneous. Since the statistic J_4 shows the degree of dissimilarity of two adjacent regions, the greater the J_4, the lower the probability of homogeneity in the window. Therefore, the intensity threshold, δ, must be decreased adaptively with an increase in J_4. Considering the hardware realisation, the function for the change is determined empiricaly. This is explained as follows:

$$\psi : 6 .. 12 \longrightarrow R$$

and

j_4	6	7	8	9	10	11	12
$\psi(j_4)$	1	1	.5	.25	.25	.25	.25

4 Algorithm Implementation

The algorithm is implemented in three phases. In the first phase called preprocessing, the homogeneity of adjacent pixels in each of the red, green, and blue component is tested.

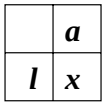


Figure 3: Pixel x is compared with adjacent pixel: l is to the left and a above.

The output from the preprocessing in each component is three signals: *homog, left_merge,* and *above_merge,* followed by the computed colour indices. For example, if the *left_merge* is set, the current pixel can be merged with its left pixel; if the *homog* signal for the current pixel is set, it can be merged with all its adjacent pixels. In the second phase, the output signals from each colour component are combined to produce a unique colour identifier. In the third phase, based on the colour identifiers, regions are grown and labeled. The overall scheme of this algorithm is shown in Figure 9.

4.1 Phase 1: Preprocessing

An *intensity value image* is defined by a function *Int* which maps the image coordinates to intensity values in different colour components.

$$Int \ : \ COORD \longrightarrow \ INTENSITY\ VALUE$$

A *homogeneous image* is defined by a function *homog* which maps the image coordinates to homogeneous signals.

$$homog \ : \ COORD \ \longrightarrow \ HOMOGENEOUS\ SIGNAL$$

A *left_merged image* is defined by a function *left_merge* which maps the image coordinates to left_merge signals.

$$left\_merg \ : \ COORD \ \longrightarrow \ LEFT\_MERGE\ SIGNAL$$

An *above_merged image* is defined as a function *above_merge* which maps the image coordinates to above_merge signals.

$$above\_merge \ : \ COORD \ \longrightarrow \ ABOVE\_MERGE\ SIGNAL$$

For the current pixel *(x : COORD),* the following Pseudo-code algorithm defines phase 1.

```
for (c: c ∈ {red,green,blue}) Do
 {
   Do in parallel
    {
      for (o: o ∈ {four different orientations shown in Figure 2 } )
        Do in parallel
          {
            compute J_co;
          }
      left_dist_c(x) = |Int_c(x) − Int_c(l)|; /* l:COORD*/
      above_dist_c(x) = |Int_c(x) − Int_c(a)|; /* a:COORD*/
    }
   J_4c = max {J_co}
   Do in parallel
    {
     if (j_4c ≤ 5)
       {
         set homog_c (x);
         set left_merge_c(x);
         set above_merge_c(x);
       }
      else
       {
         if (left_dist_c(x) ≤ ψ(J_4c) × δ) set left_merge_c(x);
         if (above_dist_c(x) ≤ ψ(J_4c) × δ) set above_merge_c(x);
       }
 }
```

If $J_{4c} \leq 5$, the current pixel, *x,* can be merged with all its adjacent pixels, therefore, both *left_merge* and *above_merge* signals are set; otherwise, depending on the absolute difference intensity value between the current pixel and its adjacent pixels, either or both of the *left_merge* and *above_merge* signals are set. Moreover, if $J_{4c} \leq 5$, the pixels to its right and below can be merged with the current pixel, *x.* In this case, the *homog (x)* is set and can be used to determine whether the pixels to the right and below the current pixel can be merged with it in the next steps.

4.2 Phase 2: Intersection

In the previous phase, three signals, *homog, left_merge,* and *abovemerge,* for the three colour spaces are produced. In this phase, the relevant signals of each colour space are ANDed to produce the unique merging identifier. For example the *left_merge* identifier is produced by the ANDing of the *left_merge* signals of all three colour. This is shown in Figure 9.

4.3 Phase 3: Region Growing & Labeling

In this phase, regions are grown and pixels belonging to a homogeneous region are assigned a unique label. A labeled image is defined by a function *label* which maps coordinates to

labels.

$$label \ : \ COORD \ \rightarrow \ LABEL$$

For this purpose, each pixel is considered as a region. A 2×2 window as shown in Figure 3, is moved across the image in a raster fashion (left to right and top to bottom). Neighbouring pixels are candidates for merging if the related signals are set. Therefore, a set M_1 is defined containing the adjacent pixels with which the current pixel can merge.

$$M_1 = \{p : (p \in \{a, l\}) \wedge (homog(p) \vee (left\_merge(x) \wedge (p = l)) \vee (above\_merge(x) \wedge (p = a)))\}$$

Due to V shaped regions in the image, region labeling needs to be performed in two passes[11]. In the second pass, the set M_2 is defined as follows:

$$M_2 = \{p : (p \in \{a, l\}) \wedge ((homog(x) \vee (left\_merge(p) \wedge (p = l)) \vee (above\_merge(p) \wedge (p = a)))\}$$

The labeling procedure may be formulated as follows:

$$label(x) = \begin{cases} new\ label & M_1 = \emptyset \wedge pass\ 1 \\ label(x) & M_2 = \emptyset \wedge pass\ 2 \\ min\ (label(p)) & (p \in M_1 \wedge pass\ 1) \vee (p \in M_2 \wedge pass\ 2) \end{cases} \quad (6)$$

For example, in the first pass, each pixel is allowed to merge with its adjacent left pixel, if either its left_merge signal, *left_merge(x),* or the homogeneous signal computed from its left pixel, *homog(l),* is set. The same procedure may be carried out for merging the current pixel with the pixel above. If the current pixel is not allowed to merge with its adjacent pixels, a new label is assigned to it. If it is allowed to merge with one of its adjacent pixels, the label of that pixel is assigned to the current pixel. On the other hand, if it is allowed to merge with both its left and above pixels, the minimum label of those two pixels is assigned to that pixel.

If the regions in the image are limited to V shaped regions in the image, the same procedure of labeling can be used in a reverse fashion (right to left, bottom to up) in the second pass. But, due to snaking regions in the image, the procedure of labeling is not so simple. In this regard, some sequential connected components algorithms are presented. Lumia's algorithm[20] performs nearly identical operations in both passes, as well as a reduced label equivalence table, making it attractive for a hardware implementation. However, the equivalence table in this algorithm must still be processed between the first and second pass of each image row, and there is the added complexity of making forward and reverse passes over the image in a pipelined system. Recently, an alternative approach was developed by Nicol in our Laboratory[25, 26]. This algorithm uses a linear systolic array to modify the labels of recently visited pixels in a raster scan and so removes the need for the label equivalence table used in the Lumia's algorithm. We have adopted the Nicol's algorithm for labeling the regions.

5 Experimental Results

The absence of widely accepted mathematical models for images has made the objective evaluation of segmentation algorithms difficult. Therefore, a subjective visual evaluation is usually presented for the output of the algorithms[11, 27, 3]. Even though some effort has been expended to provide an objective measurement[16], it is believed that the proposed measurement can not be generalised, rather, the amount of useful information provided by the segmentation algorithm to the succeeding stages of analysis is one of the best criteria[3]. Indeed, dealing with a large number of regions (information) at the higher levels of image analysis may be crippling for any system. Therefore, minimizing the number of regions, whilst maintaining expected perceptual groupings in the image, can be a very useful criterion. Another criterion may be the sensitivity of threshold setting, which is a major issue for all segmentation algorithms. Since it is not possible to set a universal threshold for all images, some algorithms provide a number of parameters which may be adjusted to define suitable thresholds for particular image types. However, setting multiple parameters to determine a threshold is particularly difficult. Robust segmentation depends on the relative insensitivity of the threshold to small variations in determining parameters.

For the algorithm presented in this paper, we do not present an objective evaluation for the experimental results. However, there are some points which we emphasize, and we believe that our algorithm has satisfied them. Firstly, this algorithm can be mapped to a real-time, colour image segmentation design. Secondly, there is one threshold parameter, δ, which must be set for this algorithm, which is used to control the intensity uniformity. For a constant threshold, $\delta = 12$, the experimental results are shown in Figures 4-6 without the common postprocessing procedures for removing small regions. The input images in the first two images are 8-bit digitized images, while the third image is a 5-bit digitized image. These results indicate a visually good segmentation in respect to oversegmentation and undersegmentation. The constructed segmented image in Figure 4 is shown in Figure 7. To show the effect of the colour components, with the same method, the grey input of Figure 4 is segmented and the result is shown in Figure 8. As can be seen, most of regions are leaked.

6 Real-time hardware realisation: System overview

The block diagram of the real-time hardware architecture to what the colour segmentation algorithm has been mapped, is shown in Figure 9. To implement this system, a custom VLSI design for the colour preprocessor can be developed. The task of each preprocessor component is to determine the homogeneity of each pixel with the pixel above it and to its left. The output of each preprocessor consists of three signals, *homog, left_merge,* and *above_merge.* The relevant signals are then combined and feed into another custom design for growing and uniquely labeling the regions. The region labels are 14 bits in our system.

To realise this real-time system, each pixel oriented computation needs to be arrayed in a parallel pipelined fashion. The architecture is pipelined, and with a careful design of architecture, logic, and technology, and the exploitation of the parallelism implicit in the algorithm, the high speed requirement of 40ms processing time per image frame (assuming an image with 512×512 resolution at 25 frames/second) can be met.

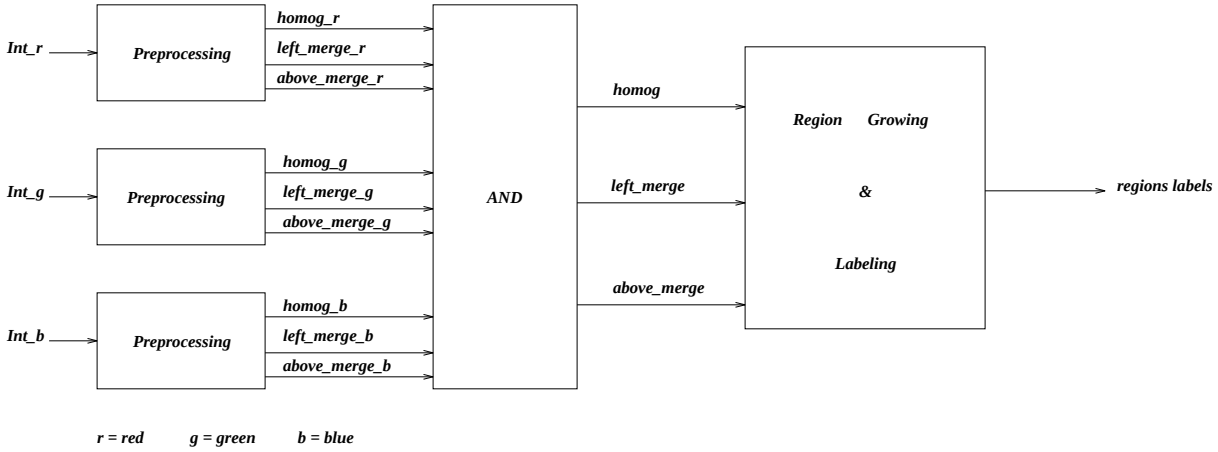


Figure 4: The block diagram of the image segmentation algorithm

6.1 Preprocessing Components

The pixel homogeneity determination algorithm, explained in Phase1 of the algorithm implementation in Section 4, can be mapped into a parallel pipelined architecture as shown in Figure 10. Since a 5×5 window is used to implement this algorithm, four rows of the image are stored in shift registers, shown in the far right in Figure 10. The broken block in Figure 10 computes the Kolmogorov-Smirnov statistic J, and the rest of the design measures the intensity uniformity of adjacent pixels, and combines these to produce the three signals, *homog, left_merge,* and *above_merge,* by using the computed statistic J_4.

Since the algorithm can be implemented completely in a pipelined manner, the floor plan shown in Figure 10 is straight forward. The output of each stage is directly connected to the input of the next stage, which reduces significantly delays. Each slice of the pipeline is repeated which increases the regularity of the design. The basic operations of each module involve mainly addition, subtraction, comparison, and logic decision. Consequently, the hardware realization is straightforward and is described below:

The Range & θ Finder module: At each pixel time (t_0), the procedure of computing the statistic J begins within the 5×5 window centered at pixel C_{22}, where $C_{ij}, i = 0, ..., 4, j = 0, ..., 4$, denotes each cell in the window. The first step in this algorithm is to calculate the range of data within the 5×5 window, and thereby calculating the set Θ in Equation (2). To achieve video rate edge detection, θ needs to be ready at time t_0, so, the computation of the range needs to be started before t_0. The block named Range & Θ Finder is located at the top of Figure 10. Computing the range requires the comparison of two 8-bit digits in each pixel time. At time t_{-1}, in addition to the range, the maximum and minimum observations are stored in the range, max and min registers respectively, and the set $\Theta = \theta_0,\ \theta_1,\ \theta_2$ is computed as follows:

$$t_{-1} = \begin{cases} \theta_0 = Range \gg 2 + min; \\ \theta_1 = Range \gg 1 + min; \\ \theta_2 = max - Range >> 2. \end{cases} \tag{7}$$

where the notation " $\gg i$" means shift right i times.

The Comparator module: In this block, all observations within the 5×5 window, at

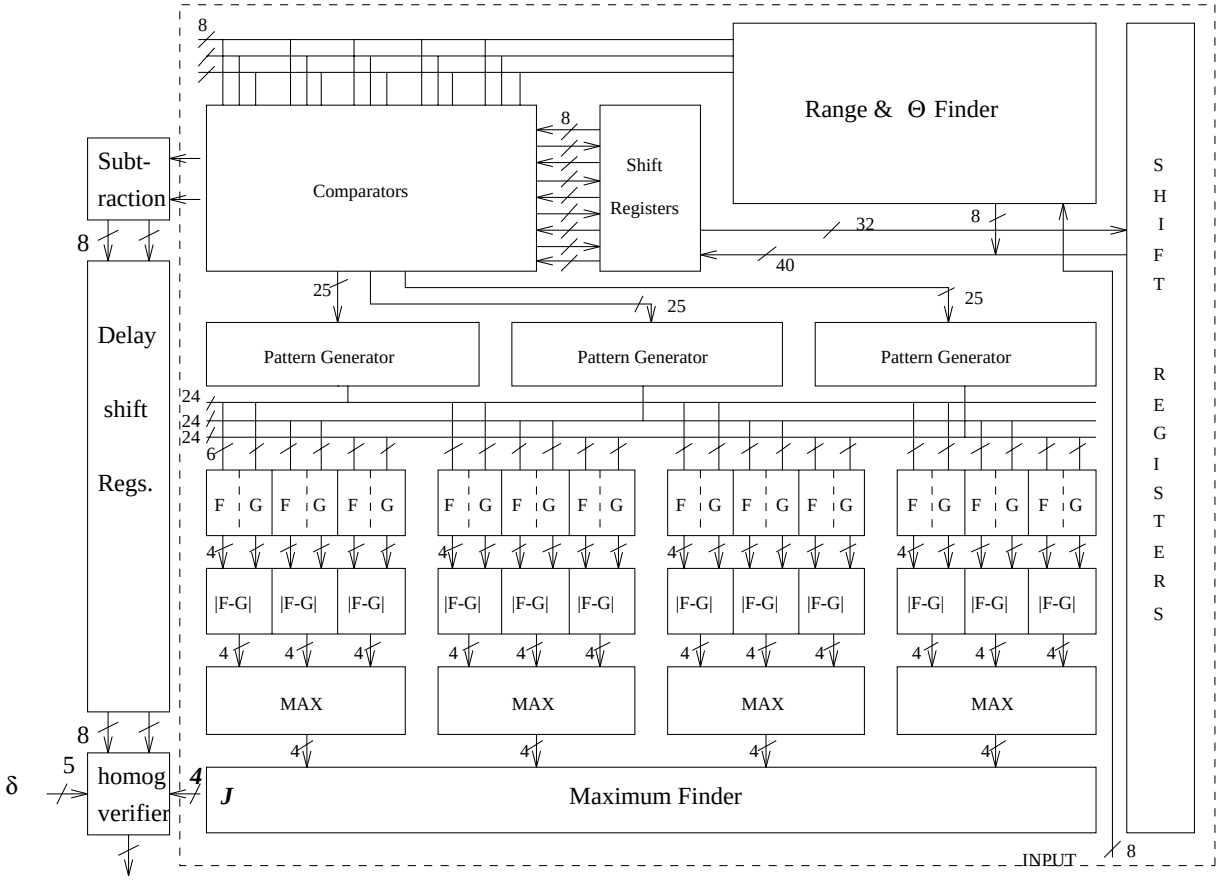


Figure 5: The floor plan of the preprocessing component

time t_0, are compared with each element of θ in parallel, and for each cell, three outputs are generated labeled γ_{ijk}, where $i = 0, ..., 4, j = 0, ..., 4$, and $k = 0, 1, 2$, and the γ_{ijk} are defined as follows:

$$\gamma_{ijk} = \begin{cases} 1 & C_{ij} < \theta_k. \\ 2 & \text{otherwise.} \end{cases} \tag{8}$$

The output of this stage consists of 25 buses each of three bits wide.

The Pattern Generation module: By determining γ_{ijk} in the previous module, to compute $F_m(\theta_k)$ and $G_n(\theta_k)$ from Equations (4) and (5) respectively, the number of γ_{ijk}'s set for the two samples, X and Y, are counted (added) in parallel to determine how many pixels in each sample are less than the θ_k, where $k = 0, 1, 2$. To compute $F_m(\theta_k)$ and $G_n(\theta_k)$, for each orientation shown in Figure 2, the number of operations can be reduced by using the common patterns used to compute the above two functions, for each orientation. To achieve this, based on sub-patterns, each consisting of three pixels as shown in Figure 11 (a), the four patterns found commonly in all partitions shown in Figure 2, are constructed. These patterns are shown in Figure 2 $(b) - (e)$

To compute F and G in Equations (4) and (5), the appropriate patterns, each being of three bits wide, are added to produce a 4-bit output. After this stage, computing J in Equation (3) is limited to 4-bit operations for addition, subtraction and comparison.

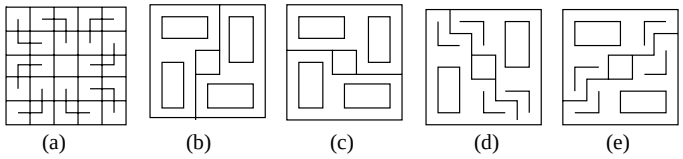


Figure 6: The four patterns used commonly in all partitions

6.2 VLSI Design

The design was undertaken in two parts. First, this algorithm was described using a hardware description language, named MODAL[12, 15], and simulated with an event-driven logic simulator, which demonstrated correct functionality and that the algorithm will run with a 100 MHz clock cycle, independent of the wiring delays and other fabrication issues. The longest delay in this model corresponded to the 8-bit adder/comparator modules, and when real delays are considered, it is clear that the speed estimate is optimistic.

In the second phase, the worst case module (the 8-bit adder/comparator) was designed and simulated. This is the critical timing element which determines the pipeline speed. The 8-bit adder/comparator was realised using a domino 4-bit carry look ahead circuit. The 8-bit comparator is realised by cascading two 4-bit domino carry look ahead circuits. The output carry of the second stage is interpreted as the greater than or equal to ($\geq$) signal. The floor plan of the comparator is shown in Figure 12. This comparator was designed and simulated using a double metal, 2μ, n-well CMOS technology. The result of the analog simulation shows a worst case 10ns computation time. The geometric layout is shown in Figure 13.

6.3 Investigation

To achieve the preprocessing for an image with a 512×512 resolution, being refreshed at 25 frames per second in real time requires that each pixel be processed within 150ns, dictating an overall frequency of 7MHz for the design. Simulation indicates the design will easily meet this target. Migrating the design to a state-of-the-art sub-micron technology will provide considerable further speed enhancement enabling the processing of images as large as 1000×1000 at 60MHz.

6.4 Region Growing and Labeling

The input to this component consists of three signals, *homog(x)*, *left_merge(x)*, and *above_merge(x)*, where each signal indicates the potential of the merging for the current pixel. According to the region growing and labeling principles explained as Phase 3 of the algorithm, the schematic diagram shown in Figure 14 is the heart of the design, and can easily perform in real-time. Assuming that a maximum of 2^{14} regions occur in the segmented image, the worst delay in this subsystem occurs in the comparator module which compare two 14-bit labels.

As explained before, labeling of the connected regions needs to be run in two pass. For this purpose, a custom VLSI design was developed in our Laboratory by Nicol[24]. His approach has the advantage that all the label equivalence computations are localised in each cell in a systolic array, and therefore, the speed of operation is limited only

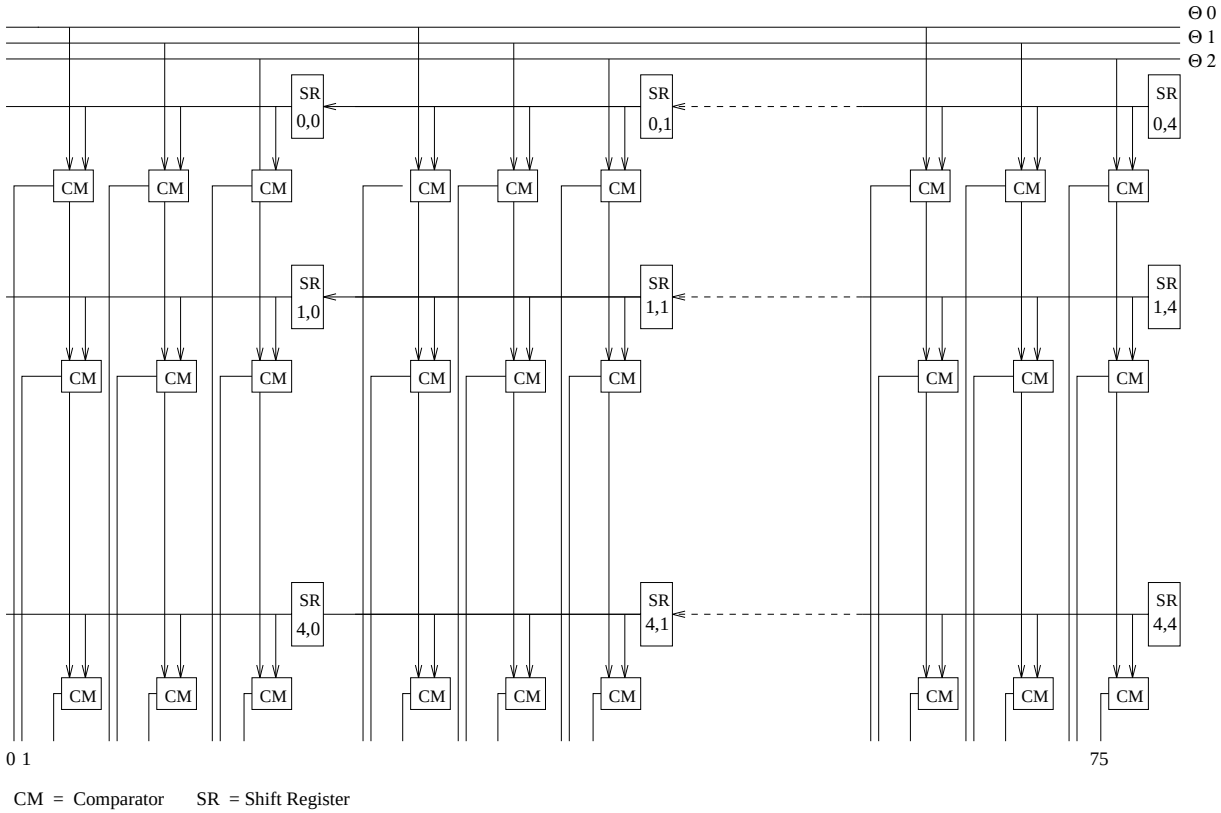


Figure 7: The floor planning of the comparator module

by communication delay between adjacent cells. Nicol's implementation uniquely labels 512×512 images at 50 frames/sec. In the second pass, the same labeling procedure is run, except that the image is scanned from bottom to top and right to left, therefore, the pixel output from the first pass must be stored in a *LIFO* stack, to feed the second pass.

Our labeling algorithm is somewhat faster than Nicols's since the the multi-valued (14 bit) comparisons needed to determine whether the current pixel can merge with its adjacent pixels has already been carried out in the preprocessing phase of our algorithm.

7 Conclusion

In this paper, based on the region growing approach, a new pixel stream oriented segmentation algorithm for colour images suitable for mapping to a real-time hardware design is presented. For this purpose, two properties of visualized signals, statistical distribution and intensity uniformity, in each colour space is used. First, within a 5×5 window, using the Kolmogorov-Smirnov test, the strength of homogeneity is tested. Pixels whose homogeneity is sufficiently high are grown without considering their intensity uniformity. Pixels with the lower homogeneity values are merged with their adjacent pixels, if their intensities are sufficiently similar. The result of the separate colour space computations are combined to produce the merging signals for region growing and labeling.

Different experimental results show the effectiveness of this algorithm. Moreover, it is shown that this algorithm is robust in its applications to various images and can be mapped to a real-time hardware design.

This algorithm lends itself to a parallel pipelined implementation and no complex

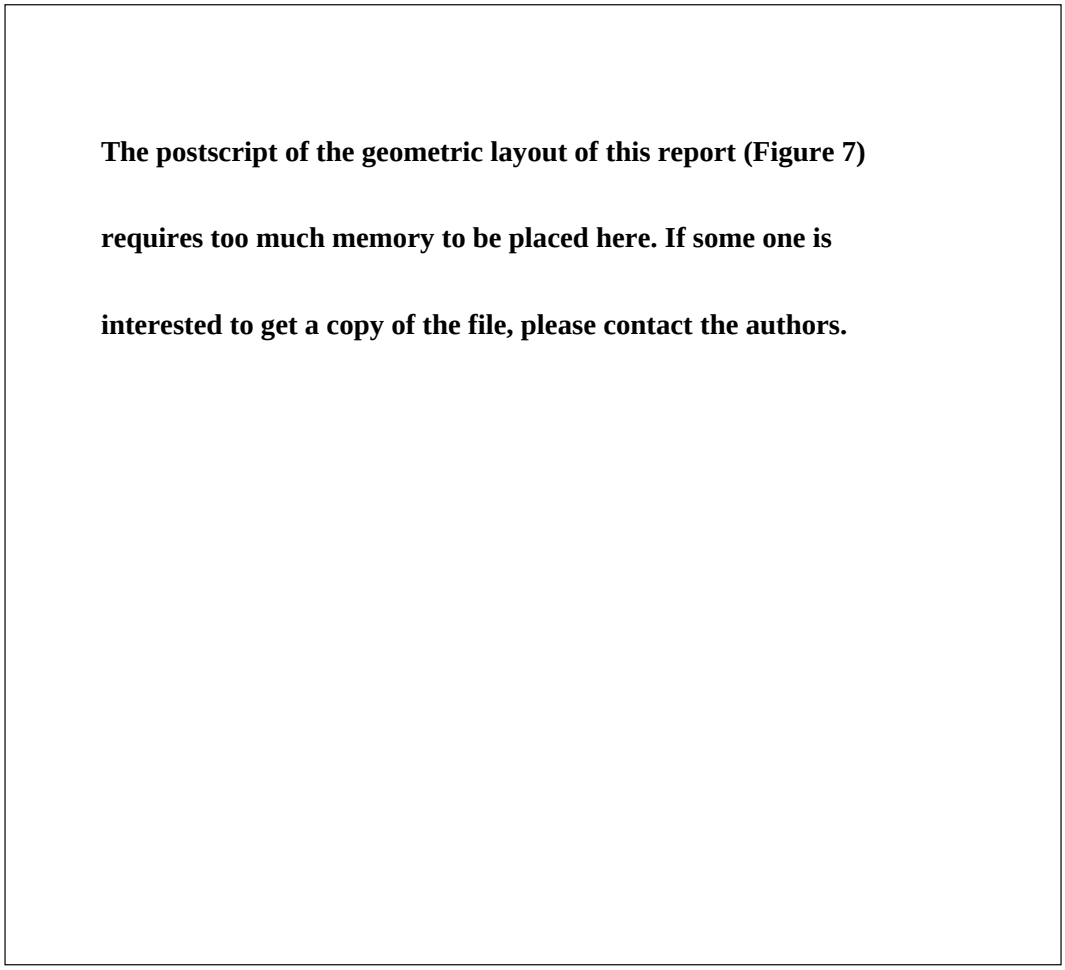


Figure 8: The geometric layout of the comparator module

operations are needed. The floor plan of the preprocessing is presented and simulation verifies its real-time performance. The worst case timing in this component is due to the adder/comparator module, for which analog simulations of these modules, implemented using a two phase double metal, 2μ CMOS technology, demonstrate a computation time of less than 10ns. Accounting for realistic wiring delays and other fabrication minutiae, we estimate the design can be realised at the video rate, for 512×512 images being displayed at 25 frames per second. The performance characterisation of the region growing and labeling component has been investigated, and its performance will easily meet the real-time requirements.

Acknowledgement
We would like to thank A. Saee for his comments for the statistical part of this paper and also C. Nicol for his comments on the region growing and labeling part.

References

[1] S. Basu. Image segmentation by semantic method. *Pattern Recognition*, 20:497–511, 1987.

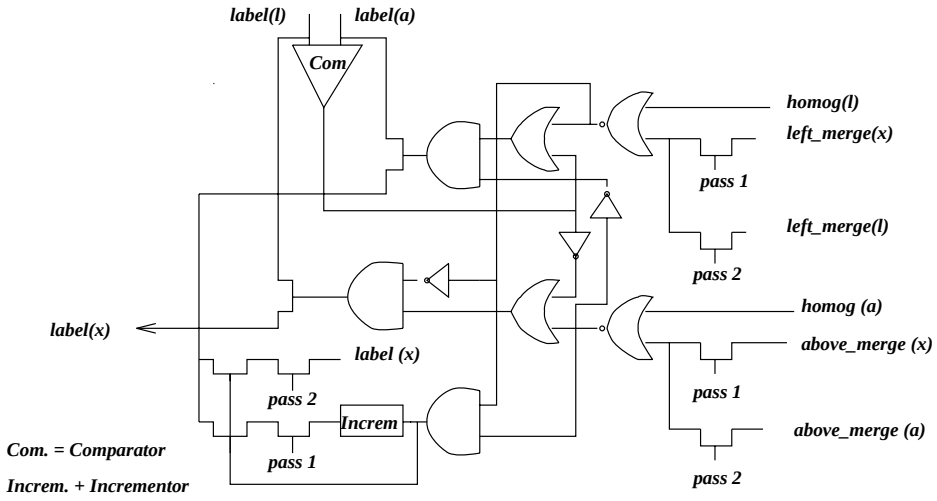


Figure 9: The schematic diagram of the label generating module

[2] P.J. Besl and R.C. Jain. Segmentation through variable-order surface fitting. *IEEE Trans. PAMI*, PAMI-10:167–192, 1988.

[3] J.R. Beveridge, J. Griffith, R.R. Kohler, A.R. Hanson, and E.M. Riseman. Segmentation images using localized histograms and region merging. *Int. j. of Computer Vision*, 2:311–347, 1989.

[4] B. Bhanu and O.D. Faaugeras. Segmentation of images having unimodal distribution. *PAMI*, PAMI-3:408, 1982.

[5] B. Bhanu, B.L. Hutchings, and K.F. Smith. Vlsi design and implementation of a real-time image segmentation processor. *Machine Vision and Application*, 3:21–44, 1990.

[6] B. Bhanu and B. Parvin. Segmentation of natural scenes. *Pattern Recognition*, 20:487, 1987.

[7] M. Celenk. A recursive clustering technique for color picture segmentation. *Proc CVPR'88*, pages 437–444, 1988.

[8] J.N. Gupta, R.L. Kettig, D.A. Landgrebe, and P.A. Wintz. Machine boundary finding and sample classification of remotely sensed agricultural data. *Machine Processing of Remotely Sensed Data*, pages 4B–25–4B–35, 1973.

[9] R.M. Haralick. Edge and region analysis for digital image data. *CGIP*, 12:60–73, 1980.

[10] R.M Haralick and I. Dinstein. A spatial clustering procedure for multi-image data. *IEEE Trans. Circuits and Systems*, CAS-22, 1975.

[11] R.M. Haralick and L.G. Shapiro. Image segmentation techniques. *CVGIP*, 29:100–132, 1985.

[12] G.R. Hellestrand. Modal: a system for digital hardware description and simulation. *j. Digital Systems*, pages 241–303, 1979.

[13] M. Hollander and D.A. Wolfe. *Nonparametric Statistical Methods*. Jhon-Wiely, 1973.

[14] S. Horowitz and T. Pavlidis. Picture segmentation by a directed split-and-merge procedure. *Proc. IJCPR-2*, pages 424–434, 1974.

[15] M.C. Kam. Tutorial on using the modal compiler and simulation system. *Internal Document, VLSI and Systems Technology Laboratory, UNSW.*, 1988.

[16] M.D. levine and A.M. Nazif. Dynamic measurements of computer generated image segmentations. *IEEE Trans. PAMI*, PAMI-7:155–164, 1985.

[17] M.D. Levine and S.I. Shaheen. A modular computer vision system for picture segmentation and interpretation. *IEEE Trans. PAMI*, PAMI-3:540–556, 1981.

[18] S. Liou, A.H. Chiu, and R.C. Jain. A parallel technique for signal-level perceptual organization. *IEEE Trans. PAMI*, PAMI-13:317–324, 1991.

[19] M. Loeve. *Probability Theory*. Van Nostrand, 1963.

[20] R. Lumia, L. Shapiro, and O. Zuniga. A new connected components algorithm for virtual memory computers. *CVGIP*, 22:287–300, 1983.

[21] J.L. Muerle. Some thoughts on texture discrimination by computer. *Picture Processing and Psychopictorics, ed., B.S.Lipkin and A. Rosenfeld*, pages 371–379, 1970.

[22] J.L. Muerle and D.C. Allen. Experimental evaluation of techniques for automatic segmentation of objects in a complex scene. *In Pictorial Pattern Recognition (G.C. Cheng et al., Eds.)*, pages 3–13, 1968.

[23] P.T. Nguyen. Image segmentation on color and texture gradient. *vision interface'86*, pages 267–272, 1986.

[24] C.J. Nicol. Design of a connected component labelling chip for real time image processing. *IEEE APCAS'92*, pages 142–147, 1992.

[25] C.J. Nicol. Vlsi for labelling the connected components of multi-valued images in real time. *Technical Report, SCSE*, 1992.

[26] C.J. Nicol. A systolic architecture for labeling the connected components of multi-valued images in real time. *Proc. CVPR'93*, pages 136–141, 1993.

[27] Y. Ohta. *Knowledge-Based Interpretation of Outdoor Natural Color Scenes*. Pitman, 1985.

[28] Y. Ohta, T. Kanade, and T. Sakai. Color information for region segmentation. *CGIP*, 13:222–241, 1980.

[29] T. Pavlidis and Y. Liow. Integrating methodologies in image analysis. *Computer vision and shape recognition ed. A.Krzyzak et al.*, pages 429–437, 1989.

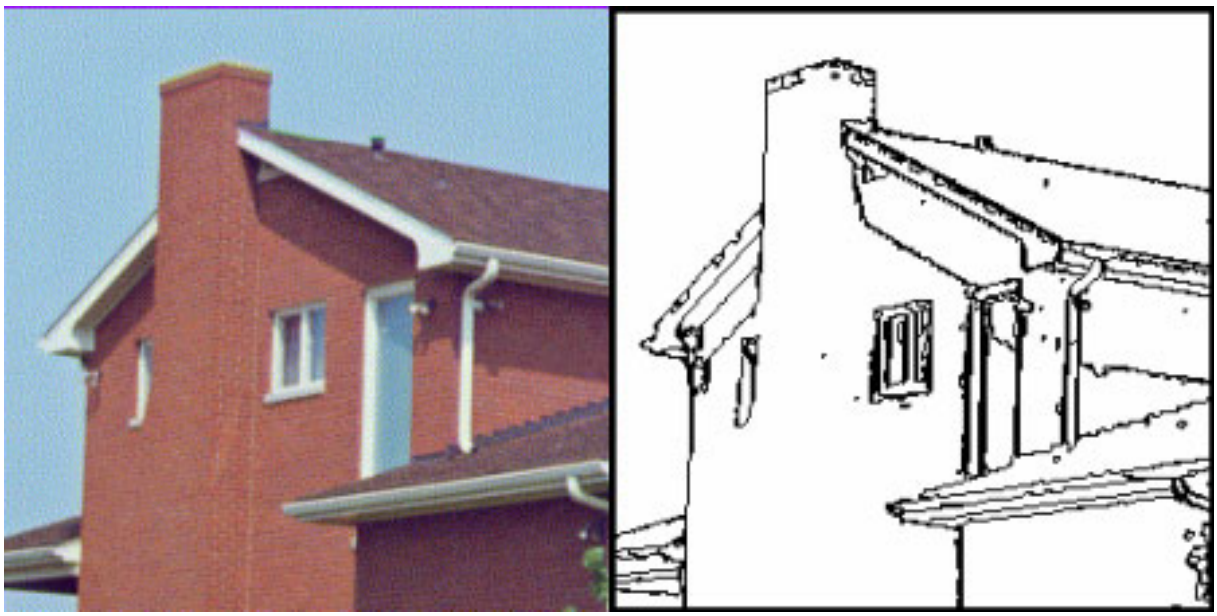
[30] T. Pong, L.G. Shapiro, L.T. Watson, and R.M. Haralick. Experiments in segmentation using a facet model region grower. *CVGIP*, 25:1–23, 1984.

[31] R.H. Randles and D.A. Wolfe. *Introduction to the Theory of Nonparametric Statistics*. Jhon-Wiely, 1979.

[32] E.M. Riseman and M.A. Arbib. Computational techniques in the visual segmentation of static scenes. *CGIP*, 6:221–270, 1977.

[33] Y. Yakimovsky. Boundary and object detection in real world images. *j. ACM*, 23:599–618, 1976.

[34] J.H. Zar. *Biostatistical Analysis*. Prentice-Hall, 1974.

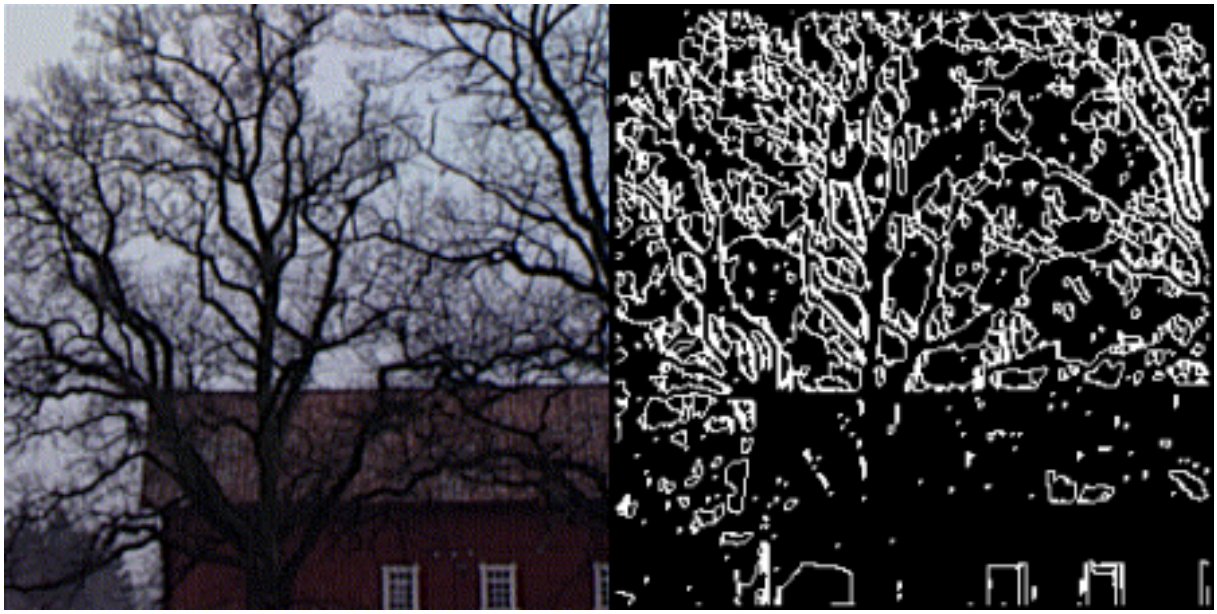


(a) *(b)*

Figure 10: *(a)* The original image. *(b)* The result of segmentation algorithm

(a) *(b)*

Figure 11: *(a)* The original image. *(b)* The result of segmentation algorithm



(a) *(b)*

Figure 12: *(a)* The original image. *(b)* The result of segmentation algorithm

Figure 13: The reconstruction of the input image after the segmentation

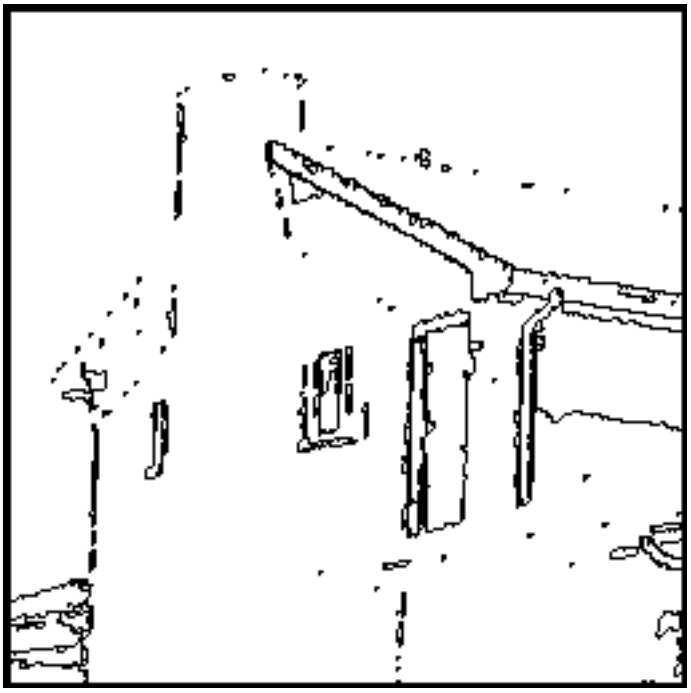


Figure 14: The result of image segmentation for the input grey scaled image