Segmentation Techniques For Video Coding

SANLAVILLE Rémy

Traineeship of 2nd year at VSSP Group
Vision Speech and Signal Processing
University of Surrey Guildford

3^{rd} of June to 3^{rd} of September 1996
under the supervision of Dr Miroslaw Bober

Acknowledgements

I will try to find the words to express my gratitude and appreciation for this marvellous experience.

This traineeship has meant a lot more to me than just a research period, it has permitted me to discover another country, language and different cultures. It will play an important role in my future.

I would like to express my gratitude to the V.S.S.P. group for their welcoming reception and all fellow members, who advised and supported me throughout. Specials thanks to my fellow colleges Filiberto, Ratna and Sanja.

I lack the words to express my heart-felt appreciation to my supervisor who despite the fact that he was constantly busy, he always took the time to follow, guide and support me throughout my research. I thank him for his confidence and trust. I hope there will be another occasion to work with him some time in the future.

I would like to thank all the people in Guildford who befriended me and who have made my stay here a pleasant and enjoyable experience, particularly Shuaimi, Sarah and Sydney.

I will hold these past three months with pleasant and fond memories for a long time to come. It has proved to be a valuable enriching experience in my life.

Abstract

A presentation of several approaches concerning the segmentation technique for video coding is made. These approaches include histogram analysis, clustering and quantization methods for gray and colour images. A compression algorithm (2D CCITT Standard) is introduced and extended to gray-level values. Some of these experiments are presented including a comparison with JPEG.

Key words : Segmentation, Histogram analysis, Thresholding, Clustering and Quantization algorithms, Compression, 2D CCITT Standard algorithm.

Contents

Chapter 1

Introduction

I have undertaken my research project with the Vision Speech and Signal Processing Group (V.S.S.P.). The research activities of the Group focus in the are of image processing and computer vision, image representation (shape, texture, colour), motion analysis, iconic search in image databases, pattern recognition and neural network. The new research areas now also include image compression, coding and document processing.

The research projects carried out can be grouped into seven major categories :

1. image processing
2. image understanding
3. industrial automation
4. robot vision
5. pattern recognition and neural net systems
6. signal analysis
7. image communication and multimedia

My work is a part of the last category. I have worked with the Image coding and Multimedia team, under the supervision of Dr Miroslaw Bober.

1.1 Segmentation techniques for video coding

The purpose of this research programme is to investigate and develop methodologies and algorithms for the processing of images for the purpose of efficient compression and coding.

Description :

Image analysis and automatic vision relies heavily on the information extracted from the image at the image processing stage. This does not only apply to static images but sequences of images as well. Examples of applications of Image Processing are image compression and image segmentation into regions characterised by some uniformity criterion and the extraction of image features. The program of the V.S.S.P. Group in this area of research includes the development and investigation of novel approaches concerned with texture analysis and segmentation, efficient motion coding and image labelling. In the context of the above applications some generic tools have been developed applicable to a large class of problems. This includes tools for motion estimation and segmentation, image segmentation based on colours and texture, multiresolution image processing etc.

1.2 Purpose of my work

The objective of the research was to investigate and to implement various segmentation techniques with the emphasis on an application to image compression and coding.

Segmentation techniques are used for region-based coding and also as a preprocessing for motion analysis.

A technique based on histogram analysis [6] was implemented and tested. In order to improve the performance, the following modifications were introduced :

- A new algorithm to find peaks in the histogram.
- A new approach concerning the determination for the best quantization level.

A novel element of research involved a design and implementation of a still image compression method. It involves segmentation of the image into regions of coherent colour and encoding the segmentation map for each colours using the 2D CCITT Standard for encoding of 2D binary images with multi-valued images. I had to extend this coding algorithm to cope.

The algorithms will be used by the Image Coding Team at the V.S.S.P. Group for further research (for example as a segmentation stage for region based coding).

Chapter 2

Segmentation Techniques

Segmentation refers to processes that partition the two-dimensional image into connected area (called regions) composed of group of pixels that are uniform in some property, such as gray level (intensity, brightness), colour or texture. Several approaches are used but they can be divided into techniques involving edge detection and clustering techniques.

Although edge detection is often used with several methods (Gradient, Roberts Operator, Sobel Operator, Prewitt Operator, Laplacien Operator) and continues to be an active research area, I worked with the regions method and particularly with techniques involving histogram analysis.

2.1 Region-based segmentation

In the region-based approach, all pixels that correspond to an object are grouped together and are marked to indicate that they belong to one region. Pixels are assigned to regions using some criterion that distinguishes them from the rest of the image. Two very important principles in segmentation are value similarity and spatial proximity. Two pixels may be assigned to the same region if they have similar intensity characteristics or if they are close to one another. For example, a specific measure of value similarity between two pixels is the difference between the gray or colour values, and a specific measure of spatial proximity is Euclidean distance. The variance of gray or colour values in a region and the compactness of a region can also be used as measures of value similarity and spatial proximity of pixels within a region, respectively.

The principles of similarity and proximity come from the assumption that points on the same object will project to pixels in the image that are spatially close and have similar gray or colour values. Clearly, this assumption is not satisfied in all situations. But in simple situations, segmentation can be done with thresholding and component labelling [1]. Complex image may require more sophisticated techniques than thresholding to assign pixels to regions that correspond to parts of objects. Thus, we will attach particularly to the clustering algorithm method [6].

2.2 Definition

Formally, region-based segmentation can be defined as a method to partition an image $\mathcal{I}$ into subimages, called regions, $R_1 \cdots R_n$ such that each subimages is an object candidate.

More exactly, given a set of image pixels $\mathcal{I}$ and a homogeneity predicate P(.), the region-based segmentation method will find a partition S of the image $\mathcal{I}$ into a set of n regions R_i such that :

- $\bigcup_{i=1}^{n} R_i = \mathcal{I}$
- P(R_i)=True (the homogeneity predicate and partitioning of the image have the properties that any region satisfies the predicate P(.))
- P($R_i \cup R_j$)=False ($i \neq j$) (for all i and any two adjacent regions cannot be merged into a single region that satisfies the predicate P(.)).

The homogeneity predicate P(.) defines the conformity of all points in the region R_i to the region model.

The process of converting a gray value image into a binary image is a simple form of segmentation where the image is partitioned into two sets. To do this, we can use the thresholding algorithm discussed below.

2.3 Thresholding

Thresholding is a method to convert a gray scale image into a binary image so that objects of interest are separated from the background. For thresholding to be affective in object-background separation, it is necessary that the objects and background have sufficient contrast and that we know the intensity levels of either the objects or the background. In a fixed thresholding scheme, these intensity characteristics determine the value of the threshold.

To make segmentation more robust, the threshold should be automatically selected by the system. Knowledge about the objects in the scene, the application, and the environment should be used in the segmentation algorithm in a form more general than a fixed threshold value. Such knowledge may include :

- Intensity characteristics of objects
- Sizes of the objects
- Fractions of an image occupied by the objects
- Number of different types of objects appearing in an image

A thresholding scheme that uses such knowledge and selects a proper threshold value for each image without human intervention is called an automatic thresholding scheme. Automatic thresholding analyses the gray values, and uses the

knowledge about the application to select the most appropriate threshold. Since the knowledge employed in these schemes is more general, the domain of applicability of the algorithm is increased.

Suppose that an image contains n objects $O_1, O_2, \cdots, O_n$, including the background, and gray values from different populations $\pi_1, \cdots, \pi_n$ with probability distributions $p_1(z), \cdots, p_n(z)$. In many applications, the probabilities $P_1, \cdots, P_n$ of the objects appearing in an image may also be known. Using this knowledge, it is possible to rigorously formulate the threshold selection problem. Since the illumination geometry of a scene controls the probability distribution of intensity values $p_i(z)$ in an image, one cannot usually precompute the threshold values. As we will see, most methods for automatic threshold selection use the size and probability of occurrence and estimate intensity distributions by computing histograms of the image intensities.

In discussing thresholds, this allows us to say that gray values below a certain threshold belong to the object and gray values above the threshold are from the background.

2.4 Algorithms

We, now, present and discuss about several thresholding algorithm and method for a gray-level value based segmentation.

2.4.1 P-Tile Method

The p-tile method uses knowledge about the area or size of the desired object to threshold image. Suppose that in a given application objects occupy about p percent of the image area. By using this knowledge to partition the gray value histogram of the input image, one or more thresholds can be chosen that assign p percent of the pixels to the object. An example of a binary image formed using this technique is given on Figure 1.

It is the most simple method and clearly it is very limited and not really used.

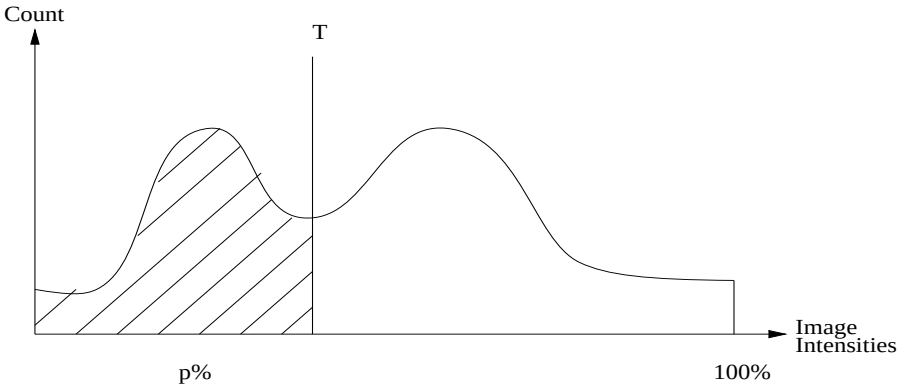

Figure 1 : An example of the p-tile method

2.4.2 Mode Method

If the objects in an image have the same gray value, the background has a different gray value, and the image pixels are not affected by a Gaussian noise, then we may assume that gray values are drawn from two normal distributions with parameters (μ_1, σ_1) and (μ_2, σ_2). The histogram for an image will them show two separate peaks. In the ideal case of constant intensity values, $\sigma_1 = \sigma_2 = 0$, there will be two spikes in the histogram and the threshold can be placed anywhere between the spikes, as shown in Figure 2. In practice, the two peaks are not so well separated (see Figure 3). In this case, we may detect peaks and valleys in the histogram, and the threshold may be set to the pixel value corresponding to the valley.

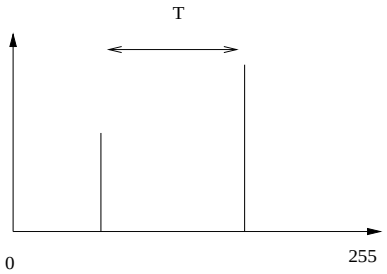

Figure 2 : Ideal intensity values $\sigma_1 = \sigma_2 = 0$

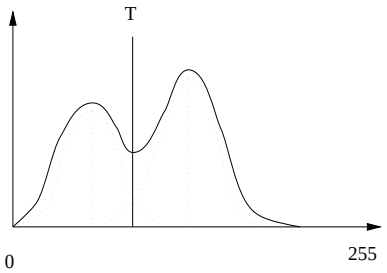

Figure 3 : In practice, peaks are not so well separated

The determination of peaks and valleys is a nontrivial problem (especially with colour images, where histogram is defined in a 3D space), but many methods have been proposed to solve it. For an automatic thresholding scheme, we should have a measure of the peakiness and valleyness of a point in a histogram. A computationally efficient method is given in Algorithm 1. This method ignores local peaks by considering peaks that are at some minimum distance apart. The peakiness is based on the height of the peaks and the depth of the valleys; the distance between the peaks and valleys is ignored.

1. *Find the two highest local maxima in the histogram that are at some minimum distance apart. Suppose these occur at gray values g_i and g_j.*
2. *Find the lowest point g_k in the histogram H between g_i and g_j.*
3. *Find the peakiness, defined as $min(H(g_i), H(g_j))/H(g_k)$.*
4. *Use the combination (g_i, g_j, g_k) with highest peakiness to threshold the image. The value g_k is a good threshold to separate object corresponding to g_i and g_j.*

Algorithm 1 : Peakiness Detection for Appropriate Threshold Selection

This approach can be generalized to images containing many objects with different mean gray values. Suppose there are n objects with normally distributed gray values with parameters $(\mu_1, \sigma_1), (\mu_2, \sigma_2), \cdots, (\mu_n, \sigma_n)$, and the back-

ground is also normally distributed with parameters (μ_0, σ_0). If the means are significantly different, the variances are small, and none of the objects is very small in size, then the histogram for the image will contain $n+1$ peaks. The valley locations $T_1, T_2, \cdots, T_n$ can be determined, and pixels with gray values in each interval $(T_i, T_{i+1}]$ can be assigned to the corresponding object (see Figure 4).

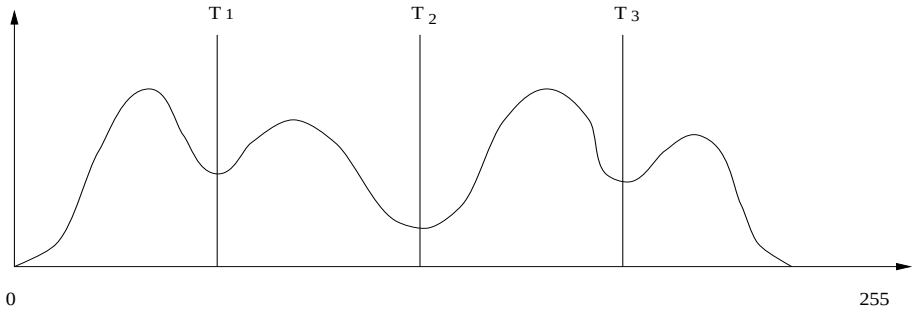

Figure 4 : Histogram for an image containing several objects with differing intensity values

This approach can be more robust and efficient by using a Gaussian filtering applied to the histogram, before running the algorithm detecting peaks. This removes the noise in the histogram and preserves the important information on the position of the clusters.

2.4.3 Iterative Threshold Selection

An iterative threshold selection method starts with an approximate threshold and then successively refines this estimate. It is expected that some property of subimages resulting from the threshold can be used to select a new threshold value that will partition the image better than the first threshold. The method is given in Algorithm 2.

1. *Select an initial estimate of the threshold T. A good initial value is the average intensity of the image.*
2. *Partition the image into two groups, R_1 and R_2, using the threshold T.*
3. *Calculate the mean gray value μ_1 and μ_2 of the partitions R_1 and R_2.*
4. *Select a new threshold :*
$$T = \frac{1}{2}(\mu_1 + \mu_2).$$
5. *Repeat steps 2-4 until the mean values μ_1 and μ_2 in successive iterations do not change.*

Algorithm 2 : Iterative Threshold Selection

2.4.4 Adaptative Thresholding

If the illumination in a scene is uneven, then the above automatic thresholding schemes may not suitable. The uneven illumination may be due to shadows or due to the direction of illumination. In all such cases, the same threshold value may not be usable throughout the complete image. Adaptative methods analyse a histogram of the entire image.

Methods to deal with uneven illumination or uneven distribution of gray values in the background should look at a small region of an image and then analyse this subimage to obtain a threshold for only that subimage. Some techniques have been developed to deal with this kind of situation.

A straightforward approach to segment such images is to partition the image into $m \times m$ subimages and select a threshold T_{ij} for each subimage based on the histogram of the ijth subimage ($1 \leq i, j \leq m$). The final segmentation of the image is the union of the regions of its subimages. The results of this method are shown in Figure 5.

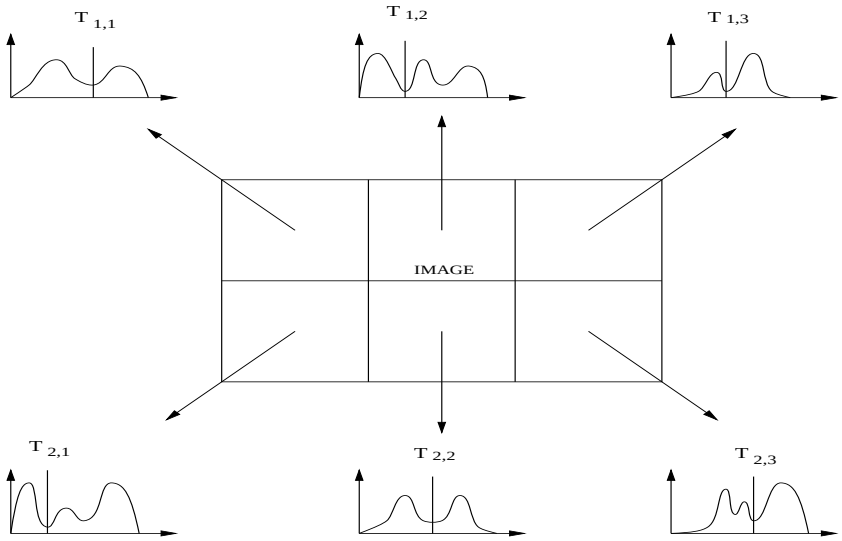

Figure 5 : The process of adaptative thresholding on an image which is not amenable to regular thresholding

2.4.5 Variable Thresholding

Another useful technique in the case of uneven illumination is to approximate the intensity values of the image by a simple function such as a plane or biquadratic. The function fit is determined in large part by the gray value of the background. Histogramming and thresholding can be done relative to the base level determined by the fitted function. This techniques is also called background normalization.

2.4.6 Double Threshold

In many applications, it is known that certain gray values belong to objects. However, there may be additional gray values that belong to either objects or the background. In such a case, one may use a conservative threshold T_1 to obtain the core of the object and then use some method to grow the object regions. The methods used for growing these regions will depend on the specific application. Common approaches include using another threshold to accept pixels if they have a neighbour that is a core pixel or by using intensity characteristics, such as a histogram, to determine points to be included in the object region. A simple approach is to accept all points that are below a second threshold T_2 and are connected to the original set of points. This approach is outlined in Algorithm 3.

1. *Select two thresholds T_1 and T_2.*
2. *Partition the image into three regions : R_1, containing all pixels with gray values below T_1; R_2, containing all pixels with gray values between T_1 and T_2, inclusive; and R_3, containing all pixels with gray values above T_2.*
3. *Visit each pixel assigned to region R_2. If the pixel has a neighbour in region R_1, then reassign the pixel to region R_1.*
4. *Repeat step 3 until no pixels are reassigned.*
5. *Reassign any pixels left in region R_2 to region R_3.*

Algorithm 3 : Double Thresholding for Region Growing

2.5 Conclusion

As discussed above, one may use information in the histogram of an image to select an appropriate threshold for segmentation. This approach is useful in those applications where objects have constant gray-level values. If the illumination is different in different part of a scene, then a single threshold may not be sufficient to segment the image, even if the image contains only one object. In such cases one must use techniques that effectively partition an image arbitrary, and select thresholds for each subimage independently. If the images are complex, these approaches will also perform poorly.

The most basic limitation of the histogram-based approaches, that we see above, is due to the fact that a histogram throws away spatial information about the intensity values in an image. The histogram describes the global intensity distribution. Several images with very different spatial distributions of gray values may have similar histograms. It does not exploit the important fact that points from same object are usually close due to surface coherence.

We presented several method for the histogram-based segmentation of gray-level images. By extending these methods, we can use them for colour images.

But it is neither simple nor efficient. Thus, we will present a new method called clustering algorithm in the next chapter, including also the notion of quantization.

Chapter 3

Clustering and Quantization Algorithms

We now discuss about one method called cluster analysis to segment an image with colour values. There are a wide variety of image segmentation techniques [1] [2], but feature space clustering is one of the most popular of these methods [3].

In a general segmentation task, very little a priori knowledge about the underlying structure of the extracted feature data from the image is available. Information such as the number of clusters in the image (corresponding to the number of clusters in the feature space) and the form of probability density of features are not known. Thus, truly non-parametric clustering techniques need to be utilized.

The literature on cluster analysis is both voluminous and diverse [4] [5] provide excellent surveys of the existing techniques. But many are limited by the fact they are not totally non-parametric and automatic. Thus we used one which have these characteristiques [6] using the quantization notion. Moreover, we introduced some improvments and new ideas.

Concerning the colour representation, we worked with two different models, namely RGB and YUV [7] [8].

Before presenting the clustering method, we will introduce the notion of histogram generation in the multidimensional space.

3.1 Histogram generation in the multidimensional space

The first step toward the generation of the histogram is the quantization of the L-dimensional feature space into non-overlapping cells. As a result, each cell will encompass a certain number of data points (feature vectors) and the histogram is generated by counting the number of vectors falling into each cell. Since the dynamic range of the vectors in each dimension can be quite different, the cell size for each dimension should be different. Hence, the cells will be hyperboxes. Each cell is then represented by an L-dimensional index,

$C(d_1, d_2, \cdots, d_L)$, where d_k is an integer, $d_k \in \{1, 2, \cdots, Q\}$ and Q being the total number of quantization levels. Hence, the kth side of the hyperbox denoted by $CS(k)$ can be expressed as

$$CS(k) = \frac{f_{max}(k) - f_{min}(k)}{Q} \quad k = 1, 2, \cdots, L$$

where $f_{max}(k)$ and $f_{min}(k)$ stand for the maximum and the minimum values of the kth dimension. The labelling convention of the cell indices for a two-dimensional space is illustrated in Figure 6

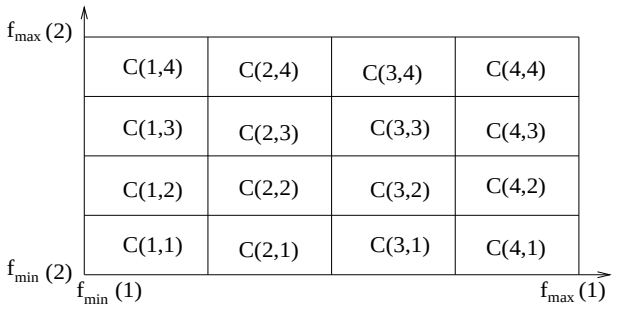

Figure 6 : An example of the labelling convention of the cell indices in a two-dimensional space

The construction of the histogram involves finding, for each feature vector, its corresponding cell location in the discrete space. Hence, all feature vectors within a cell would be assigned L integer coordinates corresponding to the indices of the cell they fall in. This is accomplished by finding the corresponding cell index for each dimension of the considered feature vector. The kth cell index, d_k, for a cell in which a feature vector $\vec{f} = [f(1), f(2), \cdots, f(L)]$ falls in, is found by

$$d_k = \begin{cases} INT(\frac{f(k) - f_{min}(k)}{CS(k)}) + 1 & \text{if } f(k) < f_{max}(k) \\ Q & \text{if } f(k) = f_{max}(k) \end{cases}, k = 1, 2, \cdots, L$$

where INT represents the integer part of the function. A cell list is created containing the coordinates of each non-empty cell, their histogram value (number of samples captured) and their "link address" which will be discussed in the next section. Thus, the corresponding cell coordinates are determined for each feature vector and compared against the current cell list. If the cell is already included in the list, its counter is incremented by one, otherwise the cell list is updated by adding that cell to it.

3.2 Clustering Method

The clustering is performed by locating the peaks of the multidimensional histogram. This is done by a peak-climbing (valley-seeking) algorithm. The procedure is illustrated in two-dimensional space in Figure 7, where the number in each cell represents a hypothetical count of the feature vectors captured by

that cell. By examining the counts of the 8-neighbours of a particular cell, a link is established between that cell and the closest cell having the largest count in the neighbourhood. If two or more neighbouring cells have the same count for a particular cell, one of them is arbitrarily chosen as a parent. At the end of the link assignment each cell is linked to one parent cell, but can be a parent of more than one cell.

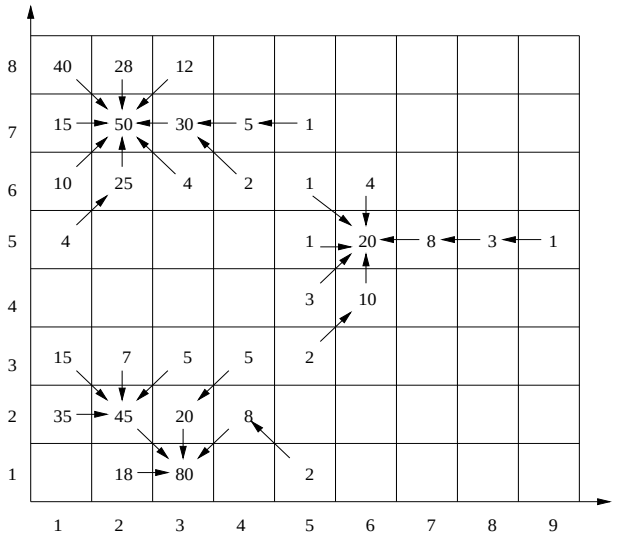

Figure 7 : Illustration of the peak climbing approach in a two-dimensional space

A peak is defined as being a cell with the largest density in the neighbourhood, i.e. a cell with no parent. At this point, a specific test is included in order to avoid any two or more peaks to be neighbours of each other. Figure 8 illustrates this situation in two dimensions where the two cells with a count of 50 are neighbours of each other. If those two particular cells were linked to the cell having the largest count in the neighbourhood, as shown in Figure 8(a), a cycle would occur. To overcome this potential problem, both of the cells are considered as peaks. At the end of the link assignment process, the location of each peak cell is examined with respect to the others. If two or more peaks are found to be neighbours, one is arbitrarily chosen as the parent of the others (Figure 8(b)). This method assumes that two or more neighbouring peaks belong to the same cluster which is a justifiable assumption.

A peak and all the cells which are linked to it are taken as a distinct cluster representing a mode of the histogram. Thus, the clustering process starts from each peak and identifies all the other cells linked to its peak. A low density region characterizes a valley in the histogram, hence, a boundary between two modes.

The clustering operation requires the examination of all the neighbouring cells of all non-empty cells. The number of neighbouring for each discrete cell is a function of the dimension of the space. Because of the discrete nature of the space, the number of possible neighbours in the two or three dimensional representation are 8 and 26 respectively. A generalization of the neighbourhood

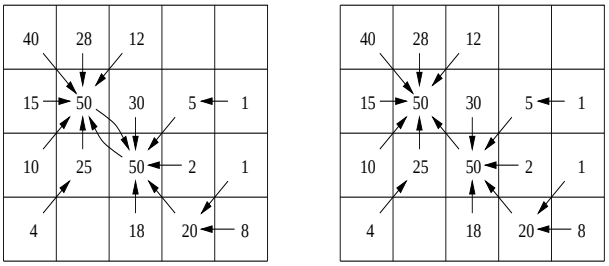

Figure 8 : (a) Example of a cycle occurring when two peaks are neighbours of each other. (b) One of the peaks is arbitrarily chosen to be parent of the other.

to k dimensions gives $(3^k - 1)$ as the closed form expression for the number of neighbours. Hence, the neighbour list of a given cell may contain at most $(3^L - 1)$ entries. Let $C(d_1, d_2, \cdots, d_L)$ be the cell under consideration. A cell $C(k_1, k_2, \cdots, k_L)$ is considered to be its neighbour if :

$$Max(|d_i - k_i|, i = 1, 2, \cdots, L) = 1$$

that is, their respective coordinates differ by at most 1 in one or more dimensions.

The method is very efficient due to the data reduction resulting from working with the histogram. The procedure is non-iterative and the number of clusters does not need to be specified a priori. The power of this method is also due of the fact it can use for any dimension. Furthermore, the structure of the algorithm is adequate for a parallel implementation resulting in nearly linear speed up.

3.3 Improvements

The clustering algorithm presented above has been improved during my research. The two new elements are listed below :

- In the case when two or more neighbouring cells have the same count, it is better to choose the cell for which the sum of counts over all neighbours is higher (see Figure 9 as example : two cells have the count 50, but the lower one is selected, because its neighbours have higher count). If the sum of all neighbours is also equal, then one of them is arbitrarily chosen as a parent with the precaution not to create a cycle as shown in the Figure 8(a).

- To establish a link for a particular cell, it is better to examine a larger neighbourhood (in the original technique, only a direct $(3^k - 1)$-neighbourhood was used for a k-dimensional). Indeed, a virtual local peaks or valleys can be created by noise and such local extrema should not be considered. Using larger neighbourhood makes the technique more

robust to noise. An illustration for a one-dimensional case is shown in Figure 10. Conversely, it is important not to take a too large neighbourhood since this could remove some significant peaks. After some experience, we found that a neighbourhood with the radius of 3 is a good choice.

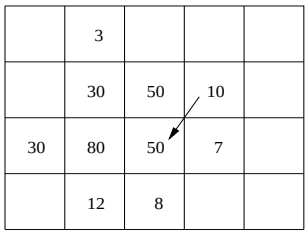

Figure 9 : Case who two neighbouring cells have the same count

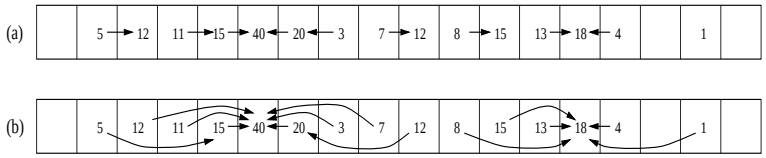

Figure 10 : (a)Clustering algorithm with a direct 2-neighbourhood.
(b)Clustering algorithm with a direct 6-neighbourhood.

To be more efficient, by using a recursive procedure we can immediately find the peak that is linked to each cell, as shown in Figure 11. This has the advantage to be more efficient. For example when one wants to find a peak corresponding to a given gray-level (colour) value.

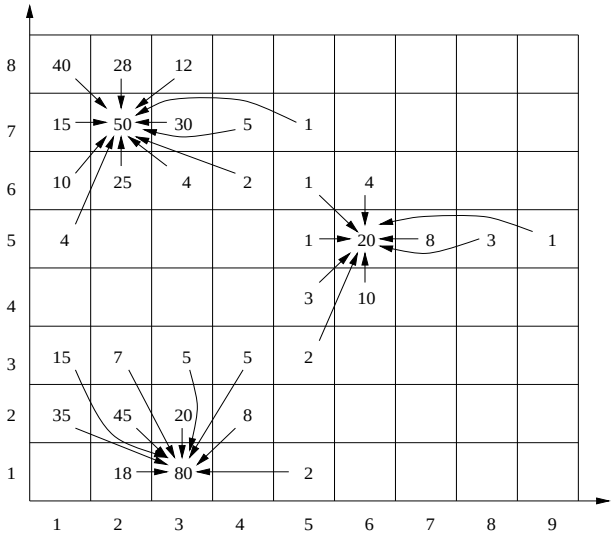

Figure 11 : Illustration of the peak climbing
approach a two-dimensional space by recursive method

3.4 Quantization Method

The quantization method plays a major role in the non-parametric and automatic clustering algorithm. A very small cell size would produce a flat histogram with no significant peaks, while too large a quantization may combine several peaks into a single mode, giving erroneous results. It is the aim of the quantization method to select the best cell size.

As we work with gray and colour image, we will use one method who is called "S" method seeing that is more appropriate for low dimensions (1-3). An another one, called "M", is suitable for larger dimensions (>3) [6].

3.4.1 The "S" method

This approach consists of evaluating the partition for several different quantizations (different cell sizes) and evaluating the goodness of each partition according to some criterion. The main advantage of this method is to provide a good insight of the ability of the data to form clusters, that is, the evolution of the different modes present in the data-set.

The histogram is generated for different values of the cell size CS(k) ranging from small to large. The cell size value for the kth dimension using the ith quantization level is

$$CS_i(k) = \left\{ \frac{f_{max}(k) - f_{min}(k)}{Q_i} \right\}, \quad k = 1, 2, \cdots, L$$

$$\text{with } Q_i = 1, 2, \cdots, N_k$$

where N_k represents the size of the kth dimension. For each quantization level, the clustering is performed and the number of peaks along with their connected cells are identified. Next, a measure of the goodness of each cluster, which takes into account its compactness and isolation, is evaluated. Let $D_i^{(m)}$ be the density of the mth cluster using Q_i. It is defined as

$$D_i^{(m)} = \frac{N_{si}^{(m)}}{N_{ci}^{(m)}}$$

where $N_{si}^{(m)}$ and $N_{ci}^{(m)}$ represent the number of data vectors and non-empty cells of the mth cluster using Q_i respectively.

A quantitative measure $S_i^{(m)}$ is then defined on $D_i^{(m)}$s which reflects the compactness of the mth cluster

$$S_i^{(m)} = \frac{|D_i^{(m)} - D_i^{(j^*)}|}{d_i^{(m)}}$$

$$\text{with } d_i^{(m)} = min_{j(j \neq m)} \|C_i^{(j)} - C_i^{(m)}\|, \quad j = 1, 2, \cdots, K_i$$

where j^* represents the value of j obtained from the above minimization.

In the above expression, K_i is the total number of clusters found using Q_i and $C_i^{(j)}$ denotes the coordinate of the jth peak cell. In simple terms, $S_i^{(m)}$ measures the difference between the density of the mth cluster and that of its nearest cluster weighted by the inverse of the distance between their peaks.

An overall goodness measure for Q_i can be defined as

$$S_i = \sum_{m=i}^{K_i} S_i^{(m)}$$

A small value of S_i would indicate a partition with compact and well isolated clusters. Thus S_i is computed for different values of Q_i and the clustering corresponding to that Q_i which minimizes S_i is selected as the best one.

3.4.2 Improvements

The drawback of the "S" method concerns mainly the computation time required by the operation, as the need to cluster the data several times may affect the efficiency of the whole procedure. Especially for a 3 dimensional space, i.e for a colour image. Although the computational burden is very heavy, the improvement resulting from the parallelization of this method makes this approach more appealing, we can further decrease the computation time required (for the colour image) by separating each colour component and using the clustering algorithm independently. For example, for a colour image in the RGB model, instead of finding all the clusters in 3-dimensional, we work in 1-dimensional for the Red, Green and Blue colour independently like a gray value. It is clear that the it is just an approximation but we found it sufficiently good for colour images. Moreover, by using a different colour model for the same image, the position and shape of the peaks will change and this could facilitate the detection of peaks.

The "S" method described above finds the best quantization level. It is important not to forget that this algorithm will be used for the compression. Thus, it is clear that the smaller the number of clusters, the better the compression is, due to the fact that the segmentation image has lower entropy. Hence, we can control the compression ration and image quality by varying the number of clusters (peaks) that we detect. The idea is to specify the number of cluster that one would like for each colour component and use the "S" method to find the best quantization for these numbers :

- Calculate the number of clusters for various quantization levels
- When the number of clusters desired is reached, use the "S" method to select the best one.

Of course, if the number of cluster specified does not exist, we take the closest number of cluster that we can find.

The power of this algorithm is that it is independent of the clustering and quantization algorithm and we can control the compression ration and image

quality. Also, a different clustering or quantization algorithm could be applied to our technique, for example the one presented on [9].

Chapter 4

Compression

As we have seen in the last chapter, we can control the number of clusters in an image. It is now time to design a compression algorithm using this property.

We will introduce the 1D CCITT Standard and the 2D CCITT Standard algorithms these developed for digital facsimile transmission [10] [11]. We had to modify these algorithms to remove the correction mechanism introduced to cope with transmission errors because it is not required by our application. In addition we also had to modify the algorithm to cope with gray-level images. This is because for a colour image each colour component is treated separately like as gray-level value.

We first introduce the original algorithm, following by description of the modification introduced during the research.

4.1 1D CCITT Standard

To understand how the 1D CCITT Standard coding scheme works, let us first examine one line of a bi-level image shown in Figure 12.

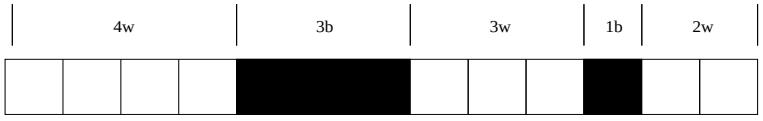

Figure 12 : One line of a bi-level image

In this example, we have a total of 13 pixels in one line of a bi-level image. The sequence of pixels is 4 white, 3 black, 3 white, 1 black, followed by 2 white pixels. These white runs and black runs alternate. Of course, we can use run-length coding to encode this line :(4w,3b,3w,1b,2w). However, it is not efficient enough. Since the probabilities of runs are different, we certainly can take advantage of this by using a variable length code to achieve a better compression efficiency.

The 1D CCITT Standard coding scheme uses a variable length code, the Modified Huffman (MH) code. A line of an image is encoded as a series of vari-

able length code words and each code word represents a run length of either all white or all black. The Modified Huffman code has two types of code words : Terminated code words and Make-up code words. Terminating code words are used to represent run lengths in the range of 0 to 63 pixels. Each make-up code word in conjunction with a terminating code word is used for encoding a run length in the range of 64 to 1728 pixels. For run lengths larger than 1728, a set of optional make-up code words can be used, supporting run lengths in the range of 1792 to 2560 pixels. Followed each line of data, there is an EOL (End-Of-Line) code word. This EOL code word is also required prior to the code words of the first image line. A variable length of string 0's may be inserted between a line data segment and an EOL. This 0 string is called Fill. Six consecutive EOL's form a special code word called RTC (Return to Control), indicating the end of an image.

The 1D CCITT Standard coding scheme can be summarized as a coding flow chart shown in Figure 13 in appendix A. Terminated code, make-up code, and additional make-up codes can be find in [11].

4.2 2D CCITT Standard

The 2D CCITT Standard coding uses the Modified Human code and the Modified Relative element address designate (MR) code. The main advantage of the 2D CCITT Standard coding over the 1D CCITT Standard coding is that compression efficiency is improved. This is due to the fact that the two-dimensional coding explores correlation of pixels in two successive lines.

To understand how the 2D CCITT Standard coding works, the definitions of the following terms are necessary :

Term	Definition
Colour of an element	Black or white
Changing element	An element whose colour is different from that of the previous element along the same scan line
Coding line	The current scan line
Reference line	The previous scan line
Coding mode	A method to code the position of each changing element along the coding line
Reference element	An element whose position determines a coding mode

The 2D CCITT Standard coding is a line-by-line coding method. The position of each changing picture element on the coding line is coded with respect to the position of a corresponding reference element. The reference element is located on either the coding line or the reference line. After the coding line has been coded, it becomes the reference line for the next coding line. In order to limit the facsimile transmission error, a Modified Human coded line (i.e., one-dimensional coded line) is sent periodically. This period is referred to as

the K factor. For facsimile, at the Standard vertical resolution, K is equal to 2; at the higher vertical resolution, K is equal to 4. For digital image, K can be any positive non-zero integer.

In summary, for every group of K lines, the 2D CCITT Standard schemes encodes 1 line in one-dimensional coding and the other K-1 lines in two-dimensional coding.

In the 2D CCITT Standard coding, there are five changing picture element as listed following :

Changing Element	Definition
a_0	The reference element on the coding line
a_1	The next changing element to the right of a_0 on the coding line
a_2	The next changing element to the right of a_1 on the coding line
b_1	The next changing element on the reference line to the right of a_0 and of opposite colour of a_0
b_2	The next changing element to the right of b_1 on the reference line

At the starting of the coding line, the changing element a_0 is first set on an imaginary white changing element located just before the first element on the coding line. During the encoding, the position of a_0 is determined by the previous coding mode.

An example of these five changing elements is shown in Figure 14.

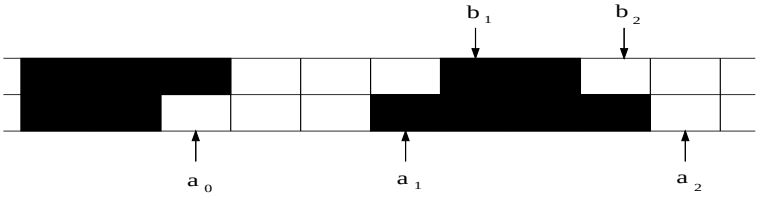

Figure 14 : An example of changing picture elements

In the 2D CCITT Standard, there are three coding modes : Pass Mode (P), Vertical Mode (V), and Horizontal Mode (H). Depending on the position of a changing element along the coding line, one of three coding modes can be identified. These three coding modes are listed following :

Coding Mode	Condition
Pass Mode	When the position of b_2 lies to the left of a_1
Vertical Mode	When the relative distance between a_1 and b_1 is less than or equal to 3
Horizontal Mode	When neither pass mode nor vertical mode occur

In the vertical Mode, depending on the relative distance between a_1 and b_1, seven possible cases can be identified :

- $V(0)$: a_1 just under b_1,
- $V_R(1)$: a_1 one pixel to the right of b_1,
- $V_R(2)$: a_1 two pixel to the right of b_1,
- $V_R(3)$: a_1 three pixel to the right of b_1,
- $V_L(1)$: a_1 one pixel to the left of b_1,
- $V_L(2)$: a_1 two pixel to the left of b_1,
- $V_L(3)$: a_1 three pixel to the left of b_1.

Besides three modes, there is also an optional extension of two-dimensional coding scheme for an uncompressed mode.

The code words for three coding modes (pass, horizontal and vertical coding modes) can be find in [11]. Examples of these three coding modes are shown as in Figure 15.

The coding flow chart of 2D CCITT Standard scheme is shown in Figure 16 in appendix A.

The decompression procedure can be implemented through searching a code tree for modified READ code as shown in Figure 17.

4.3 Modifications

Since our application is not a facsimile transmission, the first modification is to remove all the bits needed for transmission. Thus, the EOL (End-Of-Line) and RTC (Return To Control) code word were removed. These code word are not needed because the number of pixels in the image is constant and known to the decoder by the code file. Moreover, for the 2D CCITT Standard algorithm, the one-dimensionally coded line is sent periodically, in our implementation this feature was also removed. The first line of the image still has to be encoded by the one-dimensional algorithm since it is a reference line. It seems that the Extension mode is also not needed.

The second important modification was to extend these algorithms to cope with multi-valued images. The number of the region has to be known to calculate the number of bits necessary to encode the region. A look-up table, which allocates colours to regions, has to be built (as shown in Figure 18) and encoded in the header of the code file.

Each region plays the same role and is encoded in the same way. It is the changing element a_0 who determines the reference region. At the starting of the coding line, the changing element a_0 is first set on an imaginary region (the first region in the look-up table) changing element located just before the first element on the coding line. During the encoding of the coding line, the position

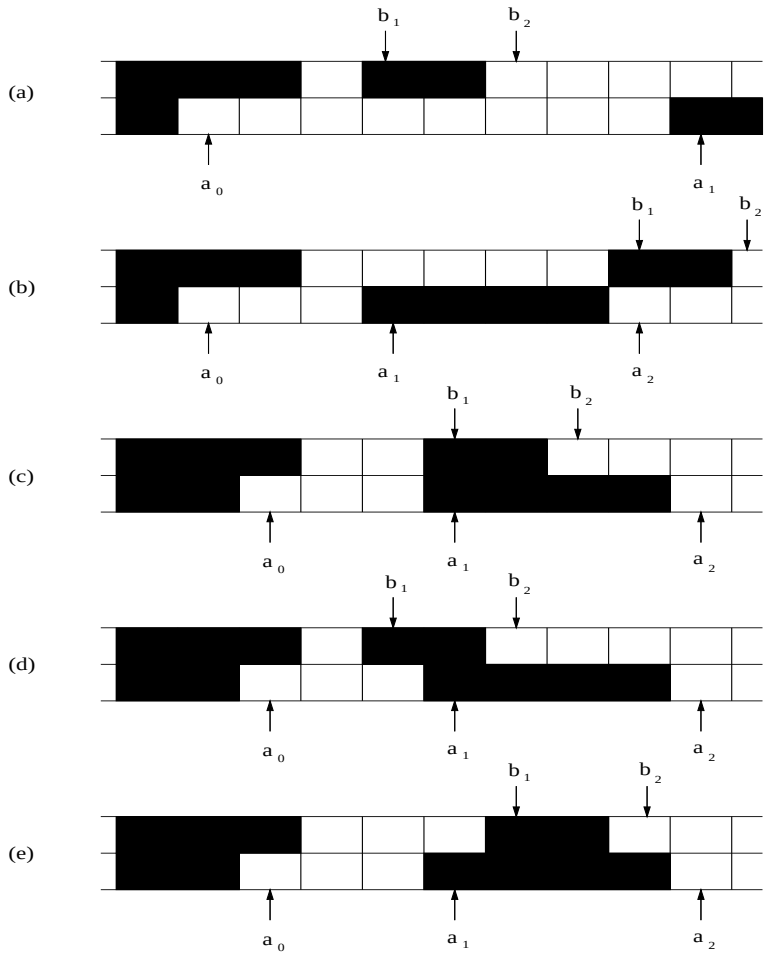

Figure 15 : (a) An example of pass mode. (b) An example of Horizontal mode. (c) An example of Vertical mode, $V(0)$. (d) An example of Vertical mode, $V_R(1)$. (e) An example of Vertical mode, $V_L(1)$.

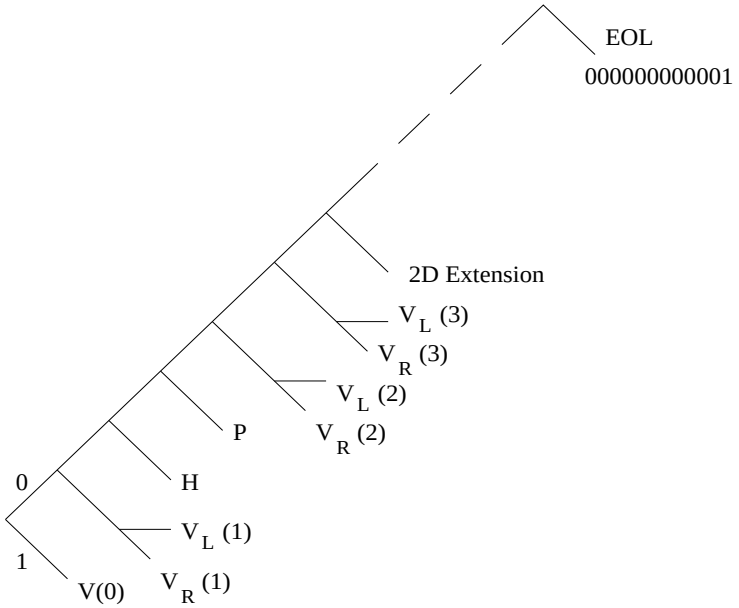

Figure 17 : Code tree for modified READ code

Gray Value	Code word
0	00
89	01
145	10
220	11

(a)

Gray Value	Code word
34	000
78	001
90	010
124	011
167	100
210	101

(b)

Figure 18 : Examples of a look-up table, which allocates colours to regions for (a) 4 colours, (b) 6 colours.

a_0 is determined by the appropriate code mode. After encode of all region, by mapping gray-level to region indices, we use the original algorithm with the exception that the first different colour encountered plays the role of the next changing element (see Figure 19).

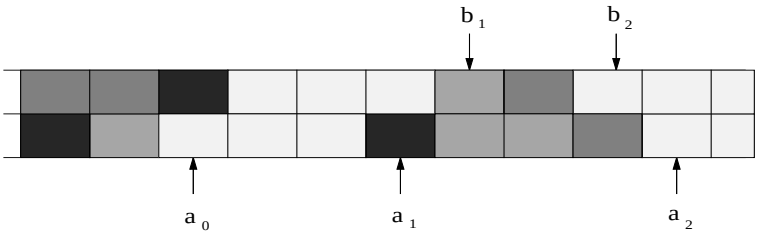

Figure 19 : An example of changing picture elements for gray-level values

The new coding flow chart for the gray-level values scheme is shown in Figure 20 in appendix A.

Chapter 5

Experiments and Results

Experiments were carried out with the RGB and YUV colour models. We used the PPM format file (Lenna.ppm) for the RGB model. Concerning the YUV model, 2 image sequence formats of different sizes namely QCIF and CIF were used. The QCIF format is of size 176 (width) by 144 (height), whereas the CIF format has images with the size 352 by 288. We used the first image file (Akiyo.qcif and Akiyo.cif) for our experimentation.

At first we present the results for the clustering and quantization algorithms followed by the 2D CCITT Standard algorithm. A comparison with JPEG will be made. To complete our experiments, we use a filter namely Majority Filtering described in appendix B.

5.1 RGB Model with Lenna.ppm

Lenna.ppm

Lenna.jpg

Lenna1

Lenna2

Lenna3

Lenna4

File	Description	Size (in bytes)
Lenna.ppm	Original Image containing 47 red clusters, 71 green clusters and 43 blue clusters	198163
Lenna.jpg	Lenna coding by JPEG	13847

File	Clustering and quantization	2D CCITT Standard (in bytes)
Lenna1	Lenna with 12 red clusters, 18 green clusters and 11 blue clusters	73639
Lenna2	Lenna with 12 red clusters, 18 green clusters and 11 blue clusters + majority filtering	37530
Lenna3	Lenna with 5 red clusters, 7 green clusters and 4 blue clusters	30522
Lenna4	Lenna with 5 red clusters, 7 green clusters and 4 blue clusters + majority filtering	15567

We noticed that the quality of the image is good in comparison with the number of clusters :

- Lenna1 and Lenna2 have 4 times less than the number of clusters of the original image (Lenna.ppm).
- Lenna3 and Lenna4 have 10 times less than the number of clusters of the original image (Lenna.ppm).

It is also interesting to notice the important effect of the Majority Filtering concerning the compression : the size of the filtering image is half that of the corresponding unfiltered image. It is due to the fact that this filter removes some pixels noising a region as shown in Figure 21 in appendix B. Thus the run length of the data is larger.

In comparison with the JPEG algorithm, our results are less satisfactory in the quality of the image and in the compression ratio.

5.2 YUV Model

5.2.1 Qcif with Akiyo.qcif

Akiyo.qcif

Akiyo.jpg

Akiyo1

Akiyo2

Akiyo3

Akiyo4

5.2.2 Cif with Akiyo.cif

Akiyo.cif

Akiyo.jpg

Akiyo5

Akiyo6

Akiyo7

Akiyo8

File	Description	Size (in bytes)
Akiyo.qcif	Original Image containing 52 Y clusters, 44 U clusters and 23 V clusters	38016
Akiyo.jpg	Akiyo coding by JPEG	2176
File	Clustering and quantization	2D CCITT Standard (in bytes)
Akiyo1	Akiyo with 13 Y clusters, 11 U clusters and 6 V clusters	6538
Akiyo2	Akiyo with 13 Y clusters, 11 U clusters and 6 V clusters + majority filtering	4949
Akiyo3	Akiyo with 5 Y clusters, 4 U clusters and 2 V clusters	2188
Akiyo4	Akiyo with 5 Y clusters, 4 U clusters and 2 V clusters + majority filtering	1656

- Akiyo1 and Akiyo2 have 4 times less than the number of clusters of the original image (Akiyo.qcif).
- Akiyo3 and Akiyo4 have 10 times less than the number of clusters of the original image (Akiyo.qcif).

File	Description	Size (in bytes)
Akiyo.cif	Original Image containing 47 Y clusters, 38 U clusters and 17 V clusters	152064
Akiyo.jpg	Akiyo coding by JPEG	5559
File	Clustering and quantization	2D CCITT Standard (in bytes)
Akiyo5	Akiyo with 12 Y clusters, 10 U clusters and 4 V clusters	19524
Akiyo6	Akiyo with 12 Y clusters, 10 U clusters and 4 V clusters + majority filtering	13429
Akiyo7	Akiyo with 5 Y clusters, 4 U clusters and 2 V clusters	7760
Akiyo8	Akiyo with 5 Y clusters, 4 U clusters and 2 V clusters + majority filtering	5156

- Akiyo5 and Akiyo6 have 4 times less than the number of clusters of the original image (Akiyo.cif).
- Akiyo7 and Akiyo8 have 10 times less than the number of clusters of the original image (Akiyo.cif).

This time, we noticed that our compression ratio can be close or better than JPEG to the detriment of the quality of the image. We still see the role of the Majority Filtering on the compression.

Chapter 6

Conclusion

The results concerning the clustering and quantization algorithms are rather satisfactory and encouraging. The quality of the image is good in comparison to the number of clusters. The fact that an image is decomposed into the different components reduces the computational load considerably.

On the contrary, we can be disappointed by the compression algorithm used : 2D CCITT Standard. Indeed, it is not efficient to just use the previous line to encode a image which means that we must encode the colour for a same region each time it is met. An effort can be made in this direction to encode a region in one pass, thus an improvement in the compression ratio and/or the quality of the image by taking more clusters.

Lastly it will be interesting to use the methods presented to try to encode image sequences. We can easily use the advantages of the histograms. For example, if histograms of 2 images are sufficiently close, we can take one as a reference for both images. We could afterwards use the notion of motion estimation to encode image sequences.

Appendix A

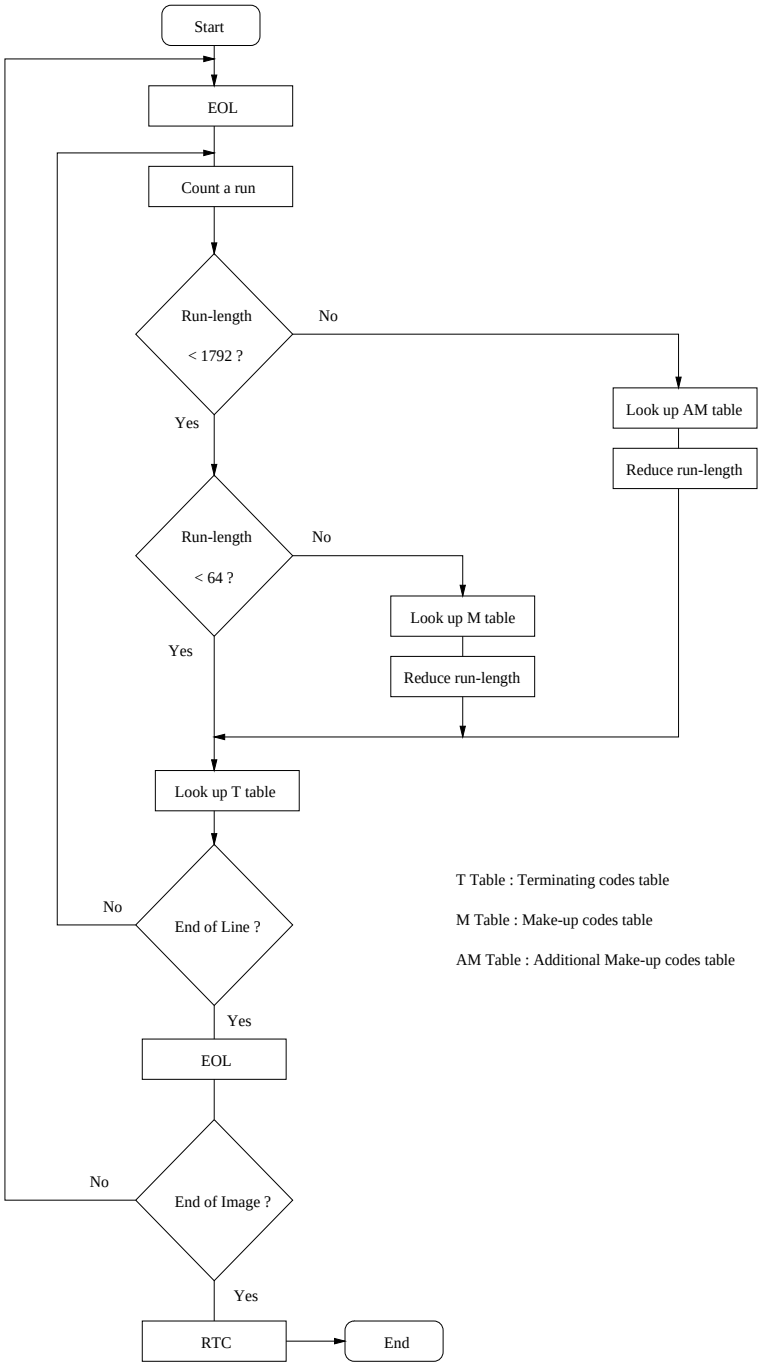

Figure 13 : 1D CCITT Standard coding flow chart

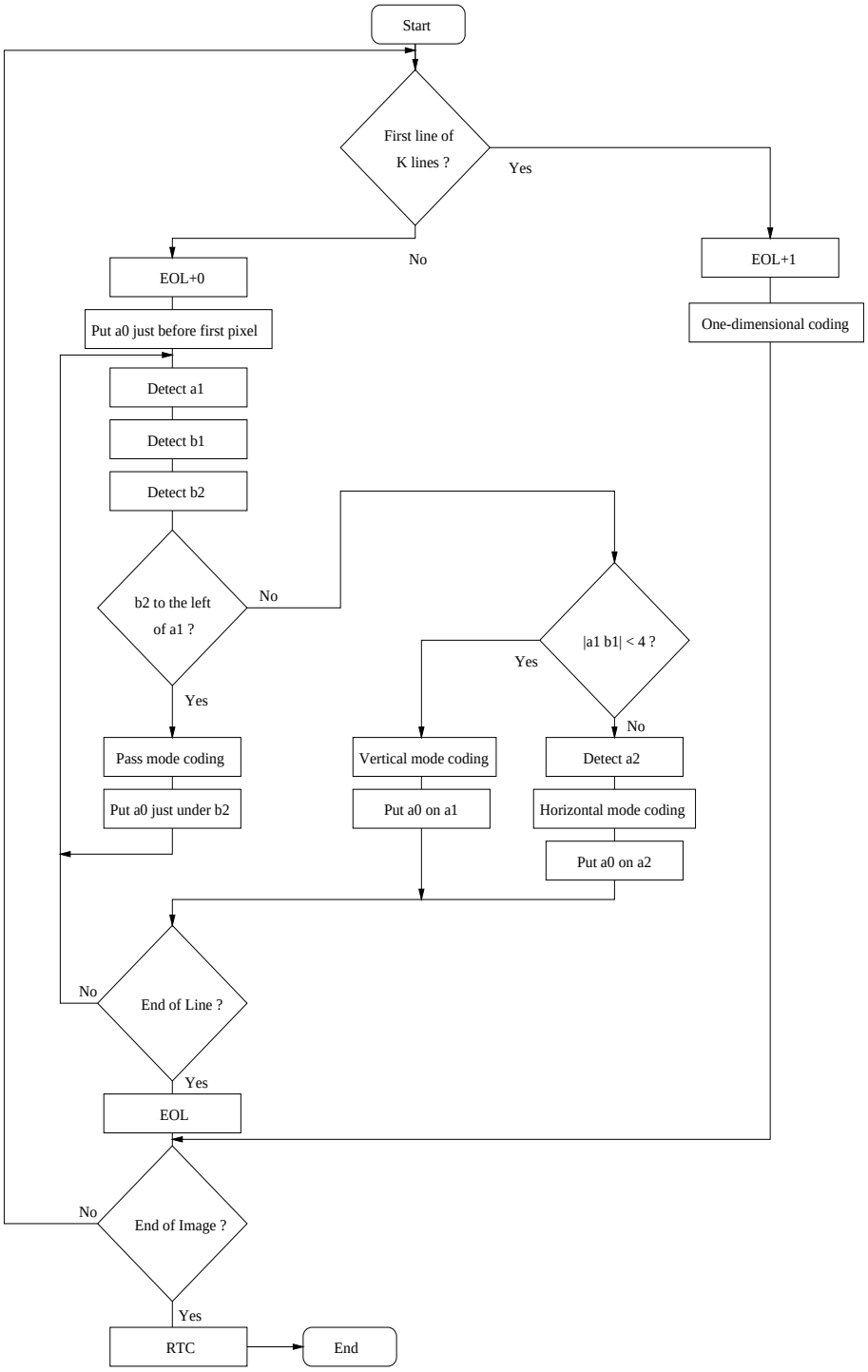

Figure 16 : 2D CCITT Standard coding flow chart

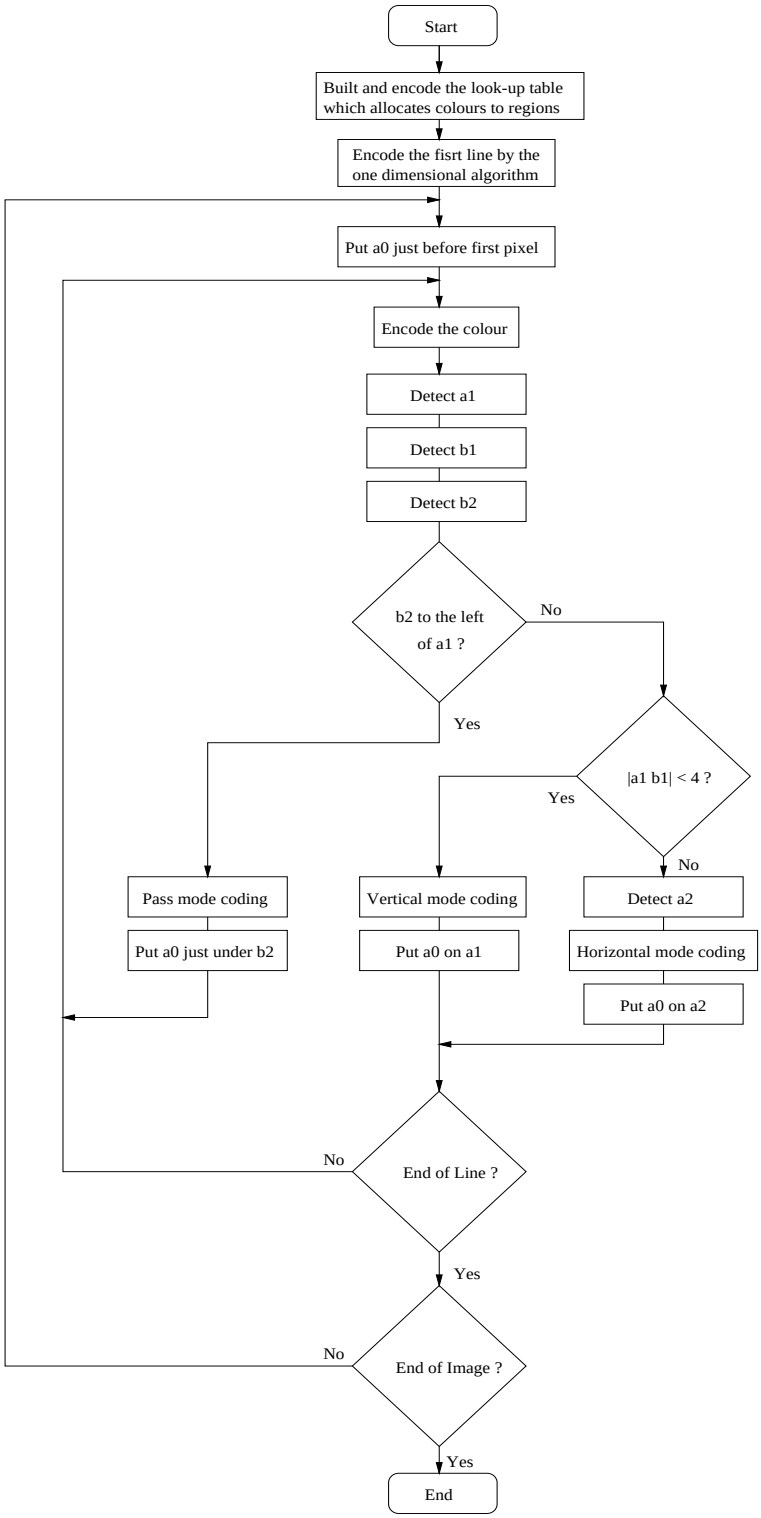

Figure 20 : gray-level coding flow chart

Appendix B

Majority Filtering

The idea of this filter is very simple. For each pixel in the image, create a local histogram and assign the most common into the pixel as shown in Figure 21.

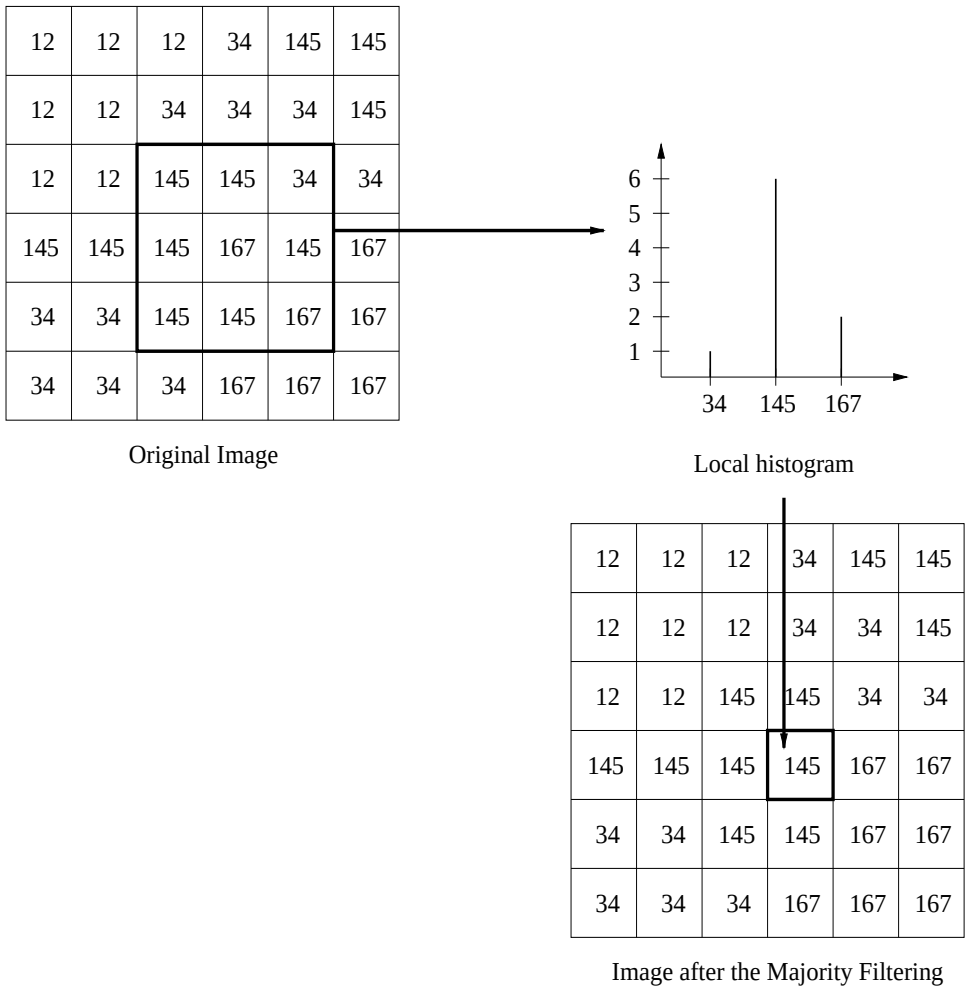

Figure 21 : An example using the Majority Filtering

Bibliography

[1] R. Jain, R. Kasturi and B. G. Schunck. *Machine Vision*. McGraw-Hill International Editions, Computer Science Series, 1995.

[2] R. M. Haralick and L. G. Shapiro. Image segmentation techniques. *Computer Vision, Graphics and Image Processing*, 29(1):100-132, 1985.

[3] G. B. Coleman and H. C. Andrews. Image segmentation by Clustering, *Proc. IEEE*, 67:773-785, May 1979.

[4] R. Dubes and A. K. Jain. Validity studies in clustering methodologies, *Pattern Recognition*, 11:235-254, 1979.

[5] A. K. Jain. Cluster Analysis, *Handbook of Pattern Recognition and Image Processing*, pp. 33-57, Academic Press,New York, 1986.

[6] A. Khotanzad and A. Bouafra. Image segmentation by a parallel, non-parametric histogram based clustering algorithm, *Pattern Recognition*, Vol 23 No. 9, pp. 961-973, 1990.

[7] D. Bourgin. Color space Frequently Asked Questions, *com.ia.vision*, June 1994.

[8] R. C. Gonzalez and R. E. Woods. Color Image Processing *Digital Image Processing*. Addison-Wesley, pp. 221-248, 1993.

[9] J. Matas and J. Kittler. Spatial and feature space clustering : Applications in image analysis, *Proc 6 International Conf. on Computer Analysis of Images and Patterns*, V Hlavac and R Sara (Eds.), Springer, 162-173, 1995.

[10] A. M. Tekalp. *Digital Video Processing*. Prentice Hall PTR, pp. 389-393, 1995.

[11] W. Kou. CCITT Facsimile Compression Standards, *Digital Image Compression : Algorithms and Standards*. Kluwer Academic Publishers, Chap. 3, pp. 37-53, 1995.