© International Conference on Color in Graphics and Image Processing - CGIP'2000

# TWO HIGH SPEED QUANTIZATION ALGORITHMS

Luc Brun and Myriam Mokhtari
L.E.R.I.
I.U.T. de Reims
Rue des Crayères
51100 Reims
brun@leri.univ-reims.fr and myriam@leri.univ-reims.fr

## ABSTRACT

*Color image quantization has been widely studied for the last fifteen years. Most of existing quantization algorithms use pure top-down or bottom-up approaches. In this paper, we present a two-pass quantization method using a split stage followed by a merge one. The split stage uses a uniform quantization algorithm to generate $N$ initial clusters. These clusters are then combined during the merge stage into $K << N$ final clusters. Finally, these clusters are used by an inverse colormap algorithm to create the output image. Two different inverse colormap algorithms are proposed leading to two quantization algorithms. These quantization algorithms are more efficient than pure split or pure merge algorithms. Moreover, the difference between the number of clusters produced by the split stage and the number of final clusters allows users to efficiently control the quality/time ratio.*

**Keywords:** *Color quantization, clustering, data structures, image compression*

## 1. INTRODUCTION

Color image quantization is the process used to reduce the number of colors of a digital image. Given the number $K$ of required colors, these algorithms produce a partition of the image color space into $K$ clusters and then assign a *representative color* to each cluster. We thus obtain a set $\mathcal{P} = \{\mathcal{C}_1, \ldots, \mathcal{C}_K\}$ of clusters and a set $\{c_1, \ldots, c_k\}$ of representative colors. The quantization algorithm creates the output image by mapping the color of each pixel in the original image to its closest representative color. The quality of the partition produced by quantization algorithms is often measured by:

$$\boldsymbol{E}(\mathcal{P}) = \sum_{i=1}^{K} \boldsymbol{SE}(\mathcal{C}_i) \text{ with } \quad \boldsymbol{SE}(\mathcal{C}_i) = \sum_{c \in \mathcal{C}_i} f(c) \parallel c - c_i \parallel^2$$

where $f(c)$ is the number of pixels with color c in the image and $c_i$ the representative color of cluster $\mathcal{C}_i$.

The term $\boldsymbol{SE}(\mathcal{C}_i)$, called the squared error of the cluster, characterizes the heterogeneity of cluster $\mathcal{C}_i$ and may be considered as the error made when assimilating $\mathcal{C}_i$ to its representative color. The term $\boldsymbol{E}(\mathcal{P})$, called the error of the partition, characterizes the heterogeneity of the partition, $\mathcal{P} = \{\mathcal{C}_1, \ldots, \mathcal{C}_K\}$ and is defined as the sum of the heterogeneities of each cluster.

A cluster $\mathcal{C}$ may also be characterized by its moments of order 0 and 1 defined by:

$$\boldsymbol{M_0}(\mathcal{C}) = \sum_{c \in \mathcal{C}} f(c) \text{ and } \boldsymbol{M_1}(\mathcal{C}) = \sum_{c \in \mathcal{C}} f(c)c.$$

The moment of order 0 of a cluster $\mathcal{C}$ is called the cardinal of the cluster and is denoted by $|\mathcal{C}|$. The mean color of $\mathcal{C}$ is defined by:

$$\mu(\mathcal{C}) \ = \ \frac{\boldsymbol{M_1}(\mathcal{C})}{|\mathcal{C}|}. \tag{1}$$

Note that the error of a partition $\mathcal{P}$ is minimal only if the representative of each cluster is equal to its mean.

Quantization algorithms may be decomposed into three main families: The top down algorithms [7, 12, 11, 2] split recursively an initial cluster containing all the colors of the image into $K$ sub-clusters. Bottom-up methods [5, 14, 13] create $K$ initial empty clusters and then merge each color of the image to one of the $K$ created clusters. Methods presented by Chou et al. [4], Lin et al. [8] and Balasubramanian et al. [1] create a binary tree with more than $K$ leaves and then prune the tree so as to select the $K$ leaves which induce the lowest partition error. Note that, in this case, the merger is restricted by the tree structure induced by the split algorithm.

In this paper, we present a quantization method which performs a uniform quantization followed by a merge step. The uniform quantization process is performed by using an implicit partition of the RGB cube into $N$ sub-cubes. The set of $N$ sub-cubes is then used as initial clusters for the merger which performs $N-K$ merges in order to obtain the $K$ final clusters.

The next sections are organized as follows: In section 2, we describe the uniform quantization algorithm and the main steps of the merger. In section 3, we describe our inverse colormap methods. In section 4, we compare our algorithms to two other quantization methods [12, 13].

## 2. ALGORITHM DESCRIPTION

Our method first uses a uniform quantization of the color space to produce $N$ clusters. Then, given the required number of output colors $K$, the merger performs $N-K$ merges in order to obtain the $K$ final clusters. At each step of the merge algorithm, we merge the two clusters which will produce the lowest increase of the partition error. Given an initial partition $\mathcal{P}=\{\mathcal{C}_0,\ldots,\mathcal{C}_N\}$, the merge of two clusters $C_i$ and $\mathcal{C}_j$ creates a new partition $\mathcal{P}'$ with an error equal to:

$$\boldsymbol{E}(\mathcal{P}')=\boldsymbol{E}(\mathcal{P})+\frac{|C_i||C_j|}{|C_i|+|C_j|}\parallel \mu(\mathcal{C}_i)-\mu(\mathcal{C}_j)\parallel^2 .$$

Therefore, given the initial partition of $N$ clusters, our merger creates a graph called the *Cluster Adjacency Graph (CAG)*. The nodes of this graph are the clusters to be merged and each edge shared by two nodes $n_i$ and $n_j$ respectively associated to the clusters $\mathcal{C}_i$ and $\mathcal{C}_j$ is valuated by $\Delta(i,j)=\frac{|C_i||C_j|}{|C_i|+|C_j|}\parallel \mu(\mathcal{C}_i)-\mu(\mathcal{C}_j)\parallel^2$. The merge operation is thus performed on the CAG by edge contractions [6].

By restricting the set of clusters which can be merged, we create a final partition with a greater error than the one obtained with unrestricted merges. Since our experiments do not show us visual artifacts due to the merge of non adjacent clusters, we do not restrict the merge and thus use a complete CAG.

If we encode the **CAG** by an half-matrix $G$, the two nodes $n_1$ and $n_2$ which have to be merged are defined by:

$$(n_1,n_2)=ArgMin_{(i,j)\in\{1,\ldots,N\}\star\{1,\ldots,i-1\}}G[i][j] \quad (2)$$

where $ArgMinF(t)$ is a value of $t$ for which $F$ is minimum.

In order to improve the search of the nodes which have to be merged, we use an array $Min$ defined by:

$$\forall i\in\{1,\ldots,N\}\ Min[i]=ArgMin_{j\in\{1,\ldots,i-1\}}G[i][j].$$

The two merged nodes $(n_1,n_2)$ are then defined by:

$$\left\{\begin{array}{lcl} n_1 & = & ArgMin_{i\in\{1,\ldots,N\}}G[i][Min[i]] \\ n_2 & = & Min[n_1]. \end{array}\right.$$

Using the array $Min$, the computation of nodes $n_1$ and $n_2$ is performed with $\mathcal{O}(N)$ tests.

If we invalidate node $n_2$, the update of this half matrix requires to invalidate line and column $n_2$ and to update line and column $n_1$. The invalidation of lines and column $n_2$ being performed in constant time, this last point requires $\mathcal{O}(N)$ operations. Moreover, we also have to update the value $Min[j]$ if $Min[j]$ is equal to $n_2$ or if the update of line and column $n_1$ has changed the minimum value of one line. In the best case where no update occurs, we have to perform $\mathcal{O}(N-n_1-1)$ tests. In the worst case, all values $Min[i]$ for $i$ greater than $n_1$ have to be updated. In this last case, the update of the array $Min$ requires $\mathcal{O}(\sum_{i=n_1+1}^{N} i-1)=\mathcal{O}(\frac{(N-n_1)(N-3(n_1+1))}{2})$ operations.

The merger needs at each node the centroid and the cardinal of the corresponding clusters. By using equation 1, the mean can be deduced from the first cumulative moment $\boldsymbol{M_1}$ and the cardinal. Moreover, as all clusters are disjoints, the first moment and the cardinal are linear operators :

$$\left\{\begin{array}{lcl} \boldsymbol{M_1}(\mathcal{C}_1\cup\mathcal{C}_2) & = & \boldsymbol{M_1}(\mathcal{C}_1)+\boldsymbol{M_1}(\mathcal{C}_2) \\ |\mathcal{C}_1\cup\mathcal{C}_2| & = & |\mathcal{C}_1|+|\mathcal{C}_2|. \end{array}\right.$$

As the mean can be deduced from the first moment and the cardinal, we only store these two data in each cluster. The size of the set of clusters is thus equal to $4N$ words. Moreover, the size of the half matrix $G$ is equal to $\frac{N(N-1)}{2}$ and the size of the array $Min$ is equal to $N$. Thus, the total memory requirement of our merge algorithm is equal to $4N+\frac{N(N-1)}{2}+N=\frac{N(N+9)}{2}$ words.

## 3. COLOR MAP INVERSION

Color map inversion is the final step of the quantization algorithm. It creates the output image by mapping each color of the original image to its representative color. We saw in Section 2 that the quantization algorithm can be decomposed in two phases: the generation of $N$ clusters $\{\mathcal{C}_1,..,\mathcal{C}_N\}$ by the uniform quantization algorithm and the merger of these clusters to obtain the $K$ final *meta-clusters* $\{\mathcal{MC}_1,\mathcal{MC}_2,...,\mathcal{MC}_K\}$ (each meta-cluster $\mathcal{MC}_i$ being equal to a cluster $\mathcal{C}_j$ with additional merger information). We saw in Section 1 that given a partition in $K$ clusters, the set of representative colors which minimizes the partition error is equal to the centroid of clusters. Thus, if $\{c_1,\ldots,c_K\}$ denotes the set of representative colors we have:

$$\forall i\in\{1,\ldots,K\}\ c_i=\mu(\mathcal{MC}_i).$$

The inverse colormap function $\boldsymbol{Q}$ which maps each color to its representative may be defined by:

$$\boldsymbol{Q}(c)=ArgMin_{z\in\{c_1,\ldots,c_K\}}||z-c||.$$

A first solution to compute $\boldsymbol{Q}(c)$ consists to perform the *exhaustive search* of $ArgMin_{z\in\{c_1,\ldots,c_K\}}||z-c||$. The exhaustive search algorithm provides an exact solution, however the time required by this algorithm to map an image is generally high. Heckbert [7] has improved the exhaustive search algorithm by using a preprocessing step. According to experiments performed by Heckbert, using his method, the $K$ tests involved by the exhaustive search fall to $\frac{K}{23}$.

The function $\boldsymbol{Q}$ may also be defined by a 3D Voronoi diagram on the centroids $c_j$. In this case, $\boldsymbol{Q}(c)$ is equal to $c_i$ if $c$ belongs to the Voronoi cell associated with $c_i$. Thomas [9] proposed an algorithm to build the 3D Voronoi diagram associated to $K$ representative colors. Since this algorithm builds a partition of the whole color space, the time required to map an image to its representative colors is high. Thomas reduces the memory requirements and the computing time by removing the last three bits of each color component. Nevertheless, his algorithm takes approximately 10 seconds to generate the 3D Voronoi diagram corresponding to 256 representative colors [3]. Since most of top-down quantization algorithms take approximately less than 30 seconds to quantize a $512 \times 512$ image, the time overhead induced by Thomas inverse colormap algorithm is high.

Brun [3] proposed to approximate the Thomas 3D Voronoi diagram by a 2D one. However, the construction of this 2D Voronoi diagram requires to compute the projection of each image color on the plane defined by the two first eigenvectors of the covariance matrix of the image color set. This method takes approximately 5 seconds to perform the inverse colormap operation and performs usually faster than the exhaustive search, Heckbert or Thomas methods. However, the time overhead induced by this method remains high for fast quantization methods such as the ones presented in this paper.

Another solution consists in approximating the 3D Voronoi cells induced by the function $\boldsymbol{Q}$ by the clusters produced by our quantization algorithm. As a matter of fact, the partition created by quantization algorithms generally does not constitute a Voronoi partition on $K$ representative colors. However, according to Wu [11], if the quantization algorithm produces a low partition error, the simple mapping of $c \in \mathcal{MC}_j$ to $c_j$ does not cause a significant increase in numerical quantization error nor a noticeable loss of image quality from the exhaustive search algorithm. If we use a top-down algorithm based on a binary tree, the complexity of the mapping function with $K$ representative colors is then equal to $\mathcal{O}(\log_2(K))$.

In this section, we present two inverse colormap algorithms. The first one approximates the function $\boldsymbol{Q}$ by mapping each color to the centroid of its enclosing cluster. The second one provides a mapping function which is almost always equal to the exact solution provided by the exhaustive search or Heckbert [7] algorithms. Both algorithms are based on the same preprocessing step which allows us to find the enclosing meta-cluster $\mathcal{MC}_i$ of a cluster $\mathcal{C}_j$ with only one indirection.

In order to perform this indirection, we add a canonical pointer in each cluster. The canonical pointer of a cluster $\mathcal{C}_i$ is initialized to its index i. When we merge two clusters $\mathcal{C}_i$ and $\mathcal{C}_j$, the information about the merger is stored in the cluster which has the greatest index. Thus, if we merge cluster $\mathcal{C}_1$ with cluster $\mathcal{C}_3$ and then cluster $\mathcal{C}_3$ with cluster $\mathcal{C}_5$ we obtain:

$$\mathcal{C}_1.canonical = 3,\ \mathcal{C}_3.canonical = 5,\ \mathcal{C}_5.canonical = 5.$$

Hence, clusters $\mathcal{C}_1$, $\mathcal{C}_3$ and $\mathcal{C}_5$ have been merged and parameters of the merge are stored in cluster $\mathcal{C}_5$.

From cluster $\mathcal{C}_1$, we have to perform two indirections to find the cluster which contains merge information. These indirections can be suppressed by using the following algorithm:

```
set_canonic (N)
{
    for i = N to 0 step -1
    {
        j = C_i.canonical
        C_i.canonical = C_j.canonical
    }
}
```

After the algorithm `set_canonic`, all clusters which have been merged into a same *meta-cluster* share the same canonical pointer. The meta-clusters $(\mathcal{MC}_i)_{i\in\{1,\dots,K\}}$ can then be defined by:

$$\forall i \in \{1,\dots,K\}\ \mathcal{MC}_i = \bigsqcup_{\{k/\mathcal{C}_k.canonical=i\}} \mathcal{C}_k.$$

Our first inverse colormap algorithm decomposes the mapping of a color $c$ to $\boldsymbol{Q}(c)$ in two operations. First, it finds the cluster $\mathcal{C}_j$ generated by the uniform quantization which encloses the color $c$. Second, given cluster $C_j$, it finds the meta-cluster $\mathcal{MC}_i$ which encloses or is equal to $\mathcal{C}_i$. The index of this meta-cluster is equal to $\mathcal{C}_j.canonical$.

For any color $c$, its enclosing initial cluster is retrieved with only two multiplications. Thus, the computation of the representative color $c_i$ of a given color $c$ is performed with two multiplications and one indirection. The complexity of `set_canonic` algorithm is equal to $\mathcal{O}(N)$. Typical values of $N$ being equal to 200 or 300, $N$ is negligible besides the size of the image and the time overhead involved by this preprocessing step is negligible.

Our second inverse colormap algorithm can be viewed as an adaptation to our data structure of the Heckbert [7] limitation of the exhaustive search. The basic idea of this algorithm is that the misclassified colors belong to adjacent clusters enclosed in different meta-clusters. Thus for each cluster $\mathcal{C}_i$, we compute a list $L_i$ which contains the list of indices of meta-clusters adjacent to cluster $\mathcal{C}_i$. This list is defined by:

$$L_i = \{C_i.canonical\} \cup \{C_j.canonical \ / \ C_j \ adj \ C_i \text{ and } C_j.canonical \neq C_i.canonical\}$$

where *adj* is a connectivity relationship between the clusters generated by the uniform quantization algorithm. Note that the uniform quantization algorithm provides a partition of the color space into a set of cubes. Thus, the 6, 18 and 26 connectivity of voxels may be defined on clusters generated by the uniform quantization algorithm.

Given a color $c$ enclosed in cluster $\mathcal{C}_i$, our second inverse colormap algorithm maps $c$ to $\boldsymbol{Q}(c) = c_k$ with $k \in L_i$ such that:

$$||c - c_k|| = Min_{j \in L_i} ||c - c_j||.$$

Experiments resumed on Table 1 show that the improved inverse colormap algorithm does not obtain the same results than the exhaustive search. However, the greatest difference between the errors obtained with the improved inverse colormap algorithm and the exhaustive search does not exceed half a one. This difference represents 0.14% of the error produced by our algorithm and is thus negligible. The 26-connexity of clusters induces lower errors than the 18-connexity of clusters. However, once again the differences between errors produced by both algorithms are negligible and the 18-connexity of clusters induces slightly shorter times than the 26-connexity. We thus decided to use the 18-connexity of clusters. The mean size of lists $L_i$ is displayed for each test image on Table 2. This table shows that using the 18-connexity of clusters the size of lists $L_i$ is about 1.65 for a quantization into 16 colors. Thus, our inverse colormap algorithm has to perform about 1.65 tests by pixel to map the output image into 16 colors.

| $\boldsymbol{E}$ | 1 | 2 | 3 |
|---|---|---|---|
| Lenna | 224.42 | 224.51 | 224.51 |
| Zelda | 347.95 | 348.42 | 348.43 |
| Girl | 250.81 | 250.91 | 250.94 |
| House | 125.27 | 125.31 | 125.36 |

| | | |
|---|---|---|
| (1) | : | Exhaustive search algorithm |
| (2) | : | 26-connexity of clusters |
| (3) | : | 18-connexity of clusters |

Table 1: Error in RGB space produced by our algorithm with three different inverse colormap algorithms.

| $\lvert L_i \rvert$ | 1 | 2 |
|---|---|---|
| Lenna | 2.30 | 1.98 |
| Zelda | 2.14 | 1.78 |
| Girl | 1.78 | 1.53 |
| House | 1.37 | 1.33 |
| Mean | 1.89 | 1.65 |

| | | |
|---|---|---|
| (1) | : | 26-connectivity of clusters |
| (2) | : | 18-connectivity of clusters |

Table 2: Average size of lists $L_i$ with different connectivities and 16 output colors. The size of lists $L_i$ represents the number of tests required to map each color to its representative color.

# 4. EXPERIMENTS

In this section, we compare our algorithms to the top-down and bottom-up quantization methods respectively designed by Wu [12] and Xiang [13]. Let us denote by *split_merge* and *improved_split_merge* our quantization algorithms respectively with the first and second inverse color map methods (see Section 3).

Experiments summarized in Table 3 show the partition errors produced by our quantization methods and Wu's one [12] with 16 output colors and 4 test images. We do not display on this Table the partition errors produced by Xiang's algorithm since this algorithm does not minimize the partition error of the output image. Table 3 shows that on each test image the *improved_split_merge* algorithm performs better than Wu's one. Using the same Table, the *split_merge* algorithm produces images with a greater partition error than Wu's one except on Girl test image. However, the maximal difference between partition errors produced by both algorithms do not exceed 10%. According to our experience, the visual quality of quantized images are difficult to differentiate for so closed errors.

| $\boldsymbol{E}$ | Lenna | Zelda | Girl | House |
|---|---|---|---|---|
| improved_split_merge | 224 | 348 | 250 | 125 |
| split_merge | 257 | 396 | 278 | 144 |
| Wu algorithm | 241 | 367 | 285 | 132 |

Table 3: This table shows the partition errors computed in the $RGB$ color space of the 3 tested algorithms to perform a quantization of our test images into 16 output colors. The *improved_split_merge* and *split_merge* algorithms use an initial partition of the $RGB$ cube into $N = 3000$ cubes.

We have compared the images produced by our algorithms with the one produced by Xiang's one [13] by using 6 comparison algorithms defined by Trémeau [10]. If $image1$ and $image2$ denote the two compared images, Trémeau's algorithms [10] perform the following operations on each pixel $P$ of $image1$ and $image2$:

1. Compute two mean values based on a neighborhood of $P$ in $image1$ and $image2$.

2. Subtract these values and normalize them between 0 and 1 according to local criteria. A 0 value denoting a maximal difference while a 1 value denotes a lack of difference.

The value associated to each pixel of the output image depends on the criterion used for the comparison. The 6 used criteria are:

1. The luminance: the value computed on each pixel of input images is its mean luminance.

2. The chrominance: the mean chromatic coordinates of each pixel are used for comparison.

3. The emergence: the emergence of one pixel is defined as the mean distance between its color and the color of its neighbors. The emergence of each pixel is then used for comparison.

4. The correlation: the local covariance between the colors of both images is used. This covariance is normalized by the product of the standard deviations computed on both images.

5. The Edge detection: this criterion applies a color edge detection algorithm on each image and then subtracts the obtained values.

6. The Fisher criterion: this criterion computes the distance between the mean color of each pixel on both images (see Figure 1).

Further details about each criterion may be found in [10].

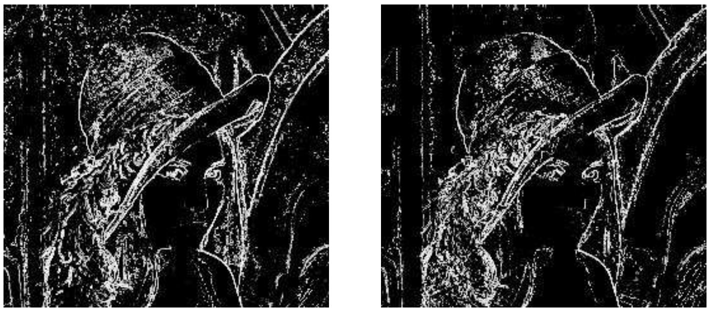
The gray value of each pixel may be summed over each image produced by Trémeau [10] algorithms in order to obtain a global criterion. This value is then normalized so that a value equal to 100 is obtained for two identical images according to the considered criterion while a value equal to 0 represents a maximal difference. Table 4 shows the obtained values for each criterion by the *improved_split_merge*, *split_merge* and *Xiang* quantization algorithms on Lenna test image quantized in 16 colors. For each criterion, the lowest value is obtained by Xiang algorithm. With a small difference, the *split_merge* algorithm obtains higher or equal values than the *improved_split_merge* algorithm for the Chrominance, Emergence and Edge criteria. In the opposite, with a significant difference, the *improved_split_merge* algorithm obtains greater values than the *split_merge* algorithm for the Luminance, Correlation, and Fisher criteria. Therefore, according to Trémeau's criteria, the *improved_split_merge* algorithm performs better than the *split_merge* and Xiang's algorithms.

| Criterion | 1 | 2 | 3 |
|---|---|---|---|
| Luminance | 77.4 | 75.7 | 64.4 |
| Chrominance | 20.7 | 21.4 | 21.1 |
| Emergence | 11.1 | 11.7 | 8.6 |
| Correlation | 40.6 | 38.8 | 31.9 |
| Edge | 35.5 | 35.5 | 24.0 |
| Fisher | 15.0 | 13.6 | 12.3 |

| | | |
|---|---|---|
| (1) | : | *improved_split_merge* |
| (2) | : | *split_merge* |
| (3) | : | Xiang algorithm |

Table 4: This table shows the differences between the quantized images in 16 colors and the original one computed thanks to the criteria defined by Trémeau [10].

Table 6 shows the time required by the four quantization algorithms to map our test images in 16 colors. For each test image, the algorithms may be sorted in the increasing order, according to their execution time: The *split_merge*, *improved_split_merge*, Xiang's and Wu's algorithms. Note that the low difference between the time required by the *split_merge* and *improved_split_merge* algorithms confirms that the second inverse colormap does not induce a high time overhead (see Section 3 and Table 2). The greater differences obtained for Lenna and Zelda images may be explained by the time overhead of the second inverse colormap algorithm which is approximately linear according to the size of the image (see Tables 2 and 5).



(a) (b)

Figure 1: *This Figure shows the difference of Fisher [10] computed between Lenna image quantized into 16 colors by our* improved_split_merge *(a) and Xiang (b) algorithms and the original one.*

| | Lenna | Zelda | Girl | House |
|---|---|---|---|---|
| Size | $512 \times 480$ | $504 \times 574$ | $256 \times 256$ | $256 \times 256$ |

Table 5: The Table shows the size of each test image.

| time in seconds | Lenna | Zelda | Girl | House |
|---|---|---|---|---|
| improved_split_merge | 0.7 | 1.1 | 0.3 | 0.2 |
| split_merge | 0.6 | 0.9 | 0.3 | 0.2 |
| Wu algorithm | 16.0 | 10.5 | 4.3 | 3.7 |
| Xiang algorithm | 9.2 | 6.7 | 2.3 | 2.4 |

Table 6: This table shows the execution time required by the 4 tested algorithms to perform a quantization of our test images into 16 output colors. The *improved_split_merge* and *split_merge* algorithms use an initial partition of the $RGB$ cube into $N = 3000$ cubes.

After the uniform quantization step in $N$ initial clusters, our quantization algorithms behave like bottom-up methods and perform $N - K$ merges in order to obtain the $K$ final representative colors. Figure 2 shows the influence of the first splitting step. The curves plotted in this Figure represent the partition error of the output images quantized in 16 colors by the *improved_split_merge* algorithm as a function of the initial number $N$ of clusters. These curves drop abruptly for

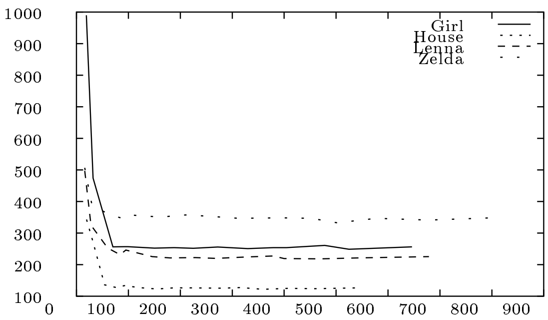



Figure 2: *This Figure shows the partition error produced by the* improved_split_merge *algorithm on four test images as a function of the number $N$ of initial clusters.*

low values of $N$ and then remain approximately constant for each test image. Therefore, the number $N$ of initial clusters must be greater than a given threshold which is approximately equal to 200 with our test images. However, a first splitting step into $N >> 200$ clusters induces useless merges and an important time overhead. Moreover, if $N$ is close to the number of colors of the initial image, our algorithms behave like pure bottom-up quantization methods. The use of a reduced number of data obtained by the first splitting step allows us to apply more complex heuristics than the ones usually used by pure bottom-up methods. Indeed, the iterative determination of the best two clusters to be merged leads to unbearable processing times if $N$ represents the number of colors of the input image.

## 5. CONCLUSION

Our algorithms are controlled by the number $K$ of final colors and the number $N$ of initial clusters. Given $N = K$, our algorithms perform only a split operation and may thus be considered as top-down algorithms. If $N$ is equal to the number of image's pixel, our algorithms behave like bottom-up algorithms. Our experiments show us that the partition error remains approximately constant when the number of clusters is greater than a given threshold approximately equal to 200 in our experiments. Therefore, using a first uniform quantization avoids the computation of a set of useless merge and allows us to apply more efficient heuristics than usual pure bottom-up quantization algorithms.

## 6. REFERENCES

[1] R. Balasubramanian and J. Allebach. A new approach to palette selection for color images. *Journal of imaging technology*, 17(6):284–290, december 1991.

[2] J. P. Braquelaire and L. Brun. Comparison and optimization of methods of color image quantization. *IEEE Transactions on Image Processing*, 6(7):1048–1052, july 1997.

[3] L. Brun and Claude Secroun. A fast algorithm for inverse color map computation. *Computer Graphics Forum*, 17(4):263–271, december 1998.

[4] P. A. Chou, Lookabaugh T., and Gray R. M. Optimal pruning with applications to tree-structured source coding and modeling. *IEEE Trans. Inf. Theory*, 2:299–315, 1989.

[5] M. Gervautz and W. Purgathofer. A Simple Method for Color Quantization: Octree Quantization. In A. S. Glassner, editor, *Graphics Gems*, pages 287–293. Academic Press Professional, Cambridge, MA, 1990.

[6] Frank Harary. *Graph Theory.* Addison-Wesley, 1972.

[7] Paul Heckbert. Color image quantization for frame buffer display. In *Proceedings of SIG-GRAPH'82*, pages 297–307, 1982.

[8] J. Lin, J. Storer, and M. Cohn. On the complexity of the optimal tree pruning for source coding. In *Proc. of data compression conference*, pages 63–72. IEEE Computer Society Press Los Angeles, 1991.

[9] S. W. Thomas. Efficient Inverse Color Map Computation. In J. Arvo, editor, *Graphics Gems II*, pages 116–125, 528–535. Academic Press Professional, Cambridge, MA, 1991.

[10] A. Trémeau, É. Dinet, and É. Favier. Measurement and display of color image differences based on visual attention. *Journal of Imaging Science and Technology*, 40(6):522–534, 1996. IS&T/SID.

[11] Xiaolin Wu. Color quantization by dynamic programming and principal analysis. *ACM Transaction on Graphics*, 11(4):348–372, 1992.

[12] Xiaolin Wu and K. Zhang. A better tree-structured vector quantizer. In *Proceedings of the IEEE Data Compression Conference*, pages 392–401. IEEE Computer Society Press, 1991.

[13] Zhigang Xiang. Color image quantization by minimizing the maximum intercluster distance. *ACM Transactions on Graphics*, 16(3):260–276, July 1997.

[14] Zhigang Xiang and Gregory Joy. Color image quantization by agglomerative clustering. *IEEE Computer Graphics and Applications*, 14(3):44–48, may 1994.