Volume xxx, (1998) number yyy pp. 000–000

A fast algorithm for inverse color map computation

L. Brun and C. Secroun

L.E.R.I., I.U.T. Léonard de Vinci, B.P. 1035, 51687 Reims Cedex 2, France
brun@leri.univ-reims.fr, secroun@univ-reims.fr

Abstract

The inverse colormap operation is the process which allows an image to be displayed with a limited set of colors. In order to obtain minimal visual distortion between the input image and the displayed image, inverse colormap algorithms associate each color with its nearest representative. The method presented in this paper is based on a Karhunen-Loève transformation, which allows us to efficiently approximate the 3D Voronoi diagram implicitly used by inverse color map algorithms by its 2D projection into the plane defined by the first two eigenvectors of the covariance matrix and then performing an additional correction step. The complexity of our algorithm is independent of the size of the colormap. Moreover, its results are equal or quite close to the optimal solution.

Keywords: color, inverse colormap, quantization, visualization

1. Introduction

Despite the increasing number of multimedia applications, most screens can only display 256 colors simultaneously. The visualization of a 24-bit-per-pixel image on such screens requires to map each color in the image into a limited set of colors. This process is called the inverse colormap operation. The limited set of colors used to display an image is called the colormap or the set of representative colors. The inverse colormap process can be used to display an image with the default colormap of the screen or to perform error-propagation dithering [1].

In order to minimize the visual distortion between the input image and the output one, inverse colormap algorithms map each color c of the input image to its nearest representative $\boldsymbol{Q}(c)$. The function $\boldsymbol{Q}$ may be defined by :

$$\begin{pmatrix} \mathcal{C} & \rightarrow & \{c_1,\ldots,c_K\} \\ c & \mapsto & \boldsymbol{Q}(c) = ArgMin_{z\in\{c_1,\ldots,c_K\}}\|z-c\| \end{pmatrix} \quad (1)$$

where $\mathcal{C}$ represents the set of colors of the input image, and $\{c_1,\ldots,c_K\}$ the set of representative colors. The symbol $\|z-c\|$ denotes the Euclidean norm of the 3D vector $z-c$ computed in a given color space. Symbol $ArgMin f(t)$ stands for a value t, which realizes the minimum of f.

Inverse colormap algorithms can also be used as the output phase of quantization algorithms [2, 3, 4]. In the latter case, the colormap used to display the input image is provided by the quantization algorithm (see Figure 1). Quantization algorithms produce a partition of the original image color set into a set of clusters to build the colormap. Most of the colors which belong to one cluster are closer to the cluster's centroid than to the other ones. Most of quantization algorithms define the colormap from the cluster's centroid and use the above property to derive an efficient inverse colormap algorithm. Such inverse colormap algorithms are devoted to a specific quantization algorithm and are therefore unable to inverse the colormap of an image with a given set of representative colors. In the following we will focus on general inverse colormap algorithms. These algorithms being able to map an image with any given colormap.

The rest of the paper is organized as follows. In section 2, we describe two existing methods used to perform inverse colormap operation. These methods are not devoted to a specific quantization algorithm and allow one to map an image with any given col-

© The Eurographics Association 1998. Published by Blackwell Publishers, 108 Cowley Road, Oxford OX4 1JF, UK and 238 Main Street, Cambridge, MA 02142, USA.

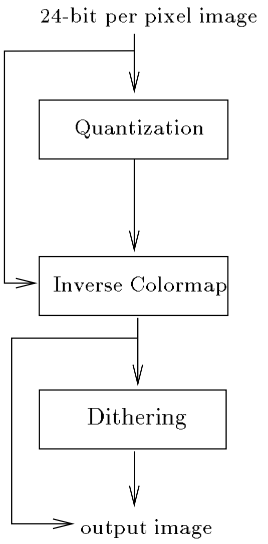

Figure 1: *The set of algorithms used to display an image on a 8-bit color display*

ormap. In section 3, we describe the main steps of our method and the overall complexity of the algorithm. Finally, in section 4, we evaluate our method through experiments.

2. The inverse colormap operation

The value $\boldsymbol{Q}(\mathrm{c})$ for a given color c may be computed thanks to an exhaustive search of the minimum of $\|z-c\|$ for all representative colors. Given a 256×256 image and a colormap of 256 colors, this trivial algorithm requires more than sixteen million distance computations (see equation 1). Thus, despite its simplicity and the fact that this method provides an exact solution, it has no practical interest for large images or interactive applications.

This trivial method has been improved by Heckbert's locally sorted search algorithm [5]. This method is based on a uniform decomposition of the RGB cube into a lattice of N cubical cells. Each cell of this lattice contains the list of representatives which could be the nearest representative of a color inside the cell. Given an original image color c, the Heckbert's method determines the cell which contains c and traverses its associated list in order to determine its closest representative. The complexity of this method depends on the mean size of representatives list which is determined by the number, the distribution of representative colors and by the number N of cells composing the lattice. Experiments show that the number of distance computations required to determine the representative $\boldsymbol{Q}(\mathrm{c})$ of a color c may be decreased by 23 for 256 representative colors and 512 cells. Nevertheless, with a fixed number N of cells, this method remains approximately linear with respect to the number K of representatives. Moreover, the optimal number N of cells is difficult to estimate.

Thomas [6] method computes the values of the function $\boldsymbol{Q}$ thanks to a 3D discrete Voronoi diagram [7, 8] defined by the representative colors of the colormap.

A Voronoi diagram defined by a set C of points included in $\mathbb{R}^n$ is a partition of $\mathbb{R}^n$ into a set of cells such as two points belong to the same cell if and only if they have the same closest point in C. The frontiers between Voronoi cells may be considered as the edges of a graph. In this case each Voronoi cell is a face of the graph and each vertex is equidistant to at least 3 points in C. The dual of this graph is called the Delaunay graph [8] associated to C. Each cell of the Voronoi diagram is then associated to one vertex of the Delaunay graph. Moreover, the set of cells adjacent to a given Voronoi cell may by retreived by computing the set of vertices adjacent to its associated vertex.
A discrete Voronoi diagram is a partition of $\mathbb{Z}^n$ or $\mathbb{N}^n$ into a set of cells which verify the same property. One advantage of discrete Voronoi diagrams beside real ones is that they can be computed thanks to incremental methods which initialize a n-dimensional array encoding the domain to be partitioned. Thus, using a discrete Voronoi diagram, the cell of a given point is retrieved by reading the index of its cell in the n-dimensional array which encodes the domain.

Using Thomas method, the Voronoi diagram is encoded thanks to a 3D array of integers. Each entry of this array represents a color and contains the index of its nearest representative color. The preprocessing step which computes the 3D discrete Voronoi diagram has several disadvantages. First, this algorithm computes the representative of each displayable color. Thus, this method involves many useless computations when applied to a few images. Secondly, using the RGB color space, the computation of the 3D Voronoi diagram requires the storage of 256^3 indexes. If the colormap sizes are limited to 256, the amount of memory required by the algorithm is equal to 16 mega bytes. In order to reduce the number of initialized values and the amount of memory required by the algorithm, Thomas removes the three last bits of each R, G, B component. The 3D Voronoi diagram is then computed on a $32 \times 32 \times 32$ cube. This heuristic decreases significantly the amount of required memory but produces several errors when the size of the colormap is greater than 256. Moreover, most of quantization algorithms [2, 9, 10] use color spaces such as Lu^*v^*, Lab [11, 12, 13] or YIQ [13] which are more adapted to human vision than the RGB one.

© The Eurographics Association 1998

In this case, inverse colormap processes which use the output of such quantization algorithms, have to use the same color space. Using Thomas method and a different color space than the RGB one, we have to allocate and initialize a 3D image which encloses the transformation of the RGB space. This constraint reinforces the problems linked to the number of data which must be allocated and initialized.

3. Our method

We saw in section 2 that the function $\boldsymbol{Q}$ may be computed thanks to a 3D discrete Voronoi diagram. The main limitation of this method comes from the preprocessing stage, which initializes many useless data. We have thus designed a method which uses a Karhunen-Loève transformation [14] in order to reduce the amount of processed data.

The Karhunen-Loève transformation is the mapping of data points into a fully decorrelated basis (see Figure 2). This basis is equal to the eigenvectors of the covariance matrix. Each eigenvalues of the covariance matrix being equal to the variance of the data set along its associated eigenvectors. The information hold by one eigenvector v_i is often measured by

$$\frac{\lambda_i}{\sum_{j=1}^{n} \lambda_j}$$

where λ_i denotes the eigenvalue of eigenvector v_i and n the dimension of the data set.

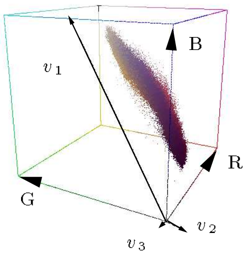

Figure 2: *This figure shows the set of colors of Lenna test image and the 3 eigenvectors (v_1, v_2, v_3) of its covariance matrix. The length of each vector is proportional to its eigenvalue*

Figures 3, 4 and 6 described in this section have been obtained with Girl test image. The colormaps used in these figures have been generated with a fast quantization algorithm [15], their sizes are odd and belong to the range (2,512).

3.1. The projection step

Experiments performed by Otha [16] and confirmed by our own experiments (see Table 1 and 2) show that up to 99% of the information of an image I is contained in the plane $\boldsymbol{P_{princ}}$ defined by the two first eigenvectors of the covariance matrix of the image. Moreover, the transformation matrix from the original color space to the eigenvector system is orthogonal. Thus, the distances computed in the original color space may be efficiently approximated by 2D ones computed in the plane $\boldsymbol{P_{princ}}$. Our method uses this property to approximate the 3D discrete Voronoi diagram associated with the colormap $\{c_1, \dots, c_K\}$ by a 2D diagram defined by the sites $\{p(c_1), \dots, p(c_K)\}$, where p denotes the projections on the plane $\boldsymbol{P_{princ}}$.

The size of the 3D array encoding a 3D discrete Voronoi diagram is defined by the range of each color coordinate. For a 2D discrete Voronoi diagram, the size of the associated 2D array is determined by the box bounding the projections of input image colors on $\boldsymbol{P_{princ}}$. Once the bounding box is determined, the 2D Voronoi diagram may be computed thanks to the method of Danielson [17] using the sites $\{p(c_1), \dots, p(c_K)\}$. The complexity of this method is equal to $\mathcal{O}(|V_I|)$, where $|V_I|$ denotes the size of the 2D array encoding the 2D discrete Voronoi diagram V_I (see Table 3 and Figure 3).

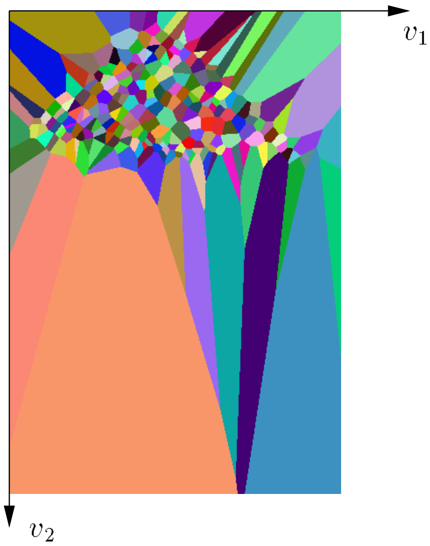

Figure 3: *This figure shows the 2D Voronoi diagram based on the projection of the colormap on $\boldsymbol{P_{princ}}$. It has been produced with Girl test image and a colormap size equal to 256. Note that for high colormap size, the area of most of Voronoi cells become small.*

The computation of V_I implies going through the input image twice: in the first step we compute the

© The Eurographics Association 1998

moments of order 0, 1 and 2 of image colors in order to determine the covariance matrix of the image. Given the covariance matrix, its two first eigenvectors are computed thanks to an incremental method [18]. In the second step we compute the bounding box of the projections of each color of the image on the plane $\boldsymbol{P_{princ}}$. The covariance matrix is immediately deduced from the moments of order 0, 1 and 2 of the image. Moreover, the time required by the computation of the two first eigenvectors is negligible for a 3×3 matrix. Therefore, the computational complexity of the 2D discrete Voronoi diagram is determined by the 2 traversals of the original image and the computation of the 2D discrete Voronoi diagram. This complexity is thus equal to $\mathcal{O}(2|I|+|V_I|)$, where $|I|$ denotes the size of the image and $|V_I|$ the size of the 2D array encoding the 2D Voronoi diagram (see Table 3).

3.2. The correction step

The computation of the 2D discrete Voronoi diagram associated to the sites $\{p(c_1),\dots,p(c_K)\}$ induces two approximations. The first one comes from the projection on $\boldsymbol{P_{princ}}$, which does not take into account the third eigenvector of the covariance matrix. The second approximation is due to the use of a discrete Voronoi diagram, which involves rounding each projection of a given color to its nearest color in V_I. Due to these two approximations, the index of the Voronoi cell which contains the projection of a given color may be different from the index of its nearest representative color (see Figure 4). Nevertheless, these approximations cause minor errors on the distance computations, so that the errors often affect the indexes of two adjacent Voronoi cells. In order to correct such errors, we evaluate, during the computation of the 2D discrete Voronoi diagram V_I, the associated Delaunay graph D_I [19]. These two data structures are then used in the following way :
Given a color c, we read in V_I the index $V_I[p(c)]$ of the Voronoi cell which contains $p(c)$. The set $D_I[V_I[p(c)]]$ contains $V_I[p(c)]$ and the set of indexes of Voronoi cells adjacent to it. Since most of the errors affect two adjacent Voronoi cells, we compute for each color c its nearest representative in $D_I[V_I[p(c)]]$. The index of this representative is given by the function $\boldsymbol{q}$ defined by :

$$\boldsymbol{q}(c) = ArgMin_{i\in D_I[V_I[p(c)]]}\|c_i - c\| \quad (2)$$

where c_i denotes a representative color of the colormap $\{c_1,\dots,c_K\}$ and $\|c_i - c\|$ denotes the Euclidean norm of the 3D vector $c_i - c$.

The color $c_{\boldsymbol{q}(c)}$ is taken as the representative color of c. Note that the index $\boldsymbol{q}$(c) is computed from 3D distances. The 2D Voronoi diagram is only used to restrict the number of distance computations. Figure 5 shows an algorithm using the Delaunay and Voronoi diagrams D_I and V_I to inverse the colormap of an image with the $\boldsymbol{q}$ function.

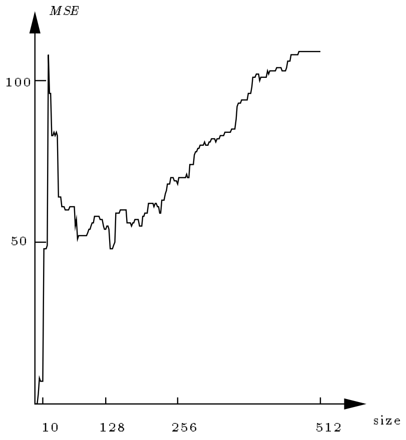

Figure 4: *This figure show the differences between the mean squared error (see equation 4) produced by our algorithm without correction, and the one provided by Heckbert algorithm, which produces an optimal solution. The data set used for this experiment is described in section 3. The increasing aspect of this curve may be explained by the fact that the size of Voronoi cells become smaller as the size of the colormaps increase (see also Figure 3). Thus, classification errors due to our two approximations increase.*

```
inverse_colormap(input image I,
        Voronoi V_I,
        Delaunay D_I)
{
    output image I'
    For each P ∈ I
    {
        proj = p[I(P)]
        ind = ArgMin_{i∈D_I[V_I[proj]]} ||c_i − c||
        I'(P) = c_ind
    }
}
```

Figure 5: *Our inverse colormap algorithm. The symbols V_I and D_I respectively denote the 2D Voronoi and Delaunay diagrams defined in section 3. The symbol p denotes the projection on the plane $\boldsymbol{P_{princ}}$ also defined in section 3. Finally, the symbol c_i denotes the i^{th} representative color of the colormap.*

The complexity of the function $\boldsymbol{q}$ is determined by the number of 2D Voronoi cells adjacent to a given cell. The following theorem determines an upper bound of this complexity :

Theorem 1 Given a 2D Voronoi graph of K sites,

© The Eurographics Association 1998

such as we can't find 4 sites belonging to the same circle. We have the following property :

$$\frac{1}{K}\sum_{i=1}^{K} d_i \leq 6$$

where d_i is the number of cells adjacent to cell i.

Proof If we can't find 4 sites belonging to the same circle, the associated Delaunay graph is a planar triangulation of $\mathbb{R}^2$. In this case, we have [20]:

$$e - 3(v-2) \leq 0$$

where e and v respectively denote the number of edges and the number of vertices of the Delaunay graph. If we denote d'_i the rank of each vertex v_i with $i \in \{1,\ldots,v\}$ we can easily show that:

$$\sum_{i=1}^{v} d'_i \leq 2e$$

Thus :

$$\sum_{i=1}^{v} d'_i \leq 6(v-2) \Rightarrow \frac{\sum_{i=1}^{v} d'_i}{v} \leq 6 - \frac{12}{v} \leq 6$$

We have by construction :

$$\begin{cases} v &= K \text{ and} \\ d'_i &= d_i, \forall i \in \{1,\ldots,K\} \end{cases}$$

Thus :

$$\frac{1}{K}\sum_{i=1}^{K} d_i \leq 6$$

□

Given an input color c, the set $D[V_I[p(c)]]$ includes $V_I[p(c)]$ and the set of indexes of Voronoi cells adjacent to it. The size of $D[V_I[p(c)]]$ is thus equal to $d_{V_I[p(c)]} + 1$, where $d_{V_I[p(c)]}$ denotes the neighborhood size of the Delaunay vertex indexed by $V_I[p(c)]$ (see theorem 1). Therefore, if we denote by S_K the mean number of distance computations required to map a given color to its representative we have :

$$S_K = \frac{1}{K}\sum_{i=1}^{K} d_i + 1 \leq 7 \tag{3}$$

We have seen at the beginning of this section that the complexity of the preprocessing step of our algorithm is equal to $\mathcal{O}(2|I| + |V_I|)$, where $|I|$ and $|V_I|$ denote respectively the size of the input image and the 2D discrete Voronoi diagram. The mean complexity of our algorithm is thus equal to $\mathcal{O}((S_K+2)|I| + |V_I|)$, the worst one being equal to $\mathcal{O}(9|I| + |V_I|)$.

Figure 6 shows the mean neighborhood size for colormap size between 2 and 512 and Girl test image. We observe that the neighborhood size of Voronoi cells varies between 5 and 6 for colormap size greater than 64. Moreover, the mean neighborhood's size is equal to 5.5 (see Table 4). This value being close to the neighborhood's size for colormap size greater than 128, we can consider that S_K is approximately constant and equal to 6.5. The overall complexity of our algorithm may thus be considered as independent of the colormap size for K greater than 128 and equal to $\mathcal{O}(8.5|I| + |V_I|)$. Table 5 summarizes our results on other test images. It shows that these results can be extended to other test images and other colormaps.

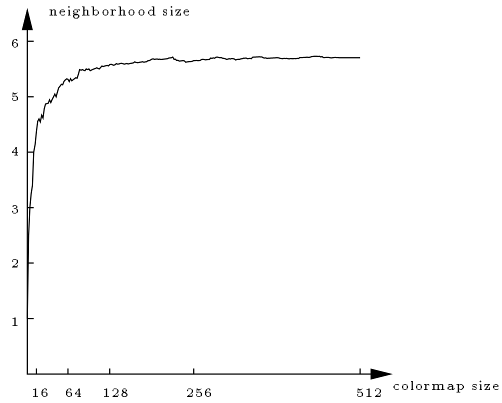

Figure 6: *This figure shows the mean neighborhood size of Voronoi cells plotted as a function of the colormap size. These colormaps have been created thanks to one of our quantization algorithm* [15]*. The size of these colormaps belongs to* $\{2, 4, \ldots, 508, 510, 512\}$.

4. Experiments

We have compared our method to those of Heckbert [5] and Thomas [6] (see section 2). The 5 bits quantization recommended by Thomas has been used in our experiments to test its performances. Heckbert and our method have been tested without this 5 bits quantization since both methods have good performances without this preprocessing step. The first criterion used for our comparisons is the execution time measured on a HP 9000 Workstation. The second criterion is the mean squared error [3, 2] denoted by *MSE* and defined by :

$$MSE = \sum_{P \in I} \|I(P) - c_{\boldsymbol{q}(I(P))}\|^2 \tag{4}$$

where I denotes the input image and $I(P)$ the color of the pixel P in I. The symbol c_i denotes the i^{th}

© The Eurographics Association 1998

representative color. The function $\boldsymbol{q}$ returns an index in the colormap and is defined by equation 2.

Figure 7 shows the linear complexity of Heckbert algorithm. This method is the one which requires the least execution times for colormap size within the range (2,64). However, for greater colormap size, Heckbert's method requires the greatest execution times. According to Thomas the complexity of its algorithm is approximately equal to $\mathcal{O}(\log(n)2^{15} + |I|)$ with the 5 bits quantization, $|I|$ denotes here the size of the original image. This theorical complexity is approximately verified by our experiments. Figure 7 also confirms that the complexity of our algorithm does not depend of the size of the colormap. The increase of our algorithm's curve for very low colormap size within the range (2,16) may be explained by the increase of the mean neighborhood size for such colormaps (see Figure 6). For colormaps size greater than 16, the mean neighborhood size remains approximately constant as the time required by our algorithm does. Note that for colormap size greater then 64, our algorithm the least execution times.

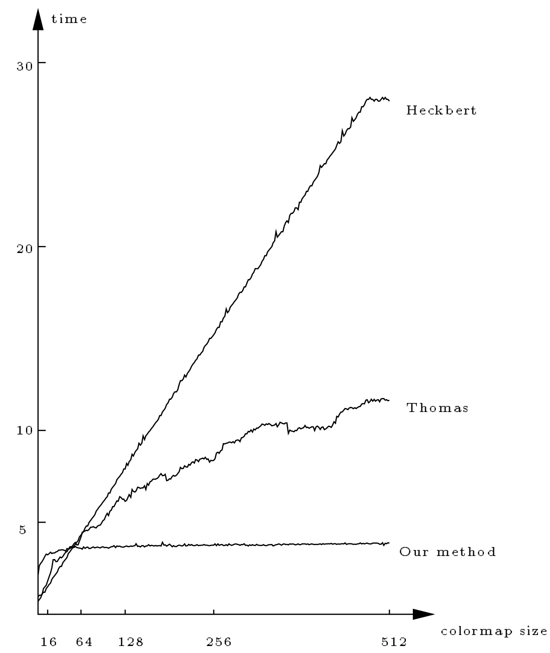

Figure 7: *This Figure show the time in seconds required by Heckbert, Thomas and our algorithms map Girl test image with colormaps described in section 3*

The curves plotted in Figure 8 represent the gap between the mean squared errors produced by Thomas and our algorithms and the one provided by the optimal solution. The curve associated to Heckbert method is not represented in this figure since this method provides the optimal solution for any colormap size.
The use of a quantization into 5 bits for each component by the Thomas method leads to several misclassifications characterized by the non null values of this curve. The smooth increase of the curve with colormaps size may be explained by the mean distance between representative colors which decreases as the size of the colormap increases. These decreasing distances between representative colors increases the effects of the approximation induced by the 5 bits quantization.
The curve associated with our method show that we obtain the optimal solution for colormap size within the range (2,64). This last point confirms the hypothesis made in section 3, which supposes that the index of the representative color of a given color c belong to $D_I[V_I[p(c)]]$. For colormap size greater than 64, the decreasing size of Voronoi cells induces several misclassifications beside our correcting step. However, the difference between the mean squared error produced by our algorithm and the one produced by an optimal method remains lower than 1 which is negligible beside the mean squared error induced by such colormaps which is always greater than 30.

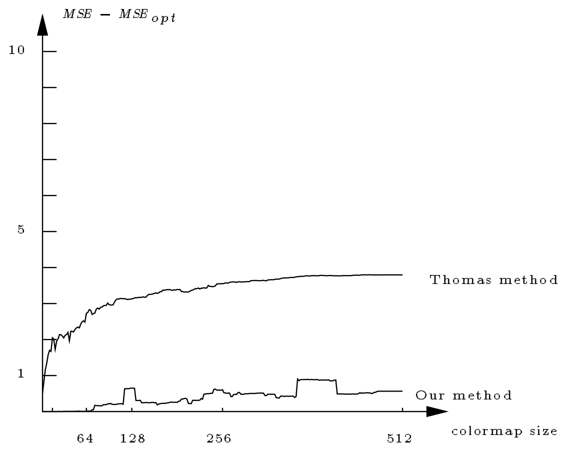

Figure 8: *This Figure show the difference between the mean squared error produced by Thomas and our method with the optimal mean squared error produced by Heckbert algorithms. The image and the colormaps used in this experiment are described in section 3*

In order to validate our results on other data sets we have measured the execution times and the mean squared error of the three algorithms on 4 test images. The colormaps used for these experiments have beeen generated by the same quantization algorithm [15] than the one used for Figures 7 and 8. The size of these colormaps is equal to 16 and 256 in order to test the performances of the three algorithms in the low and high colormap size intervals.

Table 6 illustrates the performances of the 3 algorithms for a colormap of size 16. Results on column (5) show that our algorithm provides the exact solution determined by the Heckbert method [5]. Nevertheless, the time required by our algorithm is greater than that required by the Heckbert or Thomas algorithms. This can be explained by the linear complexity of the Heckbert method which provides efficiency for small colormap sizes. Similarly, Thomas 5 bits quantization has a small influence on the quality of output images when the size of the color map is so small. Indeed, in this case, the representative colors are far enough from each other to drastically reduce the influence of this approximation.

Table 7 shows the performances of the three methods for a colormap size equal to 256. The linear complexity of the Heckbert [5] algorithm induces important processing times for such colormap size. The use of a quantization into 5 bits for each component by the Thomas method leads to several misclassifications characterized by a greater mean squared error (see columns (1) and (3)). Column (5) show that our results are largely equal to the optimal solution. Moreover, results summarized on the last line of Table 7 show that the average processing time of our method is lower to that of Heckbert's and Thomas.

5. Conclusion

Quantization algorithms have been widely studied for the last 20 years. Thanks to this intensive research, recent quantization algorithms create colormaps which represent fairly well the image color distributions. Nevertheless, published methods for inversing colormaps seem to be few and far between. This may be due to the fact that many quantization algorithms create data structures approximating the 3D Voronoi diagram used by inverse colormap methods. In this case, a general inverse colormap algorithm is often useless. Moreover, when the quantization and the inverse colormap processes need to be dissociated the exact solution may be found thanks to the trivial method or Heckbert's.

However, the trivial method and Heckbert's one induce long processing times for large colormap sizes. The Thomas method allows one to efficiently approximate the optimal solution. However, this method produces several errors when the size of the colormap is greater than 256. Finally, the complexity of these 3 methods depends on the size of the colormap.

The complexity of our method depends only on the image size and color distribution. Moreover, unlike the Thomas method, our's may be used in any color space. This last point may be fundamental if the inverse colormap algorithm is used in conjunction with a quantization algorithm using a color space other than the *RGB* one. Our method is mainly devoted to large colormap sizes such as 256 or 512 where it provides results which are equal or close to the optimal solution. Moreover, for such colormap sizes, our algorithm requires lower processing times than the Heckbert and Thomas methods.

References

1. R. Floyd and L. Steinberg, "An adaptative algorithm for spatial greyscale", *Proc. SID*, **17**(2), pp. 75–77 (1976).
2. J. P. Braquelaire and L. Brun, "Comparison and optimization of methods of color image quantization", *IEEE Transactions on Image Processing*, **6**(7), pp. 1048–1052 (1997).
3. Z. Xiang, "Color image quantization by minimizing the maximum intercluster distance", *ACM Transactions on Graphics*, **16**(3), pp. 260–276 (1997).
4. M. Gervautz and W. Purgathofer, "A Simple Method for Color Quantization: Octree Quantization", in *Graphics Gems* (A. S. Glassner, ed.), pp. 287–293, Cambridge, MA: Academic Press Professional, (1990).
5. P. Heckbert, "Color image quantization for frame buffer display", in *Proceedings of SIGGRAPH'82*, pp. 297–307, (1982).
6. S. W. Thomas, "Efficient Inverse Color Map Computation", in *Graphics Gems II* (J. Arvo, ed.), pp. 116–125, 528–535, Cambridge, MA: Academic Press Professional, (1991).
7. F. Aurenhammer, "Voronoi diagrams: a survey of fundamental geometric data structure", *ACM Computing surveys*, **33**(3), pp. 345–405 (1991).
8. E. Bertin, *Diagrammes de Voronoi 2D et 3D : application en analyse d'images.* PhD thesis, Université J. Fourier, Grenoble, (1994).
9. X. Wu, "Color quantization by dynamic programing and principal analysis", *ACM Transaction on Graphics*, **11**(4), pp. 348–372 (1992).
10. S. Wan, S. Wong, and P. Prusinkiewicz, "An algorithm for multidimensional data clustering", *ACM Trans. Math. Softw.*, **14**(4), pp. 153–162 (1988).
11. J. Tajima, "Uniform color scale applications to computer graphics", *Computer Vision, Graphics, and Image Processing*, **21**(3), pp. 305–325 (1983).

© The Eurographics Association 1998

12. N. Ohta, "Correspondence between cielab and cieluv color differences", *Color research and application*, **2**(4), pp. 178–182 (1977).

13. R. M. Boynton, *Human Color Vision.* Optical Society of America, (1992).

14. M. Savoji and R. Burge, "Note : on different methods based on the karhunen-loève expansion and used in image analysis", *Computer Vision, Graphics, and Image Processing*, **29**, pp. 259–269 (1985).

15. L. Brun, "Two high speed color quantization algorithms." Accepted under revisions by Computer & Graphics journal, (1997).

16. Y.-I. Ohta, T. Kanade, and T. Sakai, "Color information for region segmentation", *Computer Vision, Graphics, and Image Processing*, **13**, pp. 222–241 (1980).

17. P.-E. Danielson, "Euclidean distance mapping", *Computer Vision, Graphics, and Image Processing*, **14**, pp. 227–248 (1980).

18. W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C*, ch. 11. Cambridge University Press, (1991).

19. L. de Floriani and E. Puppo, "An on-line algorithm for constrained Delaunay triangulation", *Computer Vision, Graphics, and Image Processing*, **54**(4), pp. 290–300 (1992).

20. F. Harary, *Graph Theory*, ch. 11, p. 118. Addison-Wesley, (1972).

Image	λ_1	λ_2	λ_3	$\frac{\lambda_1+\lambda_2}{\sum_{i=1}^3 \lambda_i}$
Lenna	2212	138	4	99%
Zelda	795	170	27	97%
Girl	2101	307	26	99%
House	1144	206	18	99%

Table 1: *The ratio between the first two eigenvalues λ_1 and λ_2 of the covariance matrix of the test images and the sum of the three eigenvalues λ_1, λ_2 and λ_3*

	Min	*Max*	*Mean*
$\frac{\lambda_3}{\sum_{i=1}^3 \lambda_i}$	0.14%	7.07%	2.56%

Table 2: *The values represented on the 3 columns represent respectively, the min, max and mean value in % of $\frac{\lambda_3}{\sum_{i=1}^3 \lambda_i}$ computed on 105 natural images. Natural image, denotes here, images taken from photographies of natural scene. This term has to be opposed to artificial images which denotes hand written or computed images.*

Image	$\|V_I\|$
Lenna	354×533
Zelda	346×479
Girl	378×548
House	334×533

Table 3: *The size of the 2D arrays V_I on four test images. The array V_I defined in section 3 encodes our 2D discrete Voronoi diagram.*

	Min	*Max*	*Mean*
$\frac{1}{K}\sum_{i=1}^K d_i$	1.0	5.73	5.51

Table 4: *The values represented on the 3 columns represent respectively, the min, max and mean value of $\frac{1}{K}\sum_{i=1}^K d_i$ computed on 255 colormaps. The same colormaps have been used to plot curve 6.*

© The Eurographics Association 1998

Image	16 colors	256 colors
Lenna	4.25	5.7
Zelda	4.4	5.7
Girl	4.3	5.7
House	4.1	5.6

Table 5: *The mean neighborhood size of the 2D Delaunay nodes used by our algorithm. Using* 256 *representative colors, each 2D Voronoi cell is adjacent to about* 6 *other cells. Using* 16 *representative colors this number falls to* 4*.*

	Heckbert		Thomas		`our algorithm`	
	1	2	3	4	5	6
IMAGE	*MSE*	*time*	*MSE*	*time*	*MSE*	*time*
Lenna	131	5.8	132	3.2	131	9.8
Zelda	132	4.8	133	4.5	132	16.4
Girl	172	1.5	173	2.0	172	2.9
House	67	1.3	67	2.0	67	3.2
Mean	125	3.3	125	2.9	125	8.1

Table 6: *The performances of Heckbert, Thomas and our algorithm on* 4 *test images for a colormap size equal to* 16*. The symbol MSE stands for the mean squared error defined by equation 4. Execution times are given in seconds.*

	Heckbert		Thomas		`our algorithm`	
	1	2	3	4	5	6
IMAGE	*MSE*	*time*	*MSE*	*time*	*MSE*	*time*
Lenna	35	53.4	37	9.7	35	12.2
Zelda	26	33.5	27	11.8	26	19.3
Girl	31	12.9	34	8.0	31	3.8
House	17	15.2	19	9.9	18	3.6
Mean	27	28.7	29	9.9	27	9.7

Table 7: *The performances of Heckbert, Thomas and our algorithm on* 4 *test images for a colormap size equal to* 256*. The symbol MSE stands for the mean squared error defined by equation 4. Execution times are given in seconds.*

© The Eurographics Association 1998