# Une Nouvelle méthode d'inversion de tables de couleurs

**Luc Brun et Claude Secroun**

LERI
Institut Universitaire de Reims
Rue des Crayères
51100 Reims, France

brun@univ-reims.fr
secroun@univ-reims.fr

**Résumé :** *L'inversion de la table de couleurs est un processus permettant de créer une image à partir d'une image d'entrée et d'un ensemble de couleurs appelé la table de couleurs ou l'ensemble des couleurs représentatives. Afin de diminuer la distance visuelle entre l'image d'entrée et l'image résultat, ce processus affecte à chaque couleur de l'image originale sa couleur la plus proche dans la table de couleurs. La méthode que nous allons présenter est basée sur une projection de l'ensemble des couleurs de l'image sur un plan discret. L'utilisation de ce plan nous permet d'approximer efficacement le diagramme de Voronoi 3D sous-jacent aux méthodes d'inversion de table de couleurs.*
**Mots-clés :** Couleur, Quantification, table de couleurs

## 1 Introduction

L'accroissement constant des performances des processeurs rend désormais possible le fonctionnement d'applications multi-média sur de petites stations de travail ou des micro ordinateurs. Ces applications nécessitent généralement l'affichage de nombreuses images possédant un nombre de couleurs supérieur au nombre de couleurs simultanément affichables par un écran usuel. Il est donc nécessaire de réduire le nombre de couleurs d'une image avant de l'afficher. Les algorithmes réalisant ce type d'opération sont abusivement appelés des algorithmes de quantification. Ils analysent l'ensemble des couleurs d'une image afin d'en extraire un ensemble de couleurs $\{c_1, \ldots, c_K\}$ représentant au mieux la distribution des couleurs dans l'image.

Une fois cet ensemble déterminé il convient d'afficher l'image avec le nouvel ensemble de couleurs et ce avec une distorsion visuelle minimale. Cette étape est usuellement appelée l'inversion de table de couleurs. Elle consiste à affecter à chaque couleur de l'image sa couleur la plus proche dans l'ensemble $\{c_1, \ldots, c_K\}$. Le calcul de l'ensemble des couleurs plus proche d'une couleur représentative donnée peut s'interpréter comme le calcul du diagramme de Voronoi 3D de germes $\{c_1, \ldots, c_K\}$. Les méthodes utilisant explicitement le calcul du diagramme de Voronoi permettent d'obtenir très rapidement la plus proche couleur représentative d'une couleur donnée. Cependant cette efficacité est compensée par le pré-calcul du diagramme de Voronoi qui induit un surcoût de calcul important.

Nous allons présenter dans la suite de ce document une méthode basée sur une analyse en composantes principales des images à afficher. Cette méthode permet d'approximer efficacement le diagramme de Voronoi 3D défini à partir des couleurs représentatives $\{c_1, \ldots, c_K\}$ par un diagramme de Voronoi 2D plus rapide à calculer. Nous allons décrire dans la section ?? les différentes étapes du processus de quantification en nous attachant plus particulièrement à décrire les principales méthodes utilisées lors de l'étape d'inversion de la table de couleurs. La section ?? décrira les principales étapes de notre méthode. Enfin, la section ?? étudiera les performances des méthodes proposées.

## 2 Problématique de l'inversion de table de couleurs

Les algorithmes de quantification peuvent généralement se décomposer en trois étapes : la quantification, l'inversion de table de couleurs, et les post-traitements destinés a atténuer les artéfacts visuels induits par la réduction du nombre de couleurs de l'image [?].

L'étape de quantification proprement dite permet d'analyser l'ensemble des $n$ couleurs de l'image afin de partitionner cet ensemble en $K$ sous ensembles $\{C_1, \ldots, C_K\}$. La qualité de cette partition est usuellement mesurée par l'erreur quadratique définie par :

$$EQ \;=\; \sum_{i=1}^{K} \sum_{x \in C_i} f(x) \parallel x - c_i \parallel^2 \tag{1}$$

où $f(x)$ représente le nombre de pixels de couleur $x$ dans l'image et $c_i$ le barycentre de l'ensemble $C_i$.

La partition de l'ensemble des couleurs d'une image en sous-ensembles minimisant $EQ$ est un problème $\mathcal{NP}$ complet lorsque la dimension de l'image est supérieure à 1 [?].
Dans le cas d'images couleur la partition de l'ensemble des couleurs de l'image en $K$ sous-ensembles est réalisée à l'aide d'heuristiques utilisant essentiellement trois stratégies :

1. Les méthodes de quantification descendante qui découpent récursivement l'ensemble des couleurs de l'image [?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?].
2. Les méthodes ascendantes qui initialisent $K$ ensembles, puis agrègent chaque couleur de l'image à un de ces ensembles de façon à construire une partition de l'ensemble des couleurs de l'image [?, ?].
3. Les méthodes mixtes qui débutent par une partition de l'ensemble de couleur de l'image en plus d'ensembles que nécessaire puis fusionnent ces ensembles deux à deux de façon à obtenir une partition en $K$ sous ensembles [?, ?].

Une fois la partition de l'ensemble des couleurs de l'image réalisée, l'ensemble des couleurs représentatives est défini comme l'ensemble des barycentres des ensembles $C_i$. On obtient donc un ensemble $\{c_1, \ldots, c_K\}$ appelé l'ensemble des couleurs représentatives ou la table de couleurs de l'image. L'inversion de cette table est réalisée par la fonction $\boldsymbol{Q}$ qui affecte à chaque couleur de l'image sa couleur représentative la plus proche. Si $\{c_1, \ldots, c_K\}$ représente l'ensemble des couleurs représentatives, l'image $\boldsymbol{Q}(c)$ d'une couleur $c$ est définie par :

$$\boldsymbol{Q}(c) = ArgMin_{z \in \{c_1, \ldots, c_K\}} \; ||z - c|| \tag{2}$$

où $Arg\ Minf(t)$ représente la valeur d'un paramètre $t$ réalisant le minimum, et $||z-c||$ la norme du vecteur $z-c$ définie dans un espace métrique donné (par exemple la norme euclidienne dans l'espace $RGB$).

Le calcul des valeurs prises par la fonction $\boldsymbol{Q}$ pour l'ensemble des couleurs d'une image peut être réalisé par une recherche exhaustive du minimum de $||z-c||$ pour tout z dans $\{c_1, \ldots, c_K\}$. Si l'on réalise une quantification d'une image de taille $256 \star 256$ en 256 couleurs, nous devrons effectuer pour chaque pixel de l'image 256 calculs de distances. L'affichage de l'image avec sa nouvelle table de couleurs implique donc $256^3$, soit plus de 16 millons de calculs de distances. Cette méthode nous permet donc d'obtenir une valeur exacte de la fonction $\boldsymbol{Q}$ mais reste peu utilisée en raison du nombre élevé de calculs qu'elle implique.

La méthode d'Heckbert [?], basée sur une découpe uniforme de l'espace de couleurs, permet de rejeter a priori des couleurs représentatives qui ne peuvent réaliser le minimum de l'équation (??). Les tests effectués par Heckbert [?] montrent que le nombre de tests nécessaires pour calculer l'image $\boldsymbol{Q}(c)$ d'une couleur $c$ est d'environ $\mathcal{O}(\frac{K}{23})$, où $K$ est le nombre de couleurs représentatives. Cette méthode permet donc de diminuer la complexité du calcul de la fonction $\boldsymbol{Q}$ sans toutefois changer l'ordre de grandeur du temps de calcul de cette fonction.

On peut aussi diminuer l'ordre de complexité de la fonction $\boldsymbol{Q}$ en utilisant un diagramme de Voronoi 3D [?]. Chaque cellule de Voronoi $V_i$ est alors définie comme l'ensemble des points de l'espace de couleurs plus proches de la couleur $c_i$ que de toute autre couleur représentative. Ce type de méthode permet de pré-calculer de façon incrémentale l'ensemble des valeurs de la fonction $\boldsymbol{Q}$ pour tout point discret $c$ de l'espace de couleurs. L'inconvénient majeur de cette méthode est que l'on effectue bien plus de calculs que nécessaire. En effet, si l'on travaille dans l'espace de couleurs $RGB$ le pré-calcul de la fonction $\boldsymbol{Q}$ impose d'initialiser $256^3 \approx 16.000.000$ valeurs. Si l'image à quantifier a une taille de $256 \star 256$ nous aurons besoin de tester la valeur de la fonction $\boldsymbol{Q}$ pour au plus $256^2 \approx 65.000$ valeurs différentes. Cette méthode peut donc nécessiter de nombreux calculs inutiles. De plus, le stockage de l'ensemble des valeurs de la fonction $\boldsymbol{Q}$ requiert, si l'on utilise 256 couleurs représentatives, 16 méga octets de mémoire. Afin de diminuer la place mémoire requise par l'algorithme on effectue généralement une pré-quantifiquation en ne prenant en compte que les 5 bits de poids fort de chaque composante $(R, G, B)$. Cette méthode diminue la place mémoire requise par l'algorithme et le nombre de valeurs à calculer. Toutefois, malgré cette simplification le pré-calcul de la fonction $\boldsymbol{Q}$ nécessite environ 24 secondes sur une station de travail SUN 3/60 [?]. Sachant que l'obtention d'une partition de l'espace de couleur s'effectue généralement en moins d'une minute, le sur-coût de calcul qu'implique le pré-calcul de la fonction $\boldsymbol{Q}$ est important.

Il est également possible d'approcher le calcul de la fonction $\boldsymbol{Q}$ en affectant à chaque couleur $c$ le barycentre $c_i$ de son ensemble englobant $C_i$. Cette méthode proposée par Friedman et al. [?] est utilisée par la majorité des algorithmes de quantification. L'approximation des cellules de Voronoi $\{V_1, \ldots, V_K\}$ par les ensembles $\{C_1, \ldots, \boldsymbol{C}_K\}$ utilise le fait que la partition de l'espace de couleur en cellules de Voronoi est celle qui minimise l'erreur quadratique définie par l'équation (??) pour un ensemble de couleurs représentatives donné. La partition produite par les algorithmes de quantification étant proche de celle qui minimise l'équation (??), chaque ensemble $C_i$ est proche de la cellule de Voronoi $V_i$ de centre $c_i$ où $c_i$ est le barycentre de $C_i$. Les algorithmes utilisant ce type de méthode utilisent généralement la structure de données utilisée pour stocker la partition de l'ensemble des couleurs de l'image. Par exemple, les méthodes de quantification descendantes stockent généralement l'ensemble des ensembles formant la partition à l'aide d'un arbre binaire. L'utilisation de cet arbre permet alors de retrouver l'ensemble contenant une couleur donnée en $\mathcal{O}(\log_2(K))$ tests. Toutefois ce type de méthode ne peut s'appliquer que lorsque l'étape d'inversion de la table de couleur suit immédiatement l'étape de quantification. Dans le cas contraire, reproduire les structures intermédiaires utilisées par les algorithmes de quantification est souvent plus coûteux que la méthode d'Heckbert [?] ou la recherche exhaustive.

On peut enfin noter que les algorithmes de quantification utilisent souvent d'autres espaces de couleurs que l'espace $RGB$. Le but d'un algorithme de quantification étant de minimiser l'équation (??) selon une métrique donnée, un algorithme d'inversion de table des couleurs utilisant un autre espace et donc une autre métrique brisera la logique qui a conduit à la construction de la table. Il est donc extrèmement souhaitable que l'algorithme d'inversion de table de couleurs et l'algorithme de quantification utilisent le même espace. L'algorithme d'Heckbert [?] et le diagramme de Voronoi 3D discret défini par Thomas [?] utilisent fortement la nature cubique de l'espace de couleurs $RGB$. Ces deux algorithmes sont donc difficilement exploitables lorsque l'on utilise d'autres espaces que l'espace $RGB$.

# 3 Notre méthode d'inversion de table de couleurs

Nous avons vu dans la section ?? que le calcul de la fonction $\boldsymbol{Q}$ pouvait être réalisé à l'aide d'un diagramme de Voronoi 3D discret. Les principales limitations de cette méthode viennent du pré-calcul de la couleur représentative de chaque couleur de l'espace $RGB$. Les travaux menés par Otha [?] et confirmés par nos propres expérimentations (voir table ??) montrent que l'essentiel de l'information d'une image naturelle (par opposition à une image de synthèse) est contenu dans le plan vectoriel $P_{princ}$ défini par les deux premiers vecteurs propres de la matrice de covariance de l'image. Le plan $P_{princ}$ nous permet donc d'approximer efficacement l'ensemble des couleurs de l'image. De plus, la matrice de passage de l'espace de couleurs à l'espace des vecteurs propres étant une matrice orthogonale, les calculs de distances effectués dans l'espace des vecteurs propres seront similaires aux calculs de distances effectués dans l'espace de couleur initial. Il est donc possible d'approximer de manière efficace le diagramme de Voronoi 3D discret de centres $\{c_1, \ldots, c_K\}$ par un diagramme de Voronoi 2D discret défini par les centres $\{p(c_1), \ldots, p(c_K)\}$ où $p(c_i)$ représente la projection de la couleur représentative $c_i$ sur le plan $P_{princ}$. Cette projection est définie par les équations suivantes :

$$\left\{ \begin{array}{lcl} p(c)_{v_1} & = & v_1 \bullet c \\ p(c)_{v_2} & = & v_2 \bullet c \end{array} \right.$$

où $v_1$ et $v_2$ représentent les deux premiers vecteurs propres de la matrice de covariance et $\bullet$ le produit scalaire.

L'utilisation d'un diagramme de Voronoi 2D discret impose de délimiter le domaine sur lequel les calculs de distances d'un point $p(c)$ aux centres $\{p(c_1), \ldots, p(c_K)\}$ seront effectués. A cette fin, nous calculons la boite englobante des projections $p(c)$ des couleurs de l'image sur le plan $P_{princ}$. Le calcul de cette boite englobante s'effectue en déterminant les valeurs $max_1, max_2$ et $min_1, min_2$ définies par :

$$\forall i \in \{1,2\} \left\{ \begin{array}{lcl} min_i & = & Min_{c \in C}\ p(c)_{v_i} \\ max_i & = & Max_{c \in C}\ p(c)_{v_i} \end{array} \right.$$

où $C$ désigne l'ensemble des couleurs de l'image.

Le calcul du diagramme de Voronoi 2D discret peut alors s'effectuer en allouant une image de taille $(max_1 - min_1) \star (max_2 - min_2)$ et en affectant à chaque point discret de l'image l'indice de son centre le plus proche parmis les points $\{p(c_1), \ldots, p(c_K)\}$. Cette initialisation du diagramme de Voronoi 2D discret est effectuée grâce à la méthode incrémentale de Danielson [?] de complexité $\mathcal{O}(|V_I|)$ où $|V_I|$ représente la taille du diagramme de Voronoi 2D discret associé à I (voir Table ??).

Le calcul du diagramme $V_I$ impose donc de parcourir une fois l'image pour calculer les deux vecteurs propres $v_1$ et $v_2$, puis de reparcourir celle ci afin de déterminer les points $(min_1, min_2)$ et $(max_1, max_2)$. La complexité totale du calcul du diagramme $V_I$ est donc égale à $\mathcal{O}(2|I| + |V_I|)$.

Le diagramme de Voronoi $V_I$ associé à une image $I$ nous permet donc d'approximer le calcul de la fonction $\boldsymbol{Q}$ définie dans la section ??. Un algorithme utilisant cette approximation pour réaliser l'inversion de la table de couleurs est donné sur la figure ??.

L'utilisation d'un diagramme de Voronoi 2D discret induit deux types d'approximation : Une approximation due à la projection de l'ensemble des donnés 3D de l'image sur le plan $P_{princ}$ et une approximation due à la discrétisation du plan $P_{princ}$ pour calculer le diagramme de Voronoi 2D. Du fait de ces deux approximations l'algorithme ?? est suceptible de renvoyer un indice différent de celui fourni par la fonction $\boldsymbol{Q}$. Toutefois, les erreurs dues aux approximations induisant de petites erreurs pour le calcul des distances, les erreurs commises par l'algorithme ?? se situent souvent entre deux indices dont les cellules sont adjacentes. Afin de supprimer ce type d'erreur nous déterminons durant le calcul de $V_I$ le diagramme de Delaunay $D_I$ associé à $V_I$. Ces deux structures sont alors combinées de la façon suivante. Soit $V_I[p(c)]$ l'indice de la cellule contenant la projection de $c$. L'utilisation du diagramme de Delaunay nous permet d'acceder à la structure $D_I[V_I[p(c)]]$ contenant l'ensemble des cellules adjacentes à $V_I[p(c)]$. L'utilisation du nouveau diagramme $D_I$ nous permet d'affecter à $c$ la couleur la plus proche de celui-ci parmis l'ensemble des couleurs représentatives dont l'indice appartient à $D_I[V_I[p(c)]]$. Ceci définit une nouvelle fonction $\mathbf{q}$ donnée par l'équation :

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

où $V_I(p(c))$ représente l'indice de la cellule contenant $c$ et $D_I(k)$ l'ensemble des indices dont les cellules sont adjacentes à la cellule d'indice $k$.

Un algorithme utilisant le diagramme de Delaunay $D_I$ et la fonction $\mathbf{q}$ pour inverser la table de couleurs d'une image est représenté sur la figure ??.

La complexité moyenne de la fonction **q** est déterminée par le nombre moyen de cellules adjacentes à une cellule donnée. Les travaux d'Etienne Bertin [?] ont montré que le nombre moyen d'arrêtes d'une cellule de Voronoi est indépendant du nombre de germes et approximativement égal à 6.

Les résultats résumés sur la table ?? montrent de légers écarts par rapports à cette valeur théorique. Ceci est sans doute dû au fait que les travaux d'Etienne Bertin supposent une répartition uniforme des germes. Dans notre cas ceux ci sont répartis de façon priviliégée sur l'axe $v_1$, l'axe $v_2$ ayant par définition une variance moins importante. On peut toutefois noter que les écarts par rapport aux valeurs théoriques restent faibles. Nous pouvons donc considérer que la fonction **q** n'effectue en moyenne que 6 calculs de distances. Sa compléxité est donc constante et indépendante de la taille de la table de couleurs.

| Image | $\lambda_1$ | $\lambda_2$ | $\lambda_3$ | $\frac{\lambda_1+\lambda_2}{\sum_{i=1}^{3}\lambda_i}$ |
|---|---|---|---|---|
| Lenna | 2212 | 138 | 4 | 99% |
| Zelda | 795 | 170 | 27 | 97% |
| Fille | 2101 | 307 | 26 | 99% |
| House | 1144 | 206 | 18 | 99% |

TAB. 1 – *Cette table présente les 3 valeurs propres de quatre images test iso. La dernière colonne représente le ratio entre la somme des deux premières valeurs propres et la somme des valeurs propres. Cette colonne peut s'interpréter comme la quantité d'informations préservée lors de la projection de toutes les couleurs de l'image sur le plan vectoriel défini par les deux premiers vecteurs propres*

| Image | Taille image de distances |
|---|---|
| Lenna | $354 \star 533$ |
| Zelda | $346 \star 479$ |
| Fille | $378 \star 548$ |
| House | $334 \star 533$ |

TAB. 2 – *Cette table présente la taille de l'image utilisée pour calculer le diagramme de Voronoi 2D discret. On peut noter que la taille de ces images reste bien inférieure aux* 16 *millons de valeurs requises par la méthode de Thomas [?].*

| | Cardinal des voisinages. | |
|---|---|---|
| Image | 16 couleurs | 256 couleurs |
| Lenna | 4,25 | 5,7 |
| Zelda | 4,4 | 5,7 |
| Fille | 4,3 | 5,7 |
| House | 4.1 | 5.6 |

TAB. 3 – *Cette table présente le nombre moyen de cellules de Voronoi adjacentes à une cellule donnée. Ces valeurs déterminent la complexité de la fonction* **q**.

# 4 Evaluation des méthodes

Nous avons vu dans la section ?? que les principales méthodes utilisées pour inverser la table des couleurs d'une image étaient la méthode d'Heckbert [?] et le calcul du diagramme de Voronoi

```
inverse_table_couleur1(image I,diagramme Voronoi V_I)
{
    Pour tout pixel P ∈ I
    {
        proj = p[I(P)]
        ind = V_I[proj]
        I(P) = c_ind
    }
}
```

FIG. 1 – *Cette figure présente notre premier algorithme d'inversion de tables de couleurs. Les symboles* $I(P)$ *et* $V_I[proj]$ *représentent respectivement la couleur du pixel* $P$ *et l'indice de la cellule de Voronoi contenant* $proj$*. La projection sur le plan* $P_{princ}$ *est notée* $p$*.*

```
inverse_table_couleur(image I,diagramme Voronoi V_I,diagramme Delaunay D_I)
{
    Pour tout pixel P ∈ I
    {
        proj = p[I(P)]
        ind = ArgMin_{i ∈ D_I[V_I[proj]]} ||c_i - c||
        I(P) = c_ind
    }
}
```

FIG. 2 – *Cette figure présente notre second algorithme d'inversion de tables de couleurs. Les symboles* $I(P)$ *et* $V_I[proj]$ *représentent respectivement la couleur du pixel* $P$ *et l'indice de la cellule de Voronoi contenant* $proj$*.La projection sur le plan* $P_{princ}$ *est notée* $p$*.*

3D discret présenté par Thomas [?]. Nous avons donc comparé nos algorithmes à ces méthodes. Des expérimentations, non présentées dans ce document, nous ont montré que les deux approximations nécessaires à la construction du diagramme de Voronoi 2D discret (voir section ??) induisent un trop grand nombre d'erreurs dans l'algorithme `inverse_table_couleur`1 (voir figure ??). En conséquence nous ne présenterons dans cette section que les performances des méthode d'Heckbert, de Thomas et de l'algorithme `inverse_table_couleur` (voir figure ??).

Les deux critères choisis pour notre comparaison, sont les temps de calculs impliqués par les différentes méthodes et la somme pondérée des carrés des distances entre les images résultats et les images originales. On peut montrer [?] que ce dernier critère est égal à :

$$\frac{1}{|I|} EQ(\{V_1, \ldots, V_K\})$$

où $V_i$ représente l'ensemble des couleurs affectées à la couleur représentative $c_i$, $EQ(\{V_1, \ldots, V_K\})$ l'erreur quadratique (voir équation ??) associée à la partition $\{V_1, \ldots, V_K\}$ et $|I|$ la taille de l'image. Ce critère est appellé l'erreur quadratique moyenne (EQM) associée à $\{V_1, \ldots, V_K\}$.

Les méthodes d'Heckbert et de Thomas impliquant l'utilisation de l'espace $RGB$ (voir section ??) tous les tests présentés dans les tables suivantes ont été effectués dans cet espace. Toutefois, l'algorihme `inverse_table_couleur` n'utilise aucun pré-supposé sur la forme de l'espace de couleur discret. Il peut donc être utilisé avec d'autres espaces de couleurs.

| | Heckbert | | Thomas | | `inverse_table_couleur` | |
|---|---|---|---|---|---|---|
| IMAGE | $EQM$ | $t$ | $EQM$ | $t$ | $EQM$ | $t$ |
| Lenna | 131 | 5.8 | 132 | 3.2 | 131 | 9.8 |
| Zelda | 132 | 4.8 | 133 | 4.5 | 132 | 16.4 |
| Fille | 172 | 1.5 | 173 | 2.0 | 172 | 2.9 |
| House | 67 | 1.3 | 67 | 2.0 | 67 | 3.2 |
| Moyenne | 125 | 3.3 | 125 | 2.9 | 125 | 8.1 |

TAB. 4 – *Les valeurs présentées dans cette table montrent les performances des méthodes d'Heckbert, Thomas et* `inverse_table_couleur` *pour une table de couleurs de taille 16. Les performances qualitatives des différents algorithmes sont mesurées grâce à l'erreur quadratique moyenne (EQM). Les temps de calculs sont donnés en secondes.*

Les résultats résumés sur la Table ?? illustrent les performances des trois méthodes envisagées pour une table de couleurs de taille 16. Les résultats présentés sur l'avant dernière colonne de cette table montrent que l'algorithme `inverse_table_couleur` obtient des résultats identiques à la solution exacte déterminée par l'algorithme d'Heckbert. Ceci confirme l'hypothèse avancée dans la section ??, supposant que les erreurs commises par l'algorithme `inverse_table_couleur`1 s'effectuent presque toujours sur des cellules adjacentes. Les temps requis par l'algorithme `inverse_table_couleur` sont supérieurs aux temps requis par les méthodes d'Hecbert ou de Thomas. L'utilisation de cet algorithme ne peut donc se justifier pour de petites tables de couleurs que si l'on désire obtenir une solution extrêmement proche de la solution exacte en utilisant un autre espace que l'espace $RGB$. Les faibles temps de calcul requis par l'algorithme d'Heckbert s'expliquent par la complexité de cet algorithme ($\mathcal{O}(\frac{K}{23})$) qui lui permet d'être extrêmement efficace pour de petites valeurs de $K$. De même, l'utilisation d'un sous-échantillonage de l'espace de couleurs par l'algorithme de Thomas n'a que peu d'incidence sur les images résultantes lorsque le nombre de couleurs représentatives est aussi réduit. En effet, dans ce cas, les différentes couleurs représentatives sont suffisamment éloignées les unes des autres pour que les erreurs dues au sous-échantillonnage n'aient que peu d'influence.

| | Heckbert | | Thomas | | `inverse_table_couleur` | |
|---|---|---|---|---|---|---|
| IMAGE | $EQM$ | $t$ | $EQM$ | $t$ | $EQM$ | $t$ |
| Lenna | 35 | 53.4 | 37 | 9.7 | 35 | 12.2 |
| Zelda | 26 | 33.5 | 27 | 11.8 | 26 | 19.3 |
| Fille | 31 | 12.9 | 34 | 8.0 | 31 | 3.8 |
| House | 17 | 15.2 | 19 | 9.9 | 18 | 3.6 |
| Moyenne | 27 | 28.7 | 29 | 9.9 | 27 | 9.7 |

TAB. 5 – *Les valeurs contenues dans cette table permettent de mesurer les performances des algorithmes d'Heckbert, de Thomas et* `inverse_table_couleur`*.Les mesures de performances ont été effectuées en utilisant des tables de couleurs de taille 256. Le symbole EQM représente l'erreur quadratique moyenne et donne une estimation de la distance entre l'image résultante et l'image originale. Les temps d'exécution sont donnés en secondes.*

Les résultats présentés table ?? montrent les performances des différentes méthodes pour une table de taille 256. La complexité linéaire de l'algorithme d'Heckbert induit pour des tables de cette taille des temps de calculs importants. L'utilisation d'un sous échantillonnage de l'espace de couleurs par la méthode de Thomas induit, pour des tables de couleurs de grande taille de nombreuses approximations qui ne permettent pas à cette méthode d'atteindre la solution exacte. En revanche, la méthode `inverse_table_couleur` fournit des résultats presque toujours similaires à la solution exacte tout en requérant des temps de calculs inférieurs aux méthodes d'Heckbert et de Thomas. Cette méthode est donc particulièrement intéressante pour des tables de couleur de taille importante. On peut également noter que le temps moyen requis par l'algorithme `inverse_table_couleur` est quasiment constant en fonction du nombre de couleurs de la table (voir Tables ?? et ??) ce qui n'est pas le cas pour les méthodes d'Heckbert ou de Thomas.

# 5 conclusion

Nous avons présenté dans cet article une méthode permettant d'inverser la table de couleurs d'une image. Cette méthode est basée sur une projection de l'ensemble des couleurs de l'image sur le plan $P_{princ}$ défini par les deux premiers vecteurs propres de la matrice de covariance de l'image. L'ensemble de ces projections est alors utilisé pour construire un diagramme de Voronoi 2D discret définissant les cellules de Voronoi associées aux projections des couleurs représentatives. Les expériences décrites dans la section ?? montre que ce diagramme nous permet d'approximer efficacement le diagrame de Voronoi 3D associé aux couleurs représentatives.

Les principaux avantages de cette méthode sont la possibilité d'utiliser un espace de couleurs quelconque, de requérir peu de mémoire, et d'avoir des temps de calculs brefs et indépendants de la taille de la table de couleurs. Enfin cette méthode permet d'obtenir des résultats extrèmement proches de la solution optimale avec des temps de calculs inférieurs aux principales méthodes pour des taille de tables de couleurs importantes.

# Références


[BA91] R. Balasubramanian and J. Allebach. A new approach to palette selection for color images. *Journal of imaging technology*, 17(6):284–290, december 1991.

[BB97] Jean Pierre Braquelaire and Luc Brun. Comparison and optimization of methods of color image quantization. *IEEE Transactions on Image Processing*, 6(7), july 1997.

[BBA94] Raja Balasubramanian, Charles A. Bouman, and Jan P. Allebach. Sequential scalar quantization of color images. *Journal of Electronic Imaging*, 3(1):45–59, January 1994.

[Ber94] E. Bertin. *Diagrammes de Voronoi 2D et 3D : application en analyse d'images.* PhD thesis, Université J. Fourier, Grenoble, 1994.

[BFF77] Jon Louis Bentley, Jerome H. Friedman, and R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Math. Software*, 3:209–226, September 1977.

[Bru96a] Luc Brun. *Segmentation d'images couleur à base Topologique.* PhD thesis, Université Bordeaux I, 351 cours de la Libération 33405 Talence, December 1996.

[Bru96b] Luc Brun. Two high speed color quantization algorithms. To be published in Computer & Graphics journal, may 1996.

[BY95] Jean-Daniel Boissonnat and Mariette Yvinec. *Géométrie algorithmique.* Ediscience international, 28 rue Beaunier 75014 Paris, 1995.

[Dan80] Per-Erik Danielson. Euclidian distance mapping. *Computer Vision, Graphics, and Image Processing*, 14:227–248, 1980.

[Equ89] William H. Equitz. A new vector quantization clustering algorithm. *IEEE transactions on acoustics, speech, and signal processing*, 37(10):1568–1575, october 1989.

[GAW90] Ronald S. Gentile, Jan P. Allebach, and Eric Walowit. Quantization of color images bases on uniform color spaces. *Journal of imaging technology*, 16:11–21, 1990.

[Gol91] Naftaly Goldberg. Colour image quantization for high resolution graphics display. *Image and vision computing*, 9(5):303–312, october 1991.

[GP90] Michael Gervautz and Werner Purgathofer. A simple method for color quantization: Octree quantization. In Glassner, editor, *Graphics Gem*, pages 287–293. Academic Press, 1990.

[Gra84] Robert M. Gray. Vector quantization. *IEEE ASSP Magazine*, 4(2):4–29, April 1984.

[GWA90] Ronald S. Gentile, Eric Walowit, and Jan P. Allebach. Quantization and multilevel halftoning of color images for near-original image quanlity. *J. Opt. Soc. Am. A*, 7(6), 1990.

[HE86] G. Houle and E.Dubois. Quantization of colour images for display on graphics terminals. In *Proc. IEEE Global Telecomun. Conf.*, GLOBE-COM86, pages 1138–1142, december 1986.

[Hec82] Paul Heckbert. Color image quantization for frame buffer display. In *Proceedings of SIG-GRAPH'82*, pages 297–307, 1982.

[OB91] Michael T. Orchard and Charles A. Bouman. Color quantization of images. *IEEE transactions on signal processing*, 39(12):2677–2690, 1991.

[OKS80] Yu-Ichi Ohta, Takeo Kanade, and Toshiyuki Sakai. Color information for region segmentation. *Computer Vision, Graphics, and Image Processing*, 13:222–241, 1980.

[PB96] Yachin Pnueli and Alfred M. Bruckstein. Gridless halftoning : A reincarnation of the old method. *CVGIP: Graphical Models and Image Processing*, 58(1):38–64, Janury 1996.

[Tho91] Spencer W. Thomas. *Efficient inverse color map computation.* Glassner editor, 1991.

[WZ91] 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.

[Xia97] Zhigang Xiang. Color image quantization by minimizing the maximum intercluster distance. *ACM Transactions on Graphics*, 16(3):260–276, July 1997.

[XJ94] Zhigang Xiang and Gregory Joy. Color image quantization by agglomerative clustering. *IEEE Computer Graphics and Applications*, 14(3):44–48, May 1994.