Un algorithme de compression efficace de LUTs 3D couleur basé sur un schéma de diffusion anisotrope multi-échelle

David Tschumperlé, Amal Mahboubi, Christine Porquet

Un algorithme de compression efficace de *LUTs* $3D$ couleur basé sur un schéma de diffusion anisotrope multi-échelle

David TSCHUMPERLÉ, Amal MAHBOUBI, Christine PORQUET

Normandie Univ, UNICAEN, ENSICAEN, CNRS, GREYC,
F-14050 Caen, France
David.Tschumperle@ensicaen.fr, Amal.Mahboubi@unicaen.fr,
Christine.Porquet@ensicaen.fr

Résumé – Les *CLUTs* $3D$ (*Color Look Up Tables*) sont des modèles numériques très utilisés en traitement d'images et de vidéos, pour l'étalonnage couleur, la simulation de films argentiques, et plus généralement pour l'application de transformées colorimétriques quelconques. La dimension élevée de ces modèles pose des problèmes de stockage, lorsque l'on cherche à les distribuer à grande échelle. Nous proposons ici une technique de compression (*avec perte*) très efficace de *CLUTs* $3D$, fondée sur un schéma de reconstruction par diffusion anisotrope multi-échelle. Notre méthode affiche un taux de compression moyen supérieur à 99%, pour une dégradation résultante visuellement indiscernable.

Abstract – $3D$ *CLUTs* (Color Look Up Tables) are popular digital models used in image and video processing for color grading, simulation of analog films, and more generally for the application of various color transformations. The large size of these models leads to data storage issues when trying to distribute them on a large scale. Here, a highly effective lossy compression technique for $3D$ *CLUTs* is proposed. It is based on a multi-scale anisotropic diffusion scheme. Our method exhibits an average compression rate of more than 99%, while ensuring visually indistinguishable differences with the application of the original *CLUTs*.

1 Introduction

Les outils d'étalonnage ou de correction de couleur sont principalement utilisés dans les domaines de la retouche photographique, l'industrie cinématographique, et d'autres disciplines artistiques, afin de modifier les ambiances colorimétriques perçues dans les images numériques. Les *CLUTs* (*Color Look Up Tables*) comptent parmi les modèles les plus populaires pour corriger les couleurs. Notons *RGB*, le domaine continu $[0,255]^3 \subset \mathbb{R}^3$ représentant le cube $3D$ couleur. Une *CLUT* est une fonction colorimétrique dense dans *RGB*, stockée comme un tableau associatif $3D$ encodant la transformation pré-calculée de toutes les couleurs *RGB* existantes.

(a) *CLUT*, vue en $3D$ (b) Image originale (c) Image transformée

FIGURE 1 – Application d'une *CLUT* pour la transformation colorimétrique (ici, pour l'affadissement des couleurs d'une image).

Soit $\mathbf{F} : RGB \to RGB$ une *CLUT* $3D$. L'application de $\mathbf{F}$ sur une image couleur $\mathbf{I} : \Omega \to RGB$ se calcule comme :

$$\forall \mathbf{p} \in \Omega, \quad \mathbf{I}_{(\mathbf{p})}^{\text{modifiée}} = \mathbf{F}(I_{R(\mathbf{p})}, I_{G(\mathbf{p})}, I_{B(\mathbf{p})})$$

où I_R,I_G et I_B sont les composantes couleurs *R*,*G* et *B* de $\mathbf{I}$. Le plus souvent, les *CLUTs* sont des fonctions volumiques *continues* ou, au pire, *continues par morceaux*. Elles permettent la modélisation de transformations colorimétriques très variées, telles que l'affadissement, la dynamisation ou l'inversion de couleurs, l'application de teintes particulières, la conversion en noir et blanc, le rehaussement de contraste, etc. (Fig. 1 et 2).

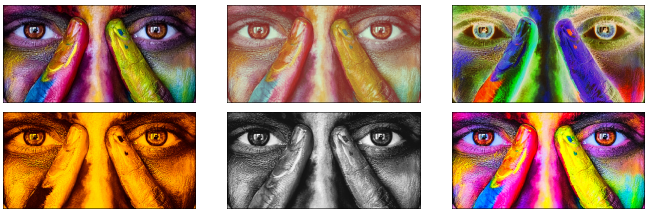


FIGURE 2 – Illustration de la généricité des transformations colorimétriques que permet l'utilisation des *CLUTs* (l'image originale est en haut à gauche).

Usuellement, une *CLUT* est stockée soit sous la forme d'un fichier *ASCII* zippé (extension `.cube.zip`) associant à chaque voxel $\mathbf{X}$ de *RGB* un triplet couleur $\mathbf{F}_{(\mathbf{X})}$, soit comme une image `.png` contenant l'ensemble des couleurs $\mathbf{F}_{(\mathbf{X})}$, déroulées en $2D$. Dans les deux cas, la grande quantité de voxels couleur composant *RGB* implique une taille de stockage pouvant dépasser le méga-octet (*Mo*) pour une unique *CLUT*, même lorsque l'espace *RGB* est sous-échantillonné. Se pose alors la question de l'efficacité de la distribution à grande échelle (plusieurs centaines) de fichiers représentant des *CLUTs*. C'est pour répondre à cette problématique que nous proposons ici une technique de compression *avec perte* de *CLUTs* génériques, ainsi que la méthode de décompression associée. Notre algorithme considère une *CLUT* $\mathbf{F}$ en entrée et en génère une représentation compressée $\mathcal{K}$, telle que sa reconstruction $\widetilde{\mathbf{F}}$ soit suffi-

samment semblable à $\mathbf{F}$ (avec une erreur perceptuellement indiscernable). Il existe étonnamment peu de travaux antérieurs portant sur la compression de *CLUTs*. Une méthode de compression est proposée dans [3], mais elle est *sans perte*. Les expérimentations ne sont réalisées que sur des *CLUTs* de petites tailles (17^3), et affichent des taux de compression ($\approx 30\%$) moins intéressants que ceux que nous allons présenter ici.

En substance, notre technique de compression de *CLUTs* repose sur le stockage d'ensembles de points clés couleur, associé à un algorithme de reconstruction par interpolation $3D$ s'appuyant sur des *EDPs* de diffusion anisotropes. Une idée similaire de reconstruction d'image par diffusion a déjà été proposée dans [6] pour des images $2D$, avec néanmoins des résultats mitigés du fait du caractère discontinu des images naturelles utilisées pour les expérimentations. Pour notre application au contraire, le modèle diffusif s'avère particulièrement approprié pour le calcul de l'interpolation des couleurs dans *RGB*, de par la continuité évidente des fonctions volumiques que l'on cherche à compresser. En section 2, nous détaillons l'algorithme proposé pour la reconstruction de *CLUTs*, puis décrivons le schéma de compression associé (section 3). Notre méthode est validée sur un panel de *CLUTs* variées (section 4).

2 Reconstruction d'une *CLUT* $3D$ à partir d'un ensemble de points clés

Dans un premier temps, on suppose que l'on connaît un ensemble $\mathcal{K} = \{\mathbf{K}_k \in RGB \times RGB \mid k = 1 \dots N\}$ de N points clés couleur, localisés dans le cube *RGB*, tel que $\mathcal{K}$ donne une représentation parcimonieuse d'une *CLUT* $\mathbf{F} : RGB \to RGB$ que l'on souhaite appliquer sur une image ou une vidéo.
Le $k^{\text{ème}}$ point clé de $\mathcal{K}$ est défini par le vecteur

$$\mathbf{K}_k = (\mathbf{X}_k, \mathbf{C}_k) = (x_k, y_k, z_k, R_k, G_k, B_k),$$

où $\mathbf{X}_k = (x_k, y_k, z_k)$ est la position $3D$ du point clé dans le cube *RGB*, et $\mathbf{C}_k = (R_k, G_k, B_k)$ sa couleur associée.

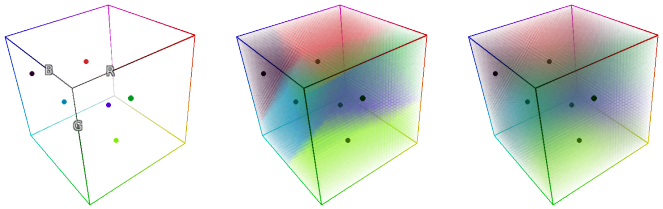
Principe de la reconstruction : Pour reconstruire $\mathbf{F}$ à partir de $\mathcal{K}$, nous proposons de propager / moyenner par diffusion, les couleurs $\mathbf{C}_k$ des points clés, et ce, dans tout le domaine *RGB*. Soit $d_{\mathcal{K}} : RGB \to \mathbb{R}^+$, la fonction indiquant en chaque point $\mathbf{X} = (x, y, z)$ de *RGB*, la distance euclidienne à l'ensemble des points clés $\mathcal{K}$, c-à-d $d_{\mathcal{K}(\mathbf{X})} = \inf_{k \in 0 \dots N} \|\mathbf{X} - \mathbf{X}_k\|$. On réalise alors une reconstruction de $\mathbf{F}$ par la résolution de l'*EDP* de diffusion anisotrope suivante :

$$\forall \mathbf{X} \in RGB, \qquad \frac{\partial \mathbf{F}}{\partial t}(\mathbf{X}) = m_{(\mathbf{X})} \, \frac{\partial^2 \mathbf{F}}{\partial \eta^2}(\mathbf{X}) \qquad (1)$$

$$\text{où} \quad \eta = \frac{\nabla d_{\mathcal{K}(\mathbf{X})}}{\|\nabla d_{\mathcal{K}(\mathbf{X})}\|} \quad \text{et} \quad m_{(\mathbf{X})} = \begin{cases} 0 & \text{si} \ \ \exists k, \ \ \mathbf{X} = \mathbf{X}_k \\ 1 & \text{sinon} \end{cases}$$

D'un point de vue algorithmique, cette résolution peut s'implémenter par une méthode itérative d'*Euler*, en partant d'une estimée initiale $\mathbf{F}_{\text{t=0}}$ la plus proche possible d'une solution de (1). On obtient une estimation raisonnable de $\mathbf{F}_{\text{t=0}}$ en propageant les couleurs $\mathbf{C}_k$ à l'intérieur des cellules de Voronoï $3D$ associées à l'ensemble des points $\mathbf{X}_k$ (par propagation de type *watershed* [4]), puis en lissant le résultat obtenu par un filtre gaussien isotrope en $3D$ (Fig.3b). Nous décrirons par la suite, un schéma multi-échelle plus efficace pour l'estimation de $\mathbf{F}_{\text{t=0}}$.

D'un point de vue géométrique, l'*EDP* de diffusion (1) s'interprète comme un moyenneur local des couleurs le long des lignes reliant chaque point $\mathbf{X}$ du cube *RGB* à son point clé le plus proche [10]. Ce moyennage s'effectue pour tous les points $\mathbf{X}$ de *RGB*, exceptés pour les points clés $\mathbf{X}_k$, qui conservent leur couleur $\mathbf{C}_k$ durant tout le processus de diffusion. La Fig. 3 ci-dessous illustre la reconstruction d'une *CLUT* dense par (1), à partir d'un ensemble $\mathcal{K}$ composé de 6 points clés couleur.



(a) Ensemble de points clés $\mathcal{K}$ (b) Etat initial $\mathbf{F}_{\text{t=0}}$ (Voronoï $3D$ lissé) (c) Etat final (Solution de (1))

FIGURE 3 – Reconstruction par diffusion anisotrope (1) d'une *CLUT* $3D$ $\mathbf{F}$ à partir d'un ensemble de points clés $\mathcal{K}$ (contenant ici $N = 6$ éléments)

Discrétisation spatiale : Numériquement, $d_{\mathcal{K}}$ est calculé en temps linéaire par l'algorithme de transformée en distance [7]. La discrétisation des directions de diffusion η nécessite un peu d'attention, car le gradient de $d_{\mathcal{K}}$ n'est pas défini formellement sur tout le domaine *RGB* ($d_{\mathcal{K}}$ est non dérivable sur ses maxima locaux). Nous proposons par conséquent le schéma numérique suivant pour la discrétisation de $\nabla d_{\mathcal{K}}$:

$$\nabla d_{\mathcal{K}(\mathbf{X})} = \begin{pmatrix} \text{maxabs}(\partial_x^{\text{for}} d_{\mathcal{K}}, \partial_x^{\text{back}} d_{\mathcal{K}}) \\ \text{maxabs}(\partial_y^{\text{for}} d_{\mathcal{K}}, \partial_y^{\text{back}} d_{\mathcal{K}}) \\ \text{maxabs}(\partial_z^{\text{for}} d_{\mathcal{K}}, \partial_z^{\text{back}} d_{\mathcal{K}}) \end{pmatrix} \qquad (2)$$

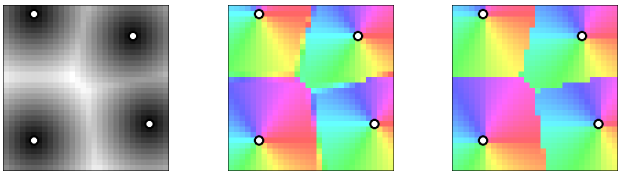
où

$$\text{maxabs}(a, b) = \begin{cases} a & \text{si} \ \ |a| > |b| \\ b & \text{sinon} \end{cases}$$

avec $\partial_x^{\text{for}} d_{\mathcal{K}} = d_{\mathcal{K}(x+1,y,z)} - d_{\mathcal{K}(x,y,z)}$ et $\partial_x^{\text{back}} d_{\mathcal{K}} = d_{\mathcal{K}(x,y,z)} - d_{\mathcal{K}(x-1,y,z)}$, les approximations discrètes *forward* et *backward* de la dérivée première de la fonction continue volumique $d_{\mathcal{K}}$ pour l'axe x (et de même pour les axes y et z). On évite ainsi le calcul d'orientations η erronées sur les crêtes de $d_{\mathcal{K}}$, qui se produit systématiquement avec les schémas numériques usuels *centrés* / *forward* / *backward* pour les dérivées (Fig.4). En pratique, l'utilisation de la discrétisation spatiale (2) a une grande influence sur la qualité de reconstruction de $\mathbf{F}$ (en comparaison avec les schémas classiques qui introduisent des artefacts spatiaux), et sur le temps de convergence de l'*EDP* (1) (l'état stable est en pratique atteint plus rapidement).

Discrétisation temporelle : Pour des raisons d'efficacité algorithmique, nous proposons également d'implémenter l'évolution de l'EDP (1) par le schéma *semi-implicite* suivant :

$$\mathbf{F}_{(\mathbf{X})}^{t+dt} = \frac{\mathbf{F}_{(\mathbf{X})}^{t} + dt \ m_{(\mathbf{X})} \left[\mathbf{F}_{(\mathbf{X}+\eta)}^{t} + \mathbf{F}_{(\mathbf{X}-\eta)}^{t}\right]}{1 + 2 \ dt \ m_{(\mathbf{X})}}$$



(a) Points clés et fonction distance $d_{\mathcal{K}}$ (b) Estimation des η par schéma *centré* (c) Estimation des η avec notre schéma (2)

FIGURE 4 – Influence du schéma numérique pour l'estimation des orientations η (ici sur une portion 40×40 de la fonction distance $d_{\mathcal{K}}$). Les teintes des couleurs affichées correspondent aux orientation des η estimés.

Un intérêt majeur à utiliser un schéma semi-implicite pour (1) réside dans le fait que l'on peut choisir dt arbitrairement grand, sans perte de stabilité ni de qualité sur les résultats de la diffusion (voir [5, 11]). Ce faisant, on obtient alors le schéma approximé suivant pour la discrétisation temporelle de (1) :

$$\begin{cases} \mathbf{F}^{t+dt}_{(\mathbf{X})} = \mathbf{F}^{t}_{(\mathbf{X})} & \text{si } m_{(\mathbf{X})} = 0 \\ \mathbf{F}^{t+dt}_{(\mathbf{X})} = \frac{1}{2}\left[\mathbf{F}^{t}_{(\mathbf{X}+\eta)} + \mathbf{F}^{t}_{(\mathbf{X}-\eta)}\right] & \text{sinon} \end{cases} \quad (3)$$

Les termes $\mathbf{F}^{t}_{(\mathbf{X}+\eta)}$ et $\mathbf{F}^{t}_{(\mathbf{X}-\eta)}$ sont calculés de manière précise (et rapide) par interpolation spatiale tricubique. En partant de $\mathbf{F}_{\text{t=0}}$, le schéma numérique (3) est alors itéré jusqu'à convergence (Fig.3c). Notons que pour chaque itération, le calcul de (3) s'effectue avantageusement en parallèle, puisque le calcul peut se faire indépendamment pour chaque $\mathbf{X} \in RGB$.

Résolution multi-échelle : Comme pour la plupart des schémas numériques d'*EDPs* de diffusion [10], on observe que le nombre d'itérations de (3) requis pour arriver à convergence de (1) augmente quadratiquement avec la résolution spatiale de $\mathbf{F}$. Pour limiter les itérations pour des résolutions de *CLUTs* élevées, nous résolvons donc (1) par une technique *multi-échelle ascendante* : plutôt que d'initialiser $\mathbf{F}_{\text{t=0}}$ par *watershed* pour le calcul à la résolution $(2^s)^3$, on estime $\mathbf{F}_{\text{t=0}}$ comme un agrandissement tri-linéaire de la *CLUT* reconstruite à une résolution deux fois inférieure $(2^{s-1})^3$. Le $\mathbf{F}_{\text{t=0}}$ obtenu s'avère plus proche d'une solution de (1) à la résolution $(2^s)^3$, et le nombre d'itérations nécessaires de (3) pour arriver à convergence se réduit considérablement. En réalisant cette opération en cascade, on peut même débuter la reconstruction de $\mathbf{F}$ à la résolution 1^3 (par simple moyennage des couleurs de tous les points clés), puis appliquer le schéma de diffusion (3) successivement sur les agrandissements des solutions obtenues aux résolutions 2^3, 4^3, 8^3 ..., jusqu'à l'obtention de la taille désirée (Fig.5).

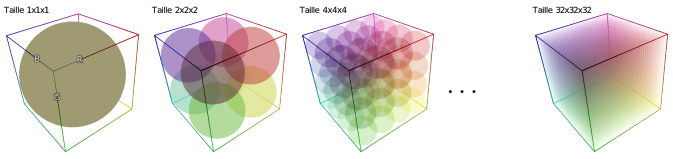


FIGURE 5 – Schéma de reconstruction multi-échelle : Une *CLUT* reconstruite par diffusion à la résolution $(2^s)^3$ est agrandie et sert d'initialisation pour la diffusion effectuée à la résolution supérieure $(2^{s+1})^3$.

Comme la complexité algorithmique d'une seule itération de diffusion à la résolution r^3 s'exprime en $O(r^3)$, la réduction du nombre d'itérations a un impact notable sur le temps effectif de reconstruction. En pratique, 20 itérations par échelle sont suffisantes, ce qui assure une reconstruction parallélisée de *CLUTs* de tailles respectables (64^3) en moins d'une seconde sur une ordinateur récent. L'interpolation $3D$ des points clés par (1) s'avère qualitativement aussi bonne que les techniques classiques d'interpolation à partir d'échantillons épars, telles que les *RBFs* [2], avec surtout une complexité algorithmique moindre (car indépendante du nombre de points clés N).

3 Génération des points clés

Nous savons maintenant reconstruire une *CLUT* complète $\mathbf{F}$ à partir d'un ensemble donné de points clés couleur $\mathcal{K}$. Inversement, en supposant seulement $\mathbf{F}$ connue, est-il possible de trouver un ensemble parcimonieux de points clés $\mathcal{K}$ qui permette une reconstruction de $\mathbf{F}$ de bonne qualité ? Notons que, comme une *CLUT* de résolution r^3 est concrètement stockée comme un tableau discret $3D$, il est bien sûr possible d'en obtenir une reconstruction parfaite en insérant les r^3 voxels couleur de $\mathbf{F}$ dans $\mathcal{K}$. Mais une *CLUT* est une fonction volumique le plus souvent continue, et il est en réalité faisable de la représenter précisément par un nombre de points clés $\mathcal{K}$ *très inférieur à* r^3 ; $\mathcal{K}$ donne alors une représentation *compressée* de $\mathbf{F}$.

L'algorithme de compression proposé ici génère un ensemble $\mathcal{K}$ de N points clés représentant une *CLUT* $\mathbf{F}$ donnée en entrée, telle que la *CLUT* $\widetilde{\mathbf{F}}_N$ reconstruite à partir de $\mathcal{K}$ soit suffisamment proche de $\mathbf{F}$, au sens de deux critères de qualité de reconstruction (paramètres de la méthode) : $\Delta_{\text{max}} = 8$, l'erreur maximale de reconstruction autorisée en un point de *RGB*, et $\Delta_{\text{moy}} = 2$, l'erreur moyenne de reconstruction autorisée pour l'ensemble de $\mathbf{F}$. L'algorithme comporte 3 étapes distinctes :

1. Initialisation : L'ensemble $\mathcal{K}$ est initialisé avec les 8 points clés localisés aux sommets du cube *RGB*, ayant les couleurs de la *CLUT* à compresser, à savoir $\mathcal{K} = \{(\mathbf{X}_k, \mathbf{F}_{(\mathbf{X}_k)} \mid k = 1 \dots 8\}$, pour tous les $\mathbf{X}_k \in RGB$ dont les coordonnées x, y et z valent soit 0, soit 255.

2. Ajout de points clés : Soit $E_N : RGB \to \mathbb{R}^+$, la mesure d'erreur point à point entre la *CLUT* originale $\mathbf{F}$ et la *CLUT* reconstruite $\widetilde{\mathbf{F}}_N$ à partir de $\mathcal{K}$, en utilisant l'algorithme de reconstruction décrit en section 2 : $E_{N(\mathbf{X})} = \|\mathbf{F}_{(\mathbf{X})} - \widetilde{\mathbf{F}}_{N(\mathbf{X})}\|$. On dénote par :

$$E_{\text{max}} = \max_{\mathbf{X} \in RGB}(E_{N(\mathbf{X})}) \quad \text{et} \quad E_{\text{moy}} = \bar{E_N}$$

respectivement l'erreur maximale et l'erreur moyenne de reconstruction. Tant que $E_{\text{max}} > \Delta_{\text{max}}$ ou $E_{\text{moy}} > \Delta_{\text{moy}}$, on ajoute un nouveau point clé $\mathbf{F}_{N+1} = (\mathbf{X}_{N+1}, \mathbf{F}_{N+1(\mathbf{X}_{N+1})})$ à $\mathcal{K}$, localisé aux coordonnées $\mathbf{X}_{N+1} = \text{argmax}_{\mathbf{X}}(E_{N(\mathbf{X})})$. On observe en pratique que ces points clés ajoutés itérativement se dispersent de manière éparse dans *RGB* (Fig.7).

3. Suppression de points clés : Il arrive que l'ajout du dernier point clé à l'étape 2 amène à une reconstruction de *CLUT* de qualité supérieure à celle espérée, c-à-d avec $E_{\text{max}} < \Delta_{\text{max}} - \epsilon$

Nom de la *CLUT*	`Bourbon 64`	`Faded 47`	`Milo 5`	`Cubicle 99`	`Fusion 88`	`Sprocket 231`	`Paladin 1875`
Résolution	16^3	32^3	48^3	64^3	64^3	128^3	144^3
Taille en `.cube.zip`	23.5 Kb	573 Kb	3 Mb	1.2 Mb	1.4 Mb	5.6 Mb	5.4 Mb
Taille en `.png`	3.7 Kb	22 Kb	72 Kb	92 Kb	127 Kb	765 Kb	979 Kb
Nb de points clés	562	294	894	394	210	290	59
PSNR	45.8 dB	45.6 dB	45 dB	45.2 dB	46.1 dB	46.4 dB	43.9 dB
Temps compression	28 s	92 s	1180 s	561 s	257 s	3003.s	1432 s
Temps décompression	67 ms	157 ms	260 ms	437 ms	452 ms	3281 ms	6739 ms
Points clés en `.png`	1.9 Kb	1.5 Kb	4.2 Kb	1.9 Kb	1.3 Kb	1.7 Kb	0.44 Kb
%Gain / `.cube.zip`	92.1%	99.7%	99.8%	99.8%	99.9%	$\approx 100\%$	$\approx 100\%$
%Gain / `.png`	49.5%	93.3%	94.2%	98%	99%	99.8%	$\approx 100\%$

FIGURE 6 – Comparaison de résultats de l'algorithme de compression de *CLUT* proposé, sur diverses *CLUTs* de [1] (avec $\Delta_{\text{max}} = 8$ et $\Delta_{\text{moy}} = 2$).

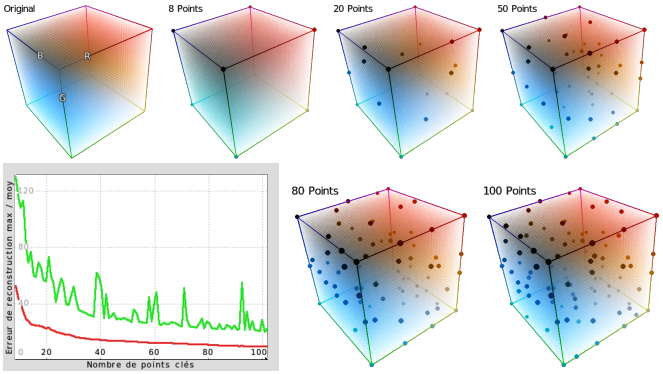


FIGURE 7 – Aperçu des 100 premières itérations de notre algorithme de compression de *CLUTs* $3D$. **En haut :** *CLUT* cible $\mathbf{F}$, et approximation itérative par ajout de points clés. **En bas à gauche :** Evolution de l'erreur maximale (en vert) et moyenne (en rouge) de la *CLUT* reconstruite $\widetilde{\mathbf{F}}_N$.

et $E_{\text{moy}} < \Delta_{\text{moy}} - \epsilon$ et un $\epsilon > 0$ non négligeable. Il existe généralement un sous-ensemble de $\mathcal{K}$ qui vérifie également les critères de qualité, mais avec ϵ plus petit. On peut donc augmenter le taux de compression en préservant la contrainte de qualité, en supprimant certains points clés de $\mathcal{K}$. Ceci se réalise en reparcourant itérativement tous les points clés $\mathbf{K}_k$ de $\mathcal{K}$ (dans l'ordre de leur insertion), et en testant si la suppression du $k^{\text{ème}}$ point clé $\mathbf{K}_k$ ne permet pas de reconstruire une *CLUT* $\widetilde{\mathbf{F}}_N$ respectant toujours les critères de qualité. Si c'est le cas, on supprime $\mathbf{K}_k$ puis on reprend le parcours de $\mathcal{K}$ là où on l'avait laissé. Cette troisième phase permet de retirer jusqu'à 25% de points clés dans $\mathcal{K}$, selon le degré de continuité de la *CLUT* traitée (à l'inverse, il arrive qu'aucun point clé ne soit retiré). À l'issue de ces 3 phases, nous disposons d'un ensemble $\mathcal{K}$ représentant une version compressée avec perte d'une *CLUT* $\mathbf{F}$, telle qu'une qualité minimale de reconstruction soit garantie.

4 Résultats

Notre méthode de compression a été validée sur une base de données comprenant 552 *CLUTs* de résolutions diverses (allant de 33^3 à 144^3), encodant des transformations colorimétriques variées, mises à disposition sur [1, 8]. Elle s'avère étonnamment performante, du fait de l'excellente adéquation du modèle diffusif $3D$ avec le type de données traitées (*CLUTs*). L'ensemble des *CLUTs* occupe initialement 708 Mo, compressé sans perte (593 Mo en `.png` et 115 Mo en `.cube.zip`). La compression de ces données par notre algorithme génère 552 ensembles de points clés, stockés dans un fichier unique de ***2.5* Mo**, soit un taux de compression global de 99.65%. La Fig. 6 fournit des mesures de compression individuelles pour un échantillon de 7 *CLUTs* de [1]. Notons qu'une valeur de *PSNR* minimale de 42.14 dB entre une *CLUT* originale et sa reconstruction est garantie, par le choix du critère de qualité $\Delta_{\text{moy}} = 2$. À des fins de reproductibilité scientifique, nous avons intégré cet algorithme de compression de *CLUTs* dans la plateforme logicielle de traitement d'images *G'MIC*, distribué gratuitement sous licence libre [9]. Nous nous efforçons de faciliter l'intégration de ces algorithmes de compression dans d'autres logiciels de traitement d'images / vidéos, pour permettre la distribution de transformations basées *CLUTs* à une échelle de magnitude bien supérieure aux standards actuels.

Références

[1] RocketStock, 35 Free LUTs for Color Grading (accédé le 2019-02-06). https ://www.rocketstock.com/free-after-effects-templates/35-free-luts-for-color-grading-videos/.

[2] Ken Anjyo, John P Lewis, and Frédéric Pighin. Scattered data interpolation for computer graphics. In *ACM SIGGRAPH 2014 Courses*, page 27.

[3] Aravindh Balaji, Gaurav Sharma, Mark Shaw, and Randall Guay. Preprocessing Methods for Improved Lossless Compression of Color Look-Up Tables. *Journal of Imaging Science and Technology*, 52, 07 2008.

[4] Serge Beucher and Fernand Meyer. The Morphological Approach to Segmentation : The Watershed Transformation. *Optical Engineering-New York-Marcel Dekker Incorporated-*, 34 :433–433, 1992.

[5] Julio M Duarte-Carvajalino, Paul E Castillo, and Miguel Velez-Reyes. Comparative Study of Semi-Implicit Schemes for Nonlinear Diffusion in Hyperspectral Imagery. *IEEE Trans. on Im. Proc.*, 16(5), 2007.

[6] Irena Galić, Joachim Weickert, Martin Welk, Andrés Bruhn, Alexander Belyaev, and Hans-Peter Seidel. Image compression with anisotropic diffusion. *Journal of Math. Imag. and Vision*, 31(2-3) :255–269, 2008.

[7] A. Meijster, J. Roerdink, and W. H Hesselink. A General Algorithm for Computing Distance Transforms in Linear Time. In *Math. Morphology and its App. to Image and Signal Proc.*, pages 331–340. Springer, 2002.

[8] RawTherapee. Film Simulation Pack (accédé le 2019-02-08). https ://rawpedia.rawtherapee.com/Film_Simulation.

[9] David Tschumperlé and Sébastien Fourey. G´MIC : GREYC´s Magic for Image Computing : A Full-Featured Open-Source Framework for Image Processing. https ://gmic.eu/, 2008–2019.

[10] David Tschumperlé and Rachid Deriche. Vector-valued Image Regularization with PDE's : A Common Framework for Different Applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(4) :506–517, 2005.

[11] Joachim Weickert, BM Ter Haar Romeny, and Max A Viergever. Efficient and reliable schemes for nonlinear diffusion filtering. *IEEE transactions on image processing*, 7(3) :398–410, 1998.