An Efficient 3D Color LUT Compression Algorithm Based on a Multi-Scale Anisotropic Diffusion Scheme

David Tschumperlé, Christine Porquet, Amal Mahboubi

An Efficient 3D Color LUT Compression Algorithm Based on a Multi-Scale Anisotropic Diffusion Scheme

David TSCHUMPERLÉ, Christine PORQUET, Amal MAHBOUBI

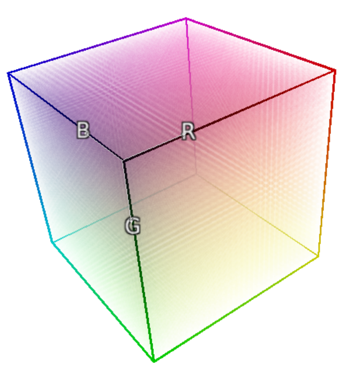
Normandie Univ, UNICAEN, ENSICAEN, CNRS, GREYC,
F-14050 Caen, France
David.Tschumperle@ensicaen.fr, Christine.Porquet@ensicaen.fr,
Amal.Mahboubi@unicaen.fr

Abstract – $3D$ *CLUT*s (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. In this paper, a highly effective lossy compression technique for $3D$ *CLUT*s is proposed. It is based on a multi-scale anisotropic diffusion reconstruction scheme. Our method exhibits an average compression rate of more than 99%, while ensuring visually indistinguishable differences with the application of the original *CLUT*s .

Keywords: $3D$ *CLUT*s , generic color transformations, compression of smooth data, anisotropic diffusion.

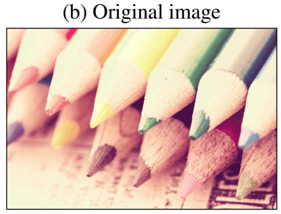
1 Introduction

Color calibration and correction tools are generally used in the fields of photograph retouching, video processing and other artistic disciplines, in order to change the color mood of digital images. *CLUT*s (*Color Look Up Tables*) are among the most popular digital models used for color calibration and alteration. Let *RGB* be the continuous domain $[0, 255]^3 \subset \mathbb{R}^3$ representing the $3D$ color cube (of discretized resolution 256^3). A *CLUT* is a compact color function on *RGB* , modelled as a $3D$ associative array encoding the precomputed transform for all existing colors [1].



(a) *CLUT*, visualized in $3D$

(b) Original image



(c) Image after transformation

FIGURE 1 – Application of a $3D$ *CLUT* to a $2D$ image for a color alteration (here, to simulate vintage color fading).

Let $\mathbf{F} : RGB \rightarrow RGB$ be a $3D$ *CLUT* . Applying $\mathbf{F}$ to a color image $\mathbf{I} : \Omega \rightarrow RGB$ is done as follows :

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

where I_R,I_G and I_B are the *RGB* color components of $\mathbf{I}$. It should be noted that, most often, a *CLUT* is a volumic function that is *continuous* or, at worst, *piecewise continuous* (Fig.1a). Fig.2 exhibits a small set of various colorimetric modifications done with *CLUT*s , taken from [2, 10]. It illustrates the large diversity of the effects that *CLUT*s allow, *e.g.* color fading, chromaticity boost, color inversion, hue shift, black-and-white conversion, contrast enhancement, etc.

Usually, a *CLUT* is stored either as an *ASCII* zipped file (with extension file `.cube.zip`) which maps a color triple $\mathbf{F}_{(\mathbf{X})}$ to each voxel $\mathbf{X}$ of the *RGB* cube (in float-valued format), or as a `.png` image corresponding to the set of all colors $\mathbf{F}_{(\mathbf{X})}$ unrolled as a $2D$ image (Fig.3b). In both cases, the large amount of color voxels composing the *RGB* cube implies a storage size often larger than a megabyte (Mb) for a single *CLUT*, even when the *RGB* space is subsampled (typically to sizes $32^3, 48^3, 64^3, \ldots$). There arises the issue of storing and delivering *CLUT*s files at a large scale (several hundreds at a time).

Here, this issue is addressed : an efficient technique for *CLUT* compression is put forward, as well as the corresponding decompression method. Our algorithm takes a *CLUT* $\mathbf{F}$ as input and generates a smaller representation $\mathbf{F}_c$. The reconstruction algorithm operates on $\mathbf{F}_c$ to generate a reconstructed *CLUT* $\widetilde{\mathbf{F}}$. Our compression scheme is said to be *lossy* [11], as $\widetilde{\mathbf{F}}$ is different from $\mathbf{F}$, but with an error that remains visually unnoticeable.

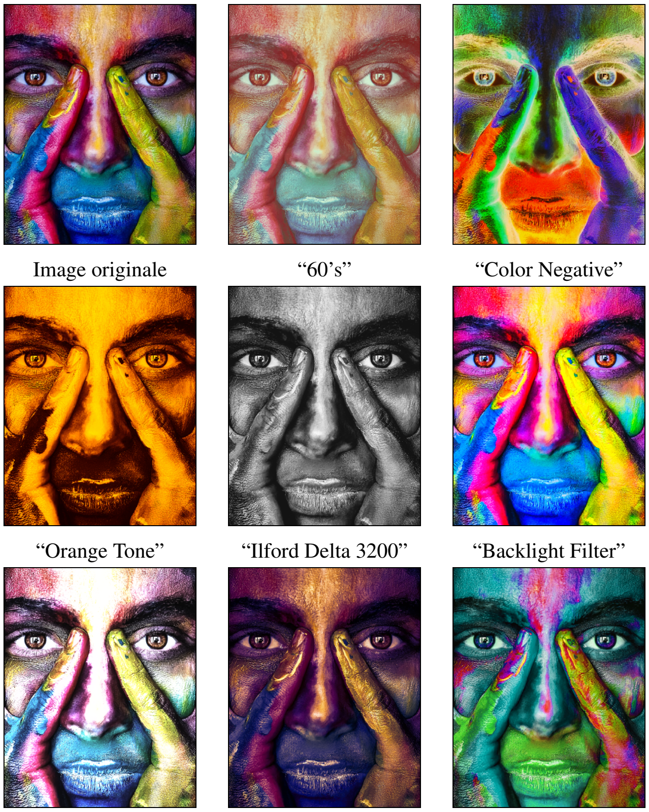


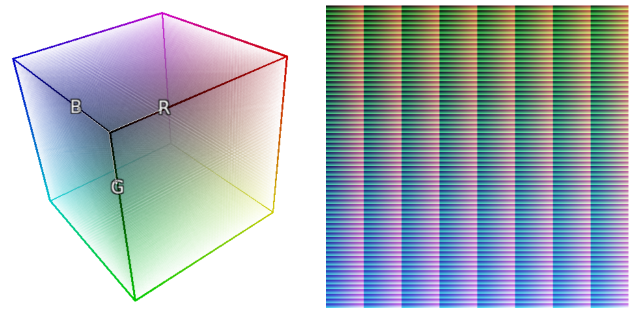
Image originale — "60's" — "Color Negative"
"Orange Tone" — "Ilford Delta 3200" — "Backlight Filter"
"Bleach Bypass" — "Late Sunset" — "Rotate Vibrant"

FIGURE 2 – Generic nature of color transformations allowed by the use of $3D$ *CLUT*s .

Surprisingly, very few references dealing with *CLUT* compression can be found in the literature. In [4], a *lossless CLUT* compression method is proposed ; it is based on two different predictive coding schemes, the former being differential hierarchical coding and the latter cellular interpolative predictive coding. In both cases, a prior preprocessing step for data reorganization is needed. However, experimentations are only made on small-sized *CLUT*s (resolution 17^3), and the *lossless* method leads to compression rates (around 30%) that are much less effective than what we get with our approach.

In essence, our *CLUT* compression technique relies on the storage of a set of color keypoints in *RGB* , associated to a fast interpolation algorithm performing a dense $3D$ reconstruction using anisotropic diffusion *PDEs* . It should be noted that the idea of compressing/decompressing $2D$ image data by diffusion *PDEs* has already been proposed in [8], but the discontinuous aspect of natural images used for their experiments makes it actually harder to achieve high compression rates. In our case, the diffusion model proves to be perfectly suited for interpolating colors in the *RGB* cube, thanks to the clear continuity of the $3D$ dense functions we are trying to compress.

The paper is organized as follows : in Section 2, our *CLUT* reconstruction algorithm is described and the corresponding compression scheme is developed in Section 3. Our method is evaluated on a large variety of *CLUT*s , and compression/reconstruction results are finally discussed in Section 4.



a) *CLUT* visualized in $3D$ — b) Storage as a $2D$ image

FIGURE 3 – Storage of a *CLUT* as a `.png` file : The 64^3 colors of the *CLUT* are here unrolled as a 2D image of size 512^2. Despite the apparent continuity of the $3D$ function, the $2D$ resulting image exhibits lots of discontinuities, which make its compression harder.

2 Reconstruction of a $3D$ *CLUT* from a set of keypoints

First, let us assume we have a set $\mathcal{K} = \{\mathbf{K}_k \in RGB \times RGB \mid k = 1 \dots N\}$ of N color keypoints, located in the *RGB* cube, such as $\mathcal{K}$ provides a sparse representation of a *CLUT* $\mathbf{F} : RGB \rightarrow RGB$.
The k^{th} keypoint of $\mathcal{K}$ is defined by vector

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

where $\mathbf{X}_k = (x_k, y_k, z_k)$ is the $3D$ keypoint position in the *RGB* cube and $\mathbf{C}_k = (R_k, G_k, B_k)$ its associated color.

Reconstruction scheme : In order to reconstruct $\mathbf{F}$ from $\mathcal{K}$, we propose to propagate/average the colors $\mathbf{C}_k$ of the keypoints in the whole *RGB* domain through a specific diffusion process. Let $d_{\mathcal{K}} : RGB \rightarrow \mathbb{R}+$ be the distance function, giving for each point $\mathbf{X} = (x, y, z)$ of *RGB* , the Euclidian distance to the set of keypoints $\mathcal{K}$, i.e.

$$\forall \mathbf{X} \in RGB, \qquad d_{\mathcal{K}(\mathbf{X})} = \inf_{k \in 0 \dots N} \|\mathbf{X} - \mathbf{X}_k\|$$

$\mathbf{F}$ is reconstructed by solving the following anisotropic diffusion *PDE* :

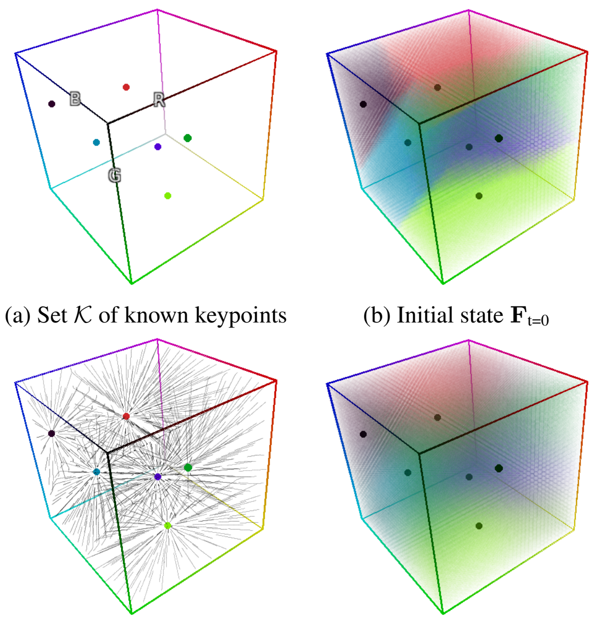
$$\forall \mathbf{X} \in RGB, \qquad \frac{\partial \mathbf{F}}{\partial t}(\mathbf{X}) = m_{(\mathbf{X})}\, \frac{\partial^2 \mathbf{F}}{\partial \eta^2}(\mathbf{X}) \tag{1}$$

where $\eta = \dfrac{\nabla d_{\mathcal{K}(\mathbf{X})}}{\|\nabla d_{\mathcal{K}(\mathbf{X})}\|}$ and $m_{(\mathbf{X})} = \begin{cases} 0 & \text{if } \exists k,\ \mathbf{X} = \mathbf{X}_k \\ 1 & \text{otherwise} \end{cases}$

From an algorithmic point of view, this *PDE* can classically be solved by an *Euler* method, starting from an initial estimate $\mathbf{F}_{\text{t=0}}$ as close as possible to a solution of (1). A quite good estimate for $\mathbf{F}_{\text{t=0}}$ is actually obtained by propagating the colors $\mathbf{C}_k$

inside the Voronoï cells associated to the set of points $\mathbf{X}_k$ (for instance by *watershed*-like propagation [5]), then by smoothing it by an isotropic $3D$ gaussian filter (Fig.4b). A more efficient multi-scale scheme for estimating $\mathbf{F}_{\text{t=0}}$ is detailed hereafter.

From a geometric point of view, the diffusion *PDE* (1) can be seen as a local color averaging filter along the lines connecting each point $\mathbf{X}$ of the *RGB* cube to its nearest keypoint [14]. This filtering is done for all points $\mathbf{X}$ of *RGB* , except for the keypoints $\mathbf{X}_k$ which keep their initial color $\mathbf{C}_k$ throughout the diffusion process. Fig.4 below shows the reconstruction of a dense *CLUT* with (1), from a set $\mathcal{K}$ composed of 6 colored keypoints.



(a) Set $\mathcal{K}$ of known keypoints (b) Initial state $\mathbf{F}_{\text{t=0}}$

(c) Diffusion orientations η (d) State at convergence

FIGURE 4 – Reconstruction of a $3D$ *CLUT* $\mathbf{F}$ from a set of keypoints $\mathcal{K}$ using anisotropic diffusion PDE (1) (here, from 6 keypoints).

Spatial discretization : Numerically, $d_{\mathcal{K}}$ is efficiently computed (in linear time) by a distance transform, such as the one proposed in [9]. The discretization of the diffusion directions η requires some care, as the gradient $\nabla d_{\mathcal{K}}$ is not formally defined on the whole *RGB* domain. Actually, $d_{\mathcal{K}}$ is not differentiable at the peaks of the distance function, *i.e.* at the points that are local maxima. Therefore, the following numerical scheme for the discretization of $\nabla d_{\mathcal{K}}$ is put forward :

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

where

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

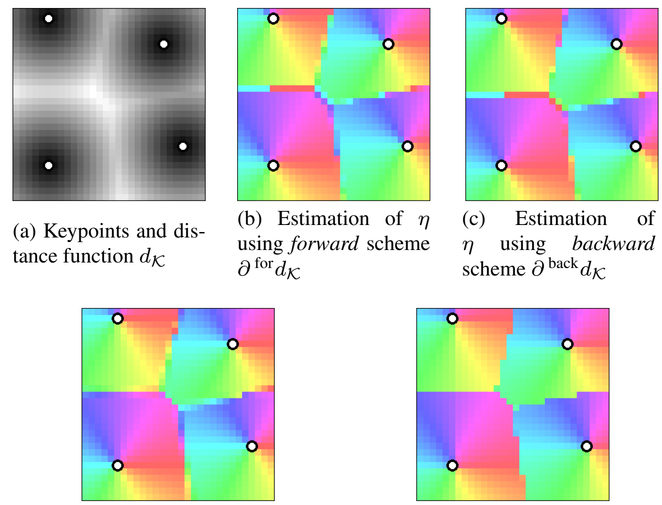
and

$$\partial_x^{\text{for}} d_{\mathcal{K}} = d_{\mathcal{K}(x+1,y,z)} - d_{\mathcal{K}(x,y,z)}$$

$$\partial_x^{\text{back}} d_{\mathcal{K}} = d_{\mathcal{K}(x,y,z)} - d_{\mathcal{K}(x-1,y,z)}$$

are the discrete *forward* and *backward* first derivative approximations of the continuous function $d_{\mathcal{K}}$ along the x axis. We proceed similarly along the y and z axes.

By doing so, one avoids locally misdirected estimations of η on the local maxima of $d_{\mathcal{K}}$, which systematically happen with the *centered*, *forward* or *backward* numerical schemes classically used for estimating the gradient, as shown on Fig.5 below.



(a) Keypoints and distance function $d_{\mathcal{K}}$ (b) Estimation of η using *forward* scheme $\partial^{\text{ for}} d_{\mathcal{K}}$ (c) Estimation of η using *backward* scheme $\partial^{\text{ back}} d_{\mathcal{K}}$

(d) Estimation of η using *centered* scheme $\frac{1}{2}(\partial^{\text{ for}} d_{\mathcal{K}} + \partial^{\text{ back}} d_{\mathcal{K}})$ (e) Estimation of η using *proposed* scheme (2)

FIGURE 5 – Influence of our scheme for estimating the diffusion orientations η (shown here on a small 40×40 crop of the distance function $d_{\mathcal{K}}$). Hues displayed at each point represent the estimated orientations η.

In practice, complying to our spatial discretization scheme (2) has a great influence, both on the reconstruction quality of the *CLUT* $\mathbf{F}$ (in comparison with usual discretization schemes introducing visible artifacts on reconstructed structures), and on the effective time of convergence towards the solution of (1). A stable state is reached more quickly. This is particularly true with the use of the multi-scale scheme described hereafter, where reconstruction errors may be amplified when switching from a low resolution scale to a more detailed one.

Temporal discretization : For the sake of algorithmic efficiency, the explicit *Euler* scheme corresponding to the evolution of (1) becomes the following *semi-implicit* scheme :

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

which leads to :

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

A major advantage of using such a *semi-implicit* scheme to implement the evolution of (1) is that you can choose dt arbitrarily large, without loss of stability or significant decrease in quality on the diffusion process (as studied in [6, 15]). Therefore, we get the following simplified temporal discretization scheme :

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

where $\mathbf{F}^{t}_{(\mathbf{X}+\eta)}$ and $\mathbf{F}^{t}_{(\mathbf{X}-\eta)}$ are accurately estimated using tricubic spatial interpolation.

Starting from $\mathbf{F}_{\text{t=0}}$, the scheme (3) is iterated until convergence (Fig.4d). It should be noted that, for each iteration, the computation of (3) can be advantageously parallelized, as the calculations are done independently for each voxel $\mathbf{X}$ of *RGB* .

Multi-scale resolution : As with most numerical schemes involving diffusion *PDEs* [14], it can be observed that the number of iterations of (3) required to converge towards a stable solution of (1) increases quadratically with the $3D$ resolution of the *CLUT* $\mathbf{F}$ to be reconstructed. In order to limit this number of iterations for high resolutions of *CLUT*s , we therefore suggest to solve (1) by a *multi-scale ascending* technique.

Rather than initializing $\mathbf{F}_{\text{t=0}}$ by *watershed*-like propagation for computing the diffusion at resolution $(2^s)^3$, $\mathbf{F}_{\text{t=0}}$ is estimated as a trilinear upscaling of the *CLUT* reconstructed at half resolution $(2^{s-1})^3$. The latter is closer to the stable state of the *PDE* (1) at resolution $(2^s)^3$, and the number of necessary iterations of (3) to reach convergence is considerably reduced. By performing this recursively, it is even possible to start the reconstruction of $\mathbf{F}$ at resolution 1^3 (by simply averaging the colors of all keypoints), then applying the diffusion scheme (3) successively on the upscaled results obtained at resolutions 2^3, 4^3, 8^3 . . . , until the desired resolution is reached (Fig.6).

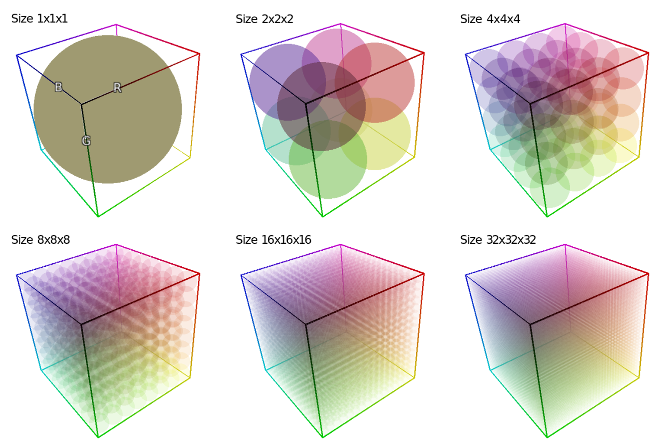


FIGURE 6 – Multi-scale reconstruction scheme : A reconstructed *CLUT*s at resolution $(2^s)^3$ is linearly upscaled and used as an initialization for applying the diffusion scheme at a higher resolution $(2^{s+1})^3$.

Comparison with *RBF* reconstruction : The reconstruction of a dense function from a set of isolated keypoints is an interpolation problem which has been already well documented in the literature [3, 12]. Most traditional solutions to this problem propose to model the function to be reconstructed as a weighted sum, whose number of terms is equal to the number of available keypoints. For instance, the popular *RBF* (*Radial Basis Function*) method applied to *CLUT* reconstruction estimates each color component $\mathbf{F}^i$ of $\mathbf{F}$ ($i = R, G$ or B) by :

$$\forall \mathbf{X} \in RGB\ , \quad \mathbf{F}^i_{(\mathbf{X})} = \sum_{k=1}^{N} w^i_k\ \phi(\|\mathbf{X} - \mathbf{X}_k\|),$$

with $\phi : \mathbb{R}^+ \to \mathbb{R}$, a given function (e. g. $\phi(r) = r^2 \ln r$, for a *thin plate spline* interpolation [7]). The weights w^i_k are obtained by solving a linear system, involving the known values of the keypoints $\mathbf{C}_k$ and a matrix whose coefficients are $\phi(\|\mathbf{X}_p - \mathbf{X}_q\|)$, calculated for all possible pairs (p, q) of keypoints. This reconstruction technique generates $3D$ interpolations of good quality, and is simple to implement, as it can be calculated directly at full resolution. Unfortunately, its algorithmic complexity is expressed as $O(N^3 + N\ r^3)$ for the reconstruction of a *CLUT* of resolution r^3, which becomes prohibitive when the number of keypoints increases notably (e.g. $N > 300$, which happens frequently in our case, see Fig. 8).

Conversely, the complexity of one single iteration of our diffusion scheme (3) is expressed as $O(r^3)$, regardless of the number of keypoints. Thanks to our multi-scale approach that speeds up convergence towards a stable state, no more than twenty diffusion iterations per reconstruction scale are necessary in practice. This ensures a reconstruction of a decent size *CLUT* (e.g. resolution 64^3) in less than one second on a standard multi-core computer (for several tens of seconds with a *RBF* approach), and this, with an equally good reconstruction quality.

3 Generation of keypoints

Now that the reconstruction of a dense CLUT $\mathbf{F}$ from a set of color keypoints $\mathcal{K}$ has been detailed, let us consider the inverse problem, *i.e.* given only $\mathbf{F}$, is it possible to find a sparse set of keypoints $\mathcal{K}$ that allows a good quality reconstruction of $\mathbf{F}$?

First of all, it is worth mentioning that a *CLUT* being practically stored as a $3D$ discrete array, it is always possible to build a set $\mathcal{K}$ allowing an *exact discrete reconstruction* from $\mathbf{F}$ at resolution r^3, by simply inserting all the r^3 color voxels from $\mathbf{F}$ as keypoints in $\mathcal{K}$. But as a *CLUT* is most often a continuous function, it is actually feasible to represent it fairly accurately by a set of keypoints $\mathcal{K}$ the size of which is *much less than the number of voxels* composing the discrete cube *RGB* . $\mathcal{K}$ then gives a *compressed* representation of $\mathbf{F}$.

The compression algorithm detailed hereafter generates a set $\mathcal{K}$ of N keypoints representing a given input *CLUT* $\mathbf{F}$, such that the *CLUT* $\widetilde{\mathbf{F}}_N$ reconstructed from $\mathcal{K}$ is close enough to $\mathbf{F}$, in the sense of two reconstruction quality criteria (which are set as parameters of the method). These quality criteria are chosen

as : $\Delta_{\max} = 8$, the maximum reconstruction error allowed at one point of *RGB* , and $\Delta_{\text{avg}} = 2$, the average reconstruction error for the entire *CLUT* $\mathbf{F}$.
The algorithm consists of three distinct steps :

1. Initialization : The set $\mathcal{K}$ is initialized with the 8 keypoints located at the vertices of the *RGB* cube, with the colors of the *CLUT* to be compressed, *i.e.* $\mathcal{K} = \{(\mathbf{X}_k, \mathbf{F}_{(\mathbf{X}_k)}) \mid k = 1 \dots 8\}$, for all $\mathbf{X}_k$ whose coordinates in x, y and z are either 0 or 255.

2. Adding keypoints : Let $E_N : RGB \rightarrow \mathbb{R}^+$ be the point-to-point error measurement between the original *CLUT* $\mathbf{F}$ and the *CLUT* $\widetilde{\mathbf{F}}_N$ reconstructed from $\mathcal{K}$, using the algorithm described in Section 2 :

$$E_{N(\mathbf{X})} = \|\mathbf{F}_{(\mathbf{X})} - \widetilde{\mathbf{F}}_{N(\mathbf{X})}\|$$

where

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

respectively denote the maximum error and the average reconstruction error. As long as $E_{\max} > \Delta_{\max}$ or $E_{\text{avg}} > \Delta_{\text{avg}}$, a new keypoint $\mathbf{F}_{N+1} = (\mathbf{X}_{N+1}, \mathbf{F}_{N+1(\mathbf{X}_{N+1})})$ is added to $\mathcal{K}$, located at coordinates $\mathbf{X}_{N+1} = \text{argmax}_{\mathbf{X}}(E_N)$ of the maximum reconstruction error. In practice, one can observe that these keypoints added iteratively are scattered throughout the entire *RGB* domain, so as to jointly minimize the two criteria of reconstruction quality $\Delta_{\max}$ and Δ_{avg} (Fig.7).

3. Deleting keypoints : Sometimes, the addition of the last keypoint at step 2 leads to a *CLUT* reconstructed with an higher quality than expected, *i.e.* with $E_{\max} < \Delta_{\max} - \epsilon$ and $E_{\text{avg}} < \Delta_{\text{avg}} - \epsilon$ and a non negligible $\epsilon > 0$. In this case, there is usually a subset of $\mathcal{K}$ that also verifies the reconstruction quality criteria, with an ϵ closer to 0. We can therefore try to increase the compression rate while maintaining the desired quality of reconstruction, by removing a few keypoints from $\mathcal{K}$. This is simply achieved by iteratively going through all the keypoints $\mathbf{K}_k$ of $\mathcal{K}$ (in the order of their insertion, $k = 1 \dots N$), and checking whether the deletion of the k^{th} keypoint $\mathbf{K}_k$ allows to reconstruct a *CLUT* $\widetilde{\mathbf{F}}_N$ with quality constraints that still hold. If this is the case, the keypoint $\mathbf{K}_k$ is discarded from $\mathcal{K}$ and the algorithm is resumed from where we left it. According to the degree of continuity of the *CLUT* , this third step sometimes allows to withdraw up to 25% of keypoints in $\mathcal{K}$ (it also happens that no keypoint can be removed this way).

At the end of these three steps, we get a set or keypoints $\mathcal{K}$ representing a compressed lossy version of a *CLUT* $\mathbf{F}$, such that a minimum quality of reconstruction is guaranteed.

4 Results

The performance of our compression method has been evaluated on publicly available datasets (including [2, 10]) for a total of 552 CLUTs at various resolutions (ranging from 33^3 to 144^3) and encoding very diverse color transformations. In our case, the relevant measurement is the *compression rate*, defined as :

$$\%cRate = 100 \left(1 - \frac{\text{Size of } \textit{compressed} \text{ data}}{\text{Size of } \textit{input} \text{ data}}\right)$$

The set of all the original *CLUT* data occupies 708 Mb of disk storage (including 593 Mb in `.png` format and 115 Mb in `.cube.zip` format). The compression of this large dataset by our algorithm generates 552 sets of keypoints, stored in a single **2.5 Mb** file, representing then an overall compression rate of 99.65% (despite the fact that the input dataset itself is already in a compressed form !). A statistical study of the sets of keypoints indicates that the average number of keypoints is 1078 (minimum : 35, maximum : 2047, standard deviation : 587), which is high enough to make our fast *PDE* -based reconstruction technique more suitable than *RBFs* .

The table in Fig.8 provides individual compression measurements for a sample of 7 *CLUTs* taken from [2]. It shows the compression rates for various *CLUTs* at different resolutions (our sets of N keypoints being stored as color `.png` images at resolution $2 \times N$), with respect to the input *CLUT* data stored in the usual way, i.e. compressed files in `.png` and `.cube.zip` formats. It is interesting to note that the number of generated keypoints does not depend on the resolution of the *CLUT* to be compressed, but rather on its *degree of continuity* (the keypoints being naturally located on the most discontinuous areas

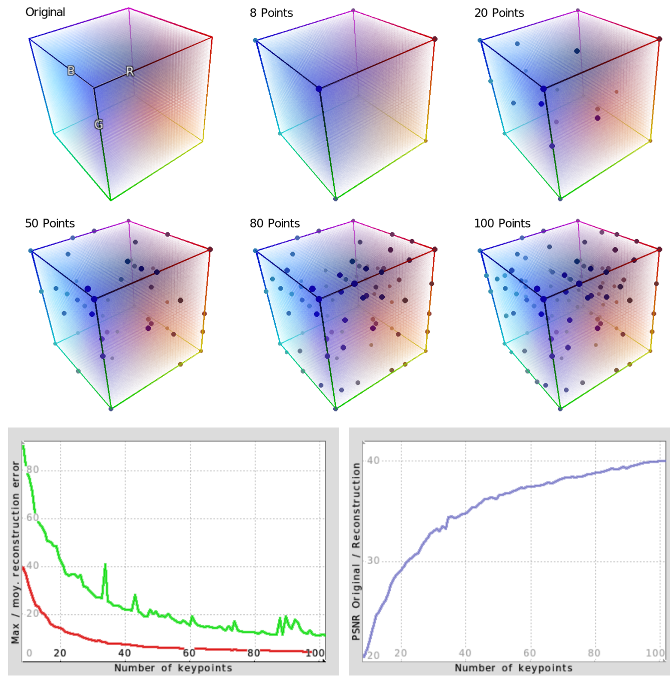


FIGURE 7 – Overview of the first 100 iterations of our $3D$ *CLUTs* compression algorithm. **Top :** Target *CLUT* $\mathbf{F}$ and approximations by iteratively adding keypoints. **Bottom :** Evolution of the maximum error (in green) and average error (in red), and of the *PSNR* (in blue) of the reconstructed *CLUT* $\widetilde{\mathbf{F}}_N$ with respect to the target *CLUT* $\mathbf{F}$.

CLUT name	`Bourbon 64`	`Faded 47`	`Milo 5`	`Cubicle 99`	`Fusion 88`	`Sprocket 231`	`Paladin 1875`
Resolution	16^3	32^3	48^3	64^3	64^3	128^3	144^3
Size in `.cube.zip`	23.5 Kb	573 Kb	3 Mb	1.2 Mb	1.4 Mb	5.6 Mb	5.4 Mb
Size in `.png`	3.7 Kb	22 Kb	72 Kb	92 Kb	127 Kb	765 Kb	979 Kb
Number of keypoints	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
Compression time	28 s	92 s	1180 s	561 s	257 s	3003.s	1432 s
Decompression time	67 ms	157 ms	260 ms	437 ms	452 ms	3281 ms	6739 ms
Keypoints in `.png`	1.9 Kb	1.5 Kb	4.2 Kb	1.9 Kb	1.3 Kb	1.7 Kb	0.44 Kb
$\%cRate$/`.cube.zip`	92.1%	99.7%	99.8%	99.8%	99.9%	$\approx$ 100%	$\approx$ 100%
$\%cRate$/`.png`	49.5%	93.3%	94.2%	98%	99%	99.8%	$\approx$ 100%

FIGURE 8 – Results of our CLUT compression algorithm, on different *CLUT*s from [2] (with $\Delta_{\max} = 8$ et $\Delta_{\text{avg}} = 2$).

of the *CLUTs*, Fig.7).
By limiting the average reconstruction error, the quality criterion $\Delta_{\text{avg}} = 2$ ensures a minimal value of 42.14 dB for the *PSNR* between an input *CLUT* $\mathbf{F}$ and its compressed reconstruction $\widetilde{\mathbf{F}}$. In theory, this criterion alone is not enough to guarantee visually imperceptible differences. However, this is the case in practice, as our algorithm simultaneously takes into account another quality criterion $\Delta_{\max} = 8$ which limits the maximum reconstruction error.

For the purpose of scientific reproducibility, our *CLUT* compression/decompression algorithms have been integrated into *G'MIC*, a full-featured open-source framework for image processing [13].

5 Conclusions

The *CLUT* compression/decompression techniques we presented in this paper are surprisingly effective. This is mainly due to the perfect adequacy of the proposed $3D$ diffusion model (1) to the type of data processed (smooth, volumetric, color-valued). As a result, all the 552 *CLUT*s compressed by our method and integrated into *G'MIC* [13] make it, to the best of our knowledge, the image editing software that offers photographers and illustrators the greatest diversity of color transformations, and this, *for a minimal storage cost*. We are convinced that the integration of these algorithms into other image or video processing software will trigger the distribution of *CLUT*-based color transformations at a much larger magnitude scale than current standards.

Références

[1] Explanation of what a 3D CLUT is (accessed 2019-02-08). http ://www.quelsolaar.com/technology/clut.html.

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

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

[4] 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.

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

[6] Julio M Duarte-Carvajalino, Paul E Castillo, and Miguel Velez-Reyes. Comparative Study of Semi-Implicit Schemes for Nonlinear Diffusion in Hyperspectral Imagery. *IEEE Transactions on Image Processing*, 16(5) :1303–1314, 2007.

[7] Jean Duchon. Splines minimizing rotation-invariant semi-norms in sobolev spaces. In *Constructive theory of functions of several variables*, pages 85–100. Springer, 1977.

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

[9] Arnold Meijster, Jos BTM Roerdink, and Wim H Hesselink. A General Algorithm for Computing Distance Transforms in Linear Time. In *Mathematical Morphology and its applications to image and signal processing*, pages 331–340. Springer, 2002.

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

[11] David Salomon and Giovanni Motta. *Handbook of Data Compression*. Springer Publishing Company, Incorporated, 5th edition, 2009.

[12] Joel A Tropp and Anna C Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Transactions on information theory*, 53(12) :4655–4666, 2007.

[13] 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.

[14] 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.

[15] 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.