Reconstruction of Smooth 3D Color Functions from Keypoints: Application to Lossy Compression and Exemplar-Based Generation of Color LUTs
David Tschumperlé, Christine Porquet, Amal Mahboubi

To cite this version:

HAL Id: hal-02940557
https://hal-normandie-univ.archives-ouvertes.fr/hal-02940557
Submitted on 19 Nov 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Copyright
Reconstruction of Smooth 3D Color Functions from Keypoints: Application to Lossy Compression and Exemplar-Based Generation of Color LUTs

David Tschumperlé, Christine Porquet, Amal Mahboubi
David.Tschumperle@ensicaen.fr Christine.Porquet@ensicaen.fr Amal.Mahboubi@unicaen.fr
Normandie Univ, UNICAEN, ENSICAEN, CNRS, GREYC, F-14000 Caen, France
19th November 2020

Figure 1: Our approach is able to generate a compressed representation of a generic, nonparametric, smooth RGB → RGB mapping, modeled as a 3D CLUT (a). We extract a set of relevant keypoints (b,c) that sparsely describes the CLUT structure, and provides a related CLUT decompression algorithm from those keypoints. Thus, hundreds of such color mappings can be stored at low cost. (e) shows the application of the original (left) and compressed (right) versions of the 3D CLUT (a) to color image (d). No perceptual difference is noticeable.

Abstract

3D CLUTs (Color Look Up Tables) are popular digital models used in artistic image and video processing, for color grading, simulation of analog films, and more generally for the description and application of generic nonparametric color transformations. The relatively large size of these models leads to high data storage requirements when trying to distribute them on a large scale (e.g. several hundred at a time). In this article, an effective technique based on a multiscale anisotropic diffusion scheme is proposed, for the lossy compression of generic CLUTs regularly sampled on a 3D grid. Our method exhibits high average compression rate, while ensuring visually indistinguishable differences with the original (uncompressed) CLUTs. In a second step, a variation of our algorithm for exemplar-based generation of CLUTs is developed, in order to create a complete CLUT from a single pair of before/after images that accounts for the color transformation.
Keywords: 3D Color LUTs, generic color transformations, compression of smooth data, anisotropic diffusion, exemplar-based CLUT generation.

1 Introduction

Color retouching tools are commonly used in the fields of digital photography, video processing and other artistic disciplines, to allow artists to set up a particular color mood for their creations. In this context, nonparametric 3D CLUTs (Color Look Up Tables) are among the most popular and versatile models for color enhancement or alteration of digital images and videos.

Let $RGB$ be the continuous domain $[0, 255]^3 \subset \mathbb{R}^3$ representing the 3D color cube of discretized resolution $256^3$. This $256^3$ resolution is chosen for the sake of simplicity, but in practice, other resolutions are actually considered. A CLUT is simply a compact color function on $RGB$, modeled as a 3D associative array encoding the precomputed transformation of all existing $RGB$ colors ([10] and Fig. 2a).

Let $F : RGB \rightarrow RGB$ be a 3D CLUT. Applying $F$ to a color image $I : \Omega \rightarrow RGB$ (defined on a rectangular domain $\Omega \subset \mathbb{R}^2$) is done as follows:

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

where $I_{R}$, $I_{G}$ and $I_{B}$ are the $RGB$ color components of $I$. It should be noted that a CLUT is a volumetric function that is, most often, fully or piecewise continuous with inflection points (Fig. 2a).

Fig. 3 exhibits a small set of various colorimetric changes that can be made with CLUTs, taken from the CLUT packs [19, 21, 26]. It illustrates the large diversity of effects that CLUTs allow, e.g. color fading, chromaticity boost, color inversion, hue shift, black-and-white conversion, contrast enhancement, etc.

In practice, for image and video retouching purposes, a CLUT is usually available, either as an ASCII zipped file (extension .cube.zip) that describe the mapping of color triples $F(\mathbf{X})$ for each voxel $\mathbf{X}$ of the $RGB$ cube (with a floating-point syntax), or as a .png image corresponding to the set of all transformed colors $F(\mathbf{X})$ of $RGB$, unrolled as a 2D image (Fig. 4b). In both cases, the high number of color voxels composing the $RGB$ cube leads
Figure 3: Generic nature of color transformations allowed by the use of 3D CLUTs.

to a quite large storage size for a single CLUT: For instance, a 256³-sized CLUT, which represents 48 megabytes (Mb) of uncompressed data, would be stored as a ≈ 2 Mb file in .png format (with lossless compression), and as a 176 Mb file in .cube.zip format (the latter being known to be an inefficient format in terms of storage space). Hence, CLUTs are generally downscaled to a regularly subsampled RGB space (typically with resolutions 17³, 33³, 48³, 64³, . . .) in order to lower the number of color voxels and thus, the file size. Missing color entries are further estimated by 3D linear or tricubic interpolations.

ICC profiles [14] are another file format that can be used for describing color mappings. ICC is an industry standard mainly used to characterize the color conversions when dealing with inputs/outputs of external devices (scanners, monitors, printers, etc.). It generally deals with parametric color conversions but can also store non-parametric CLUT transformations, through the ICC device-link profiles. These profiles do not explicitly handle
compressed colorimetric transformations, although the compression of such profiles has recently been considered [27]. As a result, a CLUT in an ICC device-link profile is stored as an uncompressed array of raw data (32 bits-float for each RGB component of the CLUT voxels).

The 64^3 colors of the CLUT are unrolled as a 512^2 image. Despite the apparent continuity of the 3D function, the resulting image exhibits lots of discontinuities, which make it harder to compress in 2D.

Figure 4: Storage of a downscaled CLUT as a .png file: The 64^3 colors of the CLUT are unrolled as a 512^2 image. Despite the apparent continuity of the 3D function, the resulting image exhibits lots of discontinuities, which make it harder to compress in 2D.

Actually, only a few references about CLUT compression can be found in the literature. Most existing methods [2, 3, 25] deal with lossless compression and are based on two different predictive coding schemes: [2, 3] propose differential hierarchical coding while [25] deals with cellular interpolated predictive coding. In both cases, a prior preprocessing step for data reorganization is needed. However, experiments are only made on small-sized CLUTs (resolution 17^3), which somehow limits the complexity of color variations these CLUTs can handle. Such lossless compression techniques lead to average compression rates (≈ 30% that are below what the storage in compressed .png generally offers, which is around 70 – 75%). To our knowledge, there are only a few existing lossy compression techniques targeted on CLUTs [27, 31]. In [27], a DCT-quantization scheme in 3D is proposed to compress CLUT data, in a very similar way to JPEG compression. In our previous conference paper [31], we proposed a CLUT compression technique that relies on the storage of a set of sparse color keypoints in RGB, associated to a fast interpolation algorithm performing a dense 3D reconstruction using anisotropic diffusion PDEs (Fig. 1). This method is able to obtain high average compression rates (≥ 95%), while reconstructing CLUTs with very few artefacts when comparing them with the original data. It should be noted that the idea of compressing/decompressing image data by diffusion PDEs has already been proposed for 2D images in [12], but the discontinuous aspect of natural images used for the experiments makes it actually harder to achieve high compression rates. In the case of CLUT data, the non-linear diffusion model proposed in [31] proves to be very well suited for interpolating colors in the RGB cube, thanks to the overall smooth aspect of the 3D dense color functions we are considering for compression. Moreover, the free localization of keypoints allows to capture, when needed, local geometric discontinuities.

In this article, our previous work [31] is reviewed and extended, and two different image processing applications are tackled:

1. **CLUT compression**, for storing and delivering large sets of nonparametric CLUT files (several hundreds at a time). To achieve this goal, we first review the technique for generic CLUT compression/reconstruction
proposed in [31] (section 2). This compression algorithm takes a CLUT $F$ as input and generates a much smaller representation $F_c$ such that $F$ can be reconstructed from $F_c$ at its full resolution. The compression scheme is said to be lossy [24], as the reconstructed CLUT $\tilde{F}$ slightly differs from $F$, but with an error that remains small enough. Then, the study is extended to new experiments and comparisons between different color metrics and keypoint determination methods as well as comparisons with other alternatives for data reconstruction (resampling, harmonic functions, RBFs), in sections 3.1 and 3.2. Finally, an experimental validation of the compression of more than 500 CLUTs is done, yielding a global compression rate higher than 99%.

2. Exemplar-based CLUT generation, i.e the construction of a complete CLUT from a unique pair of before/after images that exhibits the color transformation. It enables an artist doing color calibration or retouching to generate his own CLUT only from a single processed example, so that the customized colorimetric transform is easily reproducible by other users, with different image/video retouching software. Note that this is a slightly different task than doing color transfer between images (as done for instance in [11, 22, 20]), since the purpose here is not to explicitly transfer colors from one image to another (this transformation has already been done manually by the artist), but rather to extract a CLUT that can reproduce the same kind of color transformations on a whole series of images (section 4).

Finally, conclusions are drawn in the last section 5.

2 A PDE-based approach to 3D CLUT compression

In this section, the approach described in our previous work [31] for the compression of CLUT is detailed. The proposed compression algorithm relies on two distinct steps, that are: 1. a 3D CLUT reconstruction algorithm, based on a diffusion scheme working on a set of keypoints, and 2. a keypoint determination step, designed as a simple iterative greedy algorithm. Since the reconstruction method is a prerequisite to the compression stage, let us start by the former.

2.1 Reconstruction from a set of sparse color keypoints

First, let us consider we already have a set $K$ of colored keypoints $K_k \in RGB \times RGB$, defined as ordered pairs:

$$K_k = (X_k, C_k) = ((x_k, y_k, z_k), (R_k, G_k, B_k))$$

where $X_k = (x_k, y_k, z_k)$ is the 3D keypoint position in the RGB cube and $C_k = (R_k, G_k, B_k)$ its associated RGB color.

We assume that the set $K = \{K_1, K_2, \ldots, K_N\}$ already provides a sparse representation of a given CLUT $F$.

2.1.1 Diffusion scheme

In order to reconstruct $F$ from $K$, we propose to propagate/average the colors $C_k$ of the keypoints in the whole RGB domain through a specific diffusion process. Using diffusion is indeed a natural way to ensure smoothness of the reconstructed CLUT. Let $d_K : RGB \to \mathbb{R}^+$ be the distance function, giving for each point $X = (x, y, z)$ of RGB, the Euclidian distance to the set of keypoints $K$, i.e.

$$\forall X \in RGB, \quad d_K(X) = \min_{k \in \{0 \ldots N\}} \|X - X_k\|.$$ 

We propose to reconstruct $F$ by solving the following anisotropic diffusion PDE:

$$\forall X \in RGB, \quad \frac{\partial F}{\partial t}(X) = m(X) \frac{\partial^2 F}{\partial \eta^2}(X), \quad (1)$$
where $\eta = \frac{\nabla d_K(X)}{\|\nabla d_K(X)\|}$ and $m(X) = \begin{cases} 0 & \text{if } \exists k, X = X_k \\ 1 & \text{otherwise} \end{cases}$

At each point, the diffusion strength $m(X)$ is chosen to be maximal (i.e. constant), it only vanishes on the location of known keypoints, where the diffusion process must be locally stopped to ensure each keypoint color keeps its values over time.

From an algorithmic point of view, this PDE can classically be solved by an Euler method, starting from an initial estimate $F_{t=0}$ as close as possible to a solution of (1). Actually, one can get a quite good estimate for $F_{t=0}$ by propagating the colors $C_k$ inside the Voronoi cells associated to the set of points $X_k$ (for instance by watershed-like propagation [4]), then by smoothing it by an isotropic 3D gaussian filter (Fig. 5b). A more efficient multiscale scheme for estimating $F_{t=0}$ is detailed hereafter (section 2.1.4).

From a geometric point of view, the diffusion PDE (1) can be seen as a local color averaging filter [30] along the lines connecting each point $X$ of the RGB cube to its nearest keypoint. This filtering is done for all points $X$ of RGB, except for the known keypoints $X_k$ which keep their initial color $C_k$ throughout the diffusion process. In particular, this kind of diffusion PDEs ensures that the min/max values of the resulting colors remain within the $[\min, \max]$ range of the original values, which naturally avoids the generation of out-of-gamut values. It should also be noted that the color components reconstructed by this diffusion process are real numbers ($\in \mathbb{R}$).

In practice, it is thus possible to generate CLUTs with a color depth greater than 8 bits per channel (10 or 12 bits, as typically found in post-production). Fig. 5 below shows the reconstruction of a dense CLUT with (1), from a set $K$ composed of 6 colored keypoints.

2.1.2 Spatial discretization

Numerically, $d_K$ is efficiently computed (in linear time) by a distance transform, such as the one proposed in [18]. The discretization of the diffusion directions $\eta$ requires some care, as the gradient $\nabla d_K$ is not formally defined on the whole RGB domain. Actually, $d_K$ 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 $\nabla d_K$ is proposed:

$$\nabla d_K(X) = \begin{pmatrix} \maxabs(\partial_x^{\text{for}} d_K, \partial_x^{\text{back}} d_K) \\ \maxabs(\partial_y^{\text{for}} d_K, \partial_y^{\text{back}} d_K) \\ \maxabs(\partial_z^{\text{for}} d_K, \partial_z^{\text{back}} d_K) \end{pmatrix}$$

where

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

and

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

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

are the discrete forward and backward first derivative approximations of the continuous function $d_K$ along the $x$ axis. We proceed similarly along the $y$ and $z$ axes.

By doing so, one avoids locally misdirected estimations of $\eta$ on the local maxima of $d_K$, which systematically happen with the centered, forward or backward numerical schemes classically used for estimating the gradient, as shown on Fig. 6 below; classical gradient estimation schemes (b,c,d) result in inaccurate gradient orientations on local maxima, whereas with our scheme (e), we get an accurate orientation everywhere.
In practice, complying to our spatial discretization scheme (2) has a great influence, both on the reconstruction quality of the CLUT $\mathbf{F}$ (usual discretization schemes introduce 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 multiscale scheme described hereafter, where reconstruction errors may be amplified when switching from a low resolution scale to a more detailed one.

2.1.3 Temporal discretization

For the sake of algorithmic efficiency, the explicit Euler scheme corresponding to the evolution of (1) is transformed to the following semi-implicit scheme:

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

which leads to:

$$\mathbf{F}_{t+dt}(\mathbf{X}) = \mathbf{F}_{t}(\mathbf{X}) + dt \frac{m(\mathbf{X})}{1+2 dt m(\mathbf{X})} \left[ \mathbf{F}_{t}(\mathbf{X}+\eta) + \mathbf{F}_{t}(\mathbf{X}-\eta) \right]$$
Figure 6: Influence of our numerical schemes for estimating the diffusion orientations $\eta$ (shown here on a small $40 \times 40$ crop of a 2D distance function $d_K$). Hues displayed at each point represent the estimated orientations $\eta$.

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 [8, 32]). Therefore, by choosing $dt$ large enough, we end up with the following simplified temporal discretization scheme:

$$F_{t+dt}(X) = \begin{cases} F_t(X) & \text{if } m_t(X) = 0 \\ \frac{1}{2} \left[ F_t(X+\eta) + F_t(X-\eta) \right] & \text{otherwise} \end{cases}$$

(3)

where $F_t(X+\eta)$ and $F_t(X-\eta)$ are accurately estimated using tricubic spatial interpolation. Starting from $F_{t=0}$, the scheme (3) is iterated until convergence (Fig. 5d). 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 $X$ of RGB.

### 2.1.4 Multiscale solving

As with most numerical schemes involving diffusion PDEs [30], it can be observed that the number of iterations of (3) required to converge towards a stable solution of (1) quadratically increases with the 3D resolution of the CLUT $F$ to be reconstructed. In order to limit this number of iterations for high resolutions of CLUTs, we therefore suggest to solve (1) by a multiscale ascending technique.
Rather than initializing \( F_{t=0} \) by watershed-like propagation for computing the diffusion at resolution \((2^s)^3\), \( F_{t=0} \) 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 \( F \) 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 \( 2^3, 4^3, 8^3 \ldots \), until the desired resolution is reached (Fig. 7).

Note also that for each different resolution, the coordinates \( X_k \) of the color keypoints are downscaled accordingly, and rounded to the nearest integers.

Figure 7: Multiscale reconstruction scheme: A reconstructed CLUT at resolution \((2^s)^3\) is linearly upscaled and used as an initialization for applying the diffusion scheme at a higher resolution \((2^{s+1})^3\).

### 2.2 A greedy compression algorithm

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

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 \( K \) allowing an exact discrete reconstruction from \( F \) at resolution \( r^3 \), by simply inserting all the \( r^3 \) color voxels from \( F \) as keypoints in \( K \). But as a CLUT is most often a continuous function, it is actually feasible to represent it fairly accurately by a set of keypoints \( K \) the size of which is much less than the number of voxels composing the discrete cube \( RGB \). \( K \) then gives a compressed representation of \( F \).
2.2.1 Generation of 3D color keypoints

The compression algorithm detailed hereafter generates a set $\mathcal{K}$ of $N$ keypoints representing a given input CLUT $\mathbf{F}$, such that the CLUT $\tilde{\mathbf{F}}_N$ reconstructed from $\mathcal{K}$ is close enough to $\mathbf{F}$. Our approach is bottom-up: it starts from an empty set $\mathcal{K}$ and iteratively add keypoints to it. At any given iteration, we denote by $E_N : \text{RGB} \rightarrow \mathbb{R}^+$ the point-to-point error measurement between the original CLUT $\mathbf{F}$ and the CLUT $\tilde{\mathbf{F}}_N$ reconstructed from $\mathcal{K}$, using the algorithm previously described in Section 2.1. For the sake of simplicity $E_N$ is defined here as the $L_2$-error between $\mathbf{F}$ and $\tilde{\mathbf{F}}_N$, i.e.

$$\forall \mathbf{x} \in \text{RGB}, \quad E_N(\mathbf{x}) = \| \mathbf{F}(\mathbf{x}) - \tilde{\mathbf{F}}_N(\mathbf{x}) \|$$

The set $\mathcal{K}$ is populated until two reconstruction quality criteria are met, which are $E_{\text{max}} \leq \Delta_{\text{max}}$, the maximum reconstruction error allowed at one point of RGB, and $E_{\text{avg}} \leq \Delta_{\text{avg}}$, the average reconstruction error for the entire CLUT $\mathbf{F}$, with

$$E_{\text{max}} = \max_{\mathbf{x} \in \text{RGB}} E_N(\mathbf{x}), \quad \text{and} \quad E_{\text{avg}} = \bar{E}_N.$$

$\Delta_{\text{max}} \in \mathbb{R}^+$ and $\Delta_{\text{avg}} \in \mathbb{R}^+$ are the two main parameters of the compression method, and are chosen in our experiments as $\Delta_{\text{max}} = 8$ and $\Delta_{\text{avg}} = 2$. Note that more perceptual metrics will be considered afterwards (section 3.1).

The algorithm to construct the keypoint set $\mathcal{K}$ then consists of the following steps:

a. Initialization:

$\mathcal{K}$ 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. $\mathcal{K} = \{(\mathbf{x}_k, \mathbf{F}(\mathbf{x}_k)) \mid k = 1 \ldots 8\}$, for all $\mathbf{x}_k$ whose coordinates in $x$, $y$ and $z$ are either 0 or 255.

b. Adding keypoints:

As long as $E_{\text{max}} > \Delta_{\text{max}}$ or $E_{\text{avg}} > \Delta_{\text{avg}}$, new keypoint $\mathbf{x}_{N+1} = (\mathbf{x}_{N+1}, \tilde{\mathbf{F}}_N(\mathbf{x}_{N+1}))$ is added to the set $\mathcal{K}$, located at coordinates $\mathbf{x}_{N+1} = \arg\max_{\mathbf{x}} E_N(\mathbf{x})$ 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 $\Delta_{\text{max}}$ and $\Delta_{\text{avg}}$ (Fig. 8b). Fig. 8 illustrates the iterative construction of the set of keypoints $\mathcal{K}$ for one example CLUT. The reconstruction error is displayed below each $k$-point reconstruction, with the location of its maximal error, which is added as a new keypoint at the next iteration. As the iterations progress, the reconstruction error is visibly reduced. In the cases the CLUT contains discontinuities or inflection points, we observe that newly inserted keypoints are naturally located on either sides of these discontinuities, or on the inflection points, to ensure a correct reconstruction of the discontinuous CLUT (Fig. 9).

2.2.2 Removal of keypoints

Often, the addition of the last keypoint at step b leads to a CLUT reconstructed with an actual higher quality than expected, i.e. with $E_{\text{max}} < \Delta_{\text{max}} - \epsilon$ and/or $E_{\text{avg}} < \Delta_{\text{avg}} - \epsilon$ where $\epsilon > 0$ is not negligible. There is usually a subset of $\mathcal{K}$ that also verifies both reconstruction quality criteria, with an $\epsilon$ 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, $k = 1 \ldots N$), and checking whether the deletion of the $k^{\text{th}}$ keypoint
(a) Iterative construction of the keypoint set $K$ to compress a $CLUT \mathbf{F}$. **Top:** Reconstructed $CLUT$ with all keypoints, **Bottom:** Reconstruction error and location of the max-error point (i.e. the next keypoint to be inserted in $K$).

(b) Evolution of the error of the reconstructed $CLUT \mathbf{F}_N$ with respect to the target $CLUT \mathbf{F}$. **Left:** Evolution of the maximum error $E_{\text{max}}$ (in green) and average error $E_{\text{avg}}$ (in red), **Right:** Evolution of the PSNR.

**Figure 8:** Overview of the first 100 iterations of our proposed algorithm to compress a $3D \ CLUT$.

$K_k$ allows to reconstruct a $CLUT \mathbf{F}_N$ with quality constraints that still hold. If this is the case, the keypoint $K_k$ is discarded from $\mathcal{K}$ and the deletion process is resumed from where we left it. According to the degree of smoothness 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). It is interesting to note that in our experiments, none of the original 8 vertices has ever been deleted in this phase, for the hundreds of $CLUT$s we tried to compress.

At the end of these three steps, we get a set of keypoints $\mathcal{K}$ representing a compressed lossy version of a $CLUT \mathbf{F}$, such that a minimum quality of reconstruction is guaranteed.
3 Contributions: method analysis and improvements

In this section, a further analysis of this CLUT compression method is proposed: some possible alternatives are investigated and performances of the corresponding algorithms are compared, both in terms of visual quality and compression rate.

3.1 Considering other colorimetric spaces and color fidelity measures

Until now, and for simplicity reasons, we only considered the usual sRGB color space during the CLUT compression and decompression steps.

Let us now investigate other colorimetric spaces, both for CLUT reconstruction and sparse representation. In particular, the impact of different color representations on the performances of CLUT compression is experimentally tested, in order to get an answer to the following questions:

1. **Is there a preferred colorimetric space for the reconstruction of CLUTs by our anisotropic diffusion PDE (1)?**

Since a PDE can be seen as a local averaging process, three colorimetric spaces where the averaging of close colors remain an acceptable operation are compared: sRGB, Linear-RGB and L*a*b* [14].

2. **What error criterion \( E_{N(X)} \) should be preferred to visually minimize the effects of compression, from a perceptual point of view?**

In addition to the above \( L_2 \) error measure (denoted by \( L_2\)-sRGB hereafter), two additional criteria are considered

\[
\forall X \in RGB, \quad E_{N(X)} = \Delta E_{(76 \text{ or } 00)}(F(X), \tilde{F}_{N}(X))
\]
defined for each voxel of the RGB cube to be reconstructed, and based on measurements of the $\Delta E_{76}$ and $\Delta E_{00}$ perceptual color differences [16], computed from the original CLUTs $F$ and their reconstructed versions $\tilde{F}$.

The experiment consists in compressing a set of different CLUTs with varied contents, with a prescribed number of keypoints (here, 500 keypoints). One can thus compare the performance of the compression algorithm, for different spaces and different error measures. Table 10 shows the results of these comparisons, first for two single CLUTs (Faded 47 and Sprocket 231), then for a set of 31 separate CLUTs (displaying the median values for that set in the table).

Several observations can be drawn:

• For a given CLUT, it sometimes happens that the algorithm does not generate a compressed CLUT where the average of the error criterion used for the compression is minimal compared to other measured criteria (for instance, CLUT Faded 47 compressed with the $L_2$-sRGB metric does not have the best average measure for the $L_2$-sRGB criterion).

• However, such cases are infrequent, and the chosen error criterion is generally the one that is minimized, as clearly shown by the median values obtained on a sample of 31 different CLUTs.

• The sRGB color space used for the reconstruction often produces the best compression rates.

• The choice of a perceptual color difference as the error criterion does not require more points than $L_2$-sRGB, for reasonable values of compression constraints ($\Delta_{\text{max}} = 1.25$ and $\Delta_{\text{avg}} = 0.75$).

To conclude with these experiments, we found it optimal to keep the sRGB color space for reconstruction, and $\Delta E_{00}$ as the error criterion, which allows to minimize the perceptual color difference between the original CLUT $F$ and its compressed version $\tilde{F}$. This is what is done in the following.

3.2 Comparison with other related compression methods

3.2.1 Resampling

Since a CLUT is generally a rather smooth volumetric function, one might be tempted to regularly downsample its original definition domain (of size $N^3$) to get a smaller CLUT (of size $P^3$, typically $P = \frac{N}{2}$ or $\frac{N}{4}$), with the hope that reinterpolating the latter to the original size ($N^3$) will build a CLUT that is visually equivalent, e.g. leading to direct compression rates $\% Rate$ of 87.5% (for a $\times 2^3$ downsampling) or 98.4% (for a $\times 4^3$ downsampling). The compression rate is here calculated as:

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

Indeed, such an idea both looks trivial to implement and promising. Unfortunately, although it may work in a few very special cases (i.e. very smooth CLUTs without inflection points), this technique does not hold for general CLUT data that often exhibit a few local discontinuities (such as the one showcased in Fig. 9). To demonstrate that, the following experiment has been conducted for 6 different CLUTs taken from the [19] set: from each CLUT of size $N^3$ (with $N$ varying from 17 to 128), a smaller one of size $P^3$ is generated with 3D moving average, $P$ being 17 or 33 (typical sizes used in CLUT packs for storing small CLUTs). Then this small
<table>
<thead>
<tr>
<th>CLUT name</th>
<th>Space used for reconstruction</th>
<th>Error measure minimized for compression</th>
<th>PSNR (in dB)</th>
<th>L₂-sRGB (avg/max)</th>
<th>ΔE₇₆ (avg/max)</th>
<th>ΔE₀₀ (avg/max)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Faded</strong></td>
<td><strong>sRGB</strong></td>
<td>L₂-sRGB</td>
<td><strong>45.66</strong></td>
<td>2.08 / 5.34</td>
<td>1.03 / 3.36</td>
<td>0.59 / 3.02</td>
</tr>
<tr>
<td></td>
<td>sRGB</td>
<td>ΔE₇₆</td>
<td>45.56</td>
<td>2.01 / 7.64</td>
<td>0.93 / <strong>2.79</strong></td>
<td>0.54 / 2.48</td>
</tr>
<tr>
<td></td>
<td>sRGB</td>
<td>ΔE₀₀</td>
<td>45.42</td>
<td><strong>2.01</strong> / 8.5</td>
<td><strong>0.88</strong> / 3.9</td>
<td><strong>0.48</strong> / 1.33</td>
</tr>
<tr>
<td>Linear RGB</td>
<td>L₂-sRGB</td>
<td>ΔE₇₆</td>
<td>43.22</td>
<td>2.9 / 6.58</td>
<td>1.12 / 4.78</td>
<td>0.7 / 3.2</td>
</tr>
<tr>
<td>Linear RGB</td>
<td>L₂-sRGB</td>
<td>ΔE₀₀</td>
<td>40.52</td>
<td>3.83 / 12.52</td>
<td>1.3 / 3.52</td>
<td>0.86 / 3.11</td>
</tr>
<tr>
<td>Linear RGB</td>
<td>L₂-sRGB</td>
<td>ΔE₀₀</td>
<td>42.63</td>
<td>3 / 9.92</td>
<td>1.11 / 4.99</td>
<td>0.66 / 1.35</td>
</tr>
<tr>
<td>L<em>a</em>b*</td>
<td>L₂-sRGB</td>
<td>ΔE₇₆</td>
<td>44.29</td>
<td>2.5 / 5.06</td>
<td>1.04 / 3.72</td>
<td>0.62 / 2.65</td>
</tr>
<tr>
<td>L<em>a</em>b*</td>
<td>L₂-sRGB</td>
<td>ΔE₀₀</td>
<td>42.75</td>
<td>2.86 / 9.41</td>
<td>1.07 / 3.07</td>
<td>0.68 / 2.67</td>
</tr>
<tr>
<td>L<em>a</em>b*</td>
<td>L₂-sRGB</td>
<td>ΔE₀₀</td>
<td>43.58</td>
<td>2.61 / 9.97</td>
<td>1.02 / 4.98</td>
<td>0.58 / 1.35</td>
</tr>
<tr>
<td><strong>Sprocket</strong></td>
<td><strong>231 sRGB</strong></td>
<td>L₂-sRGB</td>
<td><strong>46.15</strong></td>
<td><strong>1.97</strong> / 11.09</td>
<td><strong>0.8</strong> / 3.64</td>
<td><strong>0.39</strong> / <strong>0.8</strong></td>
</tr>
<tr>
<td></td>
<td>sRGB</td>
<td>ΔE₇₆</td>
<td>42.58</td>
<td>3.15 / 5.63</td>
<td>1.23 / 4.6</td>
<td>0.67 / 2.28</td>
</tr>
<tr>
<td></td>
<td>sRGB</td>
<td>ΔE₀₀</td>
<td>41.94</td>
<td>3.23 / 11.86</td>
<td>0.99 / 2.04</td>
<td>0.62 / 1.87</td>
</tr>
<tr>
<td></td>
<td>sRGB</td>
<td>ΔE₀₀</td>
<td>42.35</td>
<td>3.13 / 12.85</td>
<td>1.13 / 4.23</td>
<td>0.59 / 1.14</td>
</tr>
<tr>
<td>L<em>a</em>b*</td>
<td>L₂-sRGB</td>
<td>ΔE₇₆</td>
<td>44.66</td>
<td>2.48 / 4.37</td>
<td>1.06 / 3</td>
<td>0.56 / 1.86</td>
</tr>
<tr>
<td>L<em>a</em>b*</td>
<td>L₂-sRGB</td>
<td>ΔE₀₀</td>
<td>43.25</td>
<td>2.72 / 11.59</td>
<td>0.96 / <strong>1.75</strong></td>
<td>0.54 / 2.26</td>
</tr>
<tr>
<td>L<em>a</em>b*</td>
<td>L₂-sRGB</td>
<td>ΔE₀₀</td>
<td>44.34</td>
<td>2.44 / 10.44</td>
<td>0.96 / 3.45</td>
<td>0.47 / 0.88</td>
</tr>
<tr>
<td><strong>Set of 31 CLUTs (median values)</strong></td>
<td>sRGB</td>
<td>L₂-sRGB</td>
<td><strong>43.63</strong></td>
<td><strong>2.64</strong> / 6.54</td>
<td><strong>1.16</strong> / 4.79</td>
<td><strong>0.61</strong> / 3.93</td>
</tr>
<tr>
<td></td>
<td>sRGB</td>
<td>ΔE₇₆</td>
<td>42.09</td>
<td>2.94 / 16.82</td>
<td><strong>1.07</strong> / <strong>2.76</strong></td>
<td>0.68 / 2.94</td>
</tr>
<tr>
<td></td>
<td>sRGB</td>
<td>ΔE₀₀</td>
<td>40.4</td>
<td>3.28 / 23.78</td>
<td>1.24 / 6.39</td>
<td><strong>0.64</strong> / <strong>1.73</strong></td>
</tr>
<tr>
<td>Linear RGB</td>
<td>L₂-sRGB</td>
<td>ΔE₇₆</td>
<td>39.21</td>
<td>4.56 / 9.21</td>
<td>1.71 / 7.02</td>
<td>0.97 / 5.85</td>
</tr>
<tr>
<td>Linear RGB</td>
<td>L₂-sRGB</td>
<td>ΔE₀₀</td>
<td>37.76</td>
<td>5.12 / 21.73</td>
<td>1.52 / 3.85</td>
<td>0.89 / 3.8</td>
</tr>
<tr>
<td>Linear RGB</td>
<td>L₂-sRGB</td>
<td>ΔE₀₀</td>
<td>36.09</td>
<td>5.72 / 29.56</td>
<td>1.75 / 7.67</td>
<td>0.88 / 2.26</td>
</tr>
<tr>
<td>L<em>a</em>b*</td>
<td>L₂-sRGB</td>
<td>ΔE₇₆</td>
<td>40.12</td>
<td>4.05 / 8.8</td>
<td>1.66 / 6.67</td>
<td>0.92 / 5.45</td>
</tr>
<tr>
<td>L<em>a</em>b*</td>
<td>L₂-sRGB</td>
<td>ΔE₀₀</td>
<td>39.1</td>
<td>4.3 / 21.92</td>
<td>1.41 / 3.39</td>
<td>0.74 / 3.57</td>
</tr>
<tr>
<td>L<em>a</em>b*</td>
<td>L₂-sRGB</td>
<td>ΔE₀₀</td>
<td>37.53</td>
<td>4.8 / 29.43</td>
<td>1.66 / 7.88</td>
<td>0.76 / 1.83</td>
</tr>
</tbody>
</table>

Figure 10: Comparing performances of CLUT compression/reconstruction for different colorspace and error criteria (best scores in bold).

P³ CLUT is re-interpolated to its original size N³, by tricubic interpolation. Finally, the differences between the re-interpolated and the original CLUTs are computed using three different metrics (PSNR, ΔE₀₀(avg) and ΔE₀₀(max)).

Fig. 11 summarizes these differences: are highlighted in red the cases where the difference measurements fall below the acceptable quality threshold defined for our own compression method, described in section 3.1.
(i.e. $\Delta_{\text{max}} = 1.25$ and $\Delta_{\text{avg}} = 0.75$). While the average $\Delta E_{00(\text{avg})}$ effectively remains below a visually indistinguishable threshold ($\Delta E_{00(\text{avg})} < 1$), this is far from being the case for the maximum $\Delta E_{00(\text{max})}$, which grows larger and larger as the downsampling factor increases: local discontinuities in CLUTs are obviously destroyed by downsampling, and re-interpolation locally generates very different colors from the original CLUT, leading to visually discernible differences between the two color functions.

Therefore, CLUT compression by downsampling is not a viable idea unless the CLUTs under consideration show absolutely no local discontinuities, which almost never actually happens.

<table>
<thead>
<tr>
<th>CLUT Name</th>
<th>Hypressen</th>
<th>Expired69</th>
<th>Amstragram</th>
<th>Prussian Blue</th>
<th>Street</th>
<th>Cinematic Mexico</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resolution</td>
<td>$17^3$</td>
<td>$33^3$</td>
<td>$48^3$</td>
<td>$48^3$</td>
<td>$64^3$</td>
<td>$128^3$</td>
</tr>
<tr>
<td>Downsampling factor to $17^3$</td>
<td>$\times 1$</td>
<td>$\times 1.9^3$</td>
<td>$\times 2.8^3$</td>
<td>$\times 2.8^3$</td>
<td>$\times 3.8^3$</td>
<td>$\times 7.5^4$</td>
</tr>
<tr>
<td>PSNR</td>
<td>$\infty$</td>
<td>42.2</td>
<td>42.2</td>
<td>39</td>
<td>40</td>
<td>39.7</td>
</tr>
<tr>
<td>$\Delta E_{00(\text{avg})}$</td>
<td>0</td>
<td>0.67</td>
<td>0.71</td>
<td>0.86</td>
<td>0.81</td>
<td>0.95</td>
</tr>
<tr>
<td>$\Delta E_{00(\text{max})}$</td>
<td>0</td>
<td>2.48</td>
<td>2.64</td>
<td>3.7</td>
<td>3.1</td>
<td>4.05</td>
</tr>
<tr>
<td>% Rate/uncomp.</td>
<td>0%</td>
<td>86.3%</td>
<td>95.6%</td>
<td>95.6%</td>
<td>98.1%</td>
<td>99.8%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CLUT Name</th>
<th>Hypressen</th>
<th>Expired69</th>
<th>Amstragram</th>
<th>Prussian Blue</th>
<th>Street</th>
<th>Cinematic Mexico</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resolution</td>
<td>-</td>
<td>$\times 1$</td>
<td>$\times 1.5^3$</td>
<td>$\times 1.5^3$</td>
<td>$\times 1.9^3$</td>
<td>$\times 3.9^4$</td>
</tr>
<tr>
<td>Downsampling factor to $33^4$</td>
<td>-</td>
<td>$\infty$</td>
<td>50.8</td>
<td>49</td>
<td>48.1</td>
<td>46.2</td>
</tr>
<tr>
<td>PSNR</td>
<td>-</td>
<td>$\infty$</td>
<td>0.34</td>
<td>0.32</td>
<td>0.37</td>
<td>0.51</td>
</tr>
<tr>
<td>$\Delta E_{00(\text{avg})}$</td>
<td>-</td>
<td>0</td>
<td>2.49</td>
<td>2.67</td>
<td>2.17</td>
<td>3.23</td>
</tr>
<tr>
<td>$\Delta E_{00(\text{max})}$</td>
<td>-</td>
<td>0%</td>
<td>67.5%</td>
<td>67.5%</td>
<td>86.3%</td>
<td>98.3%</td>
</tr>
</tbody>
</table>

3.2.2 Harmonic functions

Here are given some arguments for considering an anisotropic diffusion equation such as (1) to reconstruct CLUTs, rather than an extension by harmonic functions, that in fact corresponds to the solution of the following isotropic diffusion PDEs:

$$\forall X \in RGB, \quad \frac{\partial F}{\partial t}(X) = m(X) \Delta F(X)$$

Fig. 12 shows an example of reconstruction of a function $F$ from a set of color keypoints $K$, defined here on a 2D rather than 3D domain (for better illustration purpose). The 3D elevation is computed for each point as the norm of the RGB vector. The isotropic/anisotropic nature of the various reconstructions we get is pretty clear. The resulting colorimetric variations of the anisotropic scheme (3) appear to be closer to the variations that are observed in CLUTs in practice, thus justifying the use of the anisotropic model, that has the same algorithmic complexity as the isotropic one (Fig. 12b,d).

3.2.3 Radial basis functions

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 [1, 28]. Some other 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, would estimate each color component $F^i$ of $F$ ($i = R, G$ or $B$) by:

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

(5)

with $\phi : \mathbb{R}^+ \rightarrow \mathbb{R}$, a given RBF function (e.g. $\phi(r) = r^2 \ln r$, for a thin plate spline interpolation [9]). Then, one get weights $w^i_k$ by solving a linear system, involving the known values of the keypoints $C_k$ and a matrix whose coefficients are $\phi(||X_p - X_q||)$, 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 (Fig. 12c).

As far as we know, CLUT design using RBFs-based interpolation has been first proposed by the Darktable open-source project in 2016, within a specific image modification module of the software allowing to build user-personalized CLUTs from a set of 292-max color keypoints [6]. It might therefore seem appropriate to replace our CLUT reconstruction step from a set of keypoints, as presented in section 2.1, by a RBF-based reconstruction. Unfortunately, the algorithmic complexity of RBFs is expressed as $O(N^3 + N r^3)$ for the reconstruction of a CLUT of resolution $r^3$ from $N$ keypoints: $O(N^3)$ for the nonsparse matrix inversion and $O(N r^3)$ for estimating the interpolated values in 3D. Although each of these two steps can be implemented by parallel calculations, the computing time becomes prohibitive when the number of keypoints increases notably (e.g. $N > 300$, which happens very frequently in our application of CLUT compression, see Fig. 15).

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 multiscale 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 multicore computer (for several tens of seconds with a comparable RBF approach), and this, with an equally good reconstruction quality.

Fig. 13 illustrates the fact that for some CLUTs, the number of keypoints required to accurately represent the data is indeed smaller when using the RBF approach (first column). In practice, one should notice that the RBF reconstruction works remarkably well to compress CLUTs that do not exhibit complex local geometric structures, i.e. those with a 3D gradient norm that remains small enough over the whole RGB cube. On the other hand, when more keypoints are needed to capture all the small color discontinuities in CLUTs whose gradient norm is high everywhere, our PDE approach shows better performance, both in terms of the number of keypoints and the computation time to get these points (particularly when considering the removal step of our compression algorithm). Looking at the sharper structures reconstructed by our diffusion PDEs in Fig. 12d may be a part of the explanation.

3.2.4 Keypoints thinning

Our simple method to determine the set of color keypoints that represents a CLUT, described in section 2.2.1, is a straightforward greedy bottom-up approach: it tries to minimize the number of keypoints necessary to achieve a high enough quality for the reconstruction of the CLUT, as well as the number of full CLUT reconstructions required to compare them with the original CLUT data to be compressed (this 3D reconstruction step being indeed the one with the highest algorithmic complexity). Note that our method is totally deterministic and does not involve any random choice: for the same input CLUT, the set of keypoints obtained by our compression
Figure 12: Comparisons between the proposed anisotropic PDE (1), the harmonic functions (4) and the Radial Basis Functions approach (5), for the reconstruction of a dense function from a set of sparse color keypoints (here in the 2D domain, for illustration purpose). Displayed 3D elevation $E$ is computed at each point as $E = \sqrt{R^2 + G^2 + B^2}$. 
<table>
<thead>
<tr>
<th><strong>CLUT</strong> name</th>
<th><strong>Tension Green-1</strong> ((17^3))</th>
<th><strong>Cinematic-1</strong> ((17^3))</th>
</tr>
</thead>
<tbody>
<tr>
<td>3D view</td>
<td><img src="image1" alt="3D view of Tension Green-1 CLUT" /></td>
<td><img src="image2" alt="3D view of Cinematic-1 CLUT" /></td>
</tr>
<tr>
<td>3D view of the gradient norm</td>
<td><img src="image3" alt="3D view of Tension Green-1 CLUT" /></td>
<td><img src="image4" alt="3D view of Cinematic-1 CLUT" /></td>
</tr>
<tr>
<td>Keypoint construction with <strong>RBFs</strong></td>
<td>177 keypoints (obtained in 14s)</td>
<td>936 keypoints (obtained in 30m 11s)</td>
</tr>
<tr>
<td>Keypoint construction with <strong>PDEs</strong></td>
<td>446 keypoints (obtained in 57s)</td>
<td>823 keypoints (obtained in 1m 42s)</td>
</tr>
</tbody>
</table>

Figure 13: Experimenting **PDEs** and **RBFs**-based approaches for determining keypoints, on two **CLUTs** with very different local structures.

The search for a minimal subset of keypoints for the reconstruction of image data is a problem that has already been addressed in the literature, e.g. for the purpose of compression or inpainting of **2D** natural images, as in [7, 13, 15, 17]. In these papers, the keypoint determination methods are stochastic and can therefore generate different sets of points at each run, from the same input image. Several strategies are studied: In [7], the authors propose a top-down thinning approach which consists in initializing the set of keypoints with all points of the image to be compressed, a set which is iteratively reduced by removing optimal elements among a set of randomly chosen keypoints. In [17], a genetic algorithm is proposed. Starting from a set of a few hundred keypoints, this set is iteratively and slightly modified according to a few different rules (point displacement, addition, removal, etc.) and only improved sets are kept over time, until convergence towards a stable state. In [15], a bottom-up approach based on a dithering method of the error map is proposed to add new keypoints, followed by a pixel exchange step. In [13], a similar idea is used, with stochastic sparsification of image laplacian halftoning.
The main difference of all these methods compared to ours is the need to evaluate the data reconstructed from the keypoints more often. Indeed, the stochastic aspect of these methods imposes an iterative selection among several candidate keypoints at each iteration (and thus several reconstructions of the image data at each iteration), whereas our method requires only one reconstruction step per iteration (the number of iterations corresponding exactly to the number of selected keypoints).

Here our keypoint determination method is compared to the approach of [7], as it is one of the most different approaches from ours (top-down approach, while ours is bottom-up). The experiments are limited to $17^3$ CLUTs since it already represents an initializing set of $17^3 = 4913$ keypoints, which is far above the desired number of keypoints. Compared to the original algorithm described in [7], the following adjustments have been made to make the comparison with our approach relevant:

- The algorithm has been extended to CLUT datasets defined in 3D, rather than 2D images.
- We do not compute the Delaunay tetrahedrization of the 3D point set, as doing it in 3D requires complex computations. At each iteration, the selection of the keypoint candidates to be suppressed in the same neighborhood is simplified by randomly selecting about ten candidates that are close enough to each other, according to the Euclidean distance.
- The 3D CLUT reconstruction step thus does not use a linear reconstruction of the keypoints based on the 3D Delaunay tetrahedrization anymore, but uses our PDE-based method, as described in section 2.1.
- The algorithm is stopped when the reconstructed CLUT no longer satisfies the quality constraints, given by $\Delta_{\text{max}} = 1.25$ and $\Delta_{\text{avg}} = 0.75$.

Fig. 14 shows the compression results for a set of 6 different CLUTs already used in section 3.2.1 (but downsized to $17^3$). To take into account the stochastic aspect of the thinning approach, we have kept the minimal set of keypoints after launching the algorithm three times in a row (the displayed timings correspond to a single execution of the algorithm though). The number of keypoints obtained with our keypoint determination approach is also displayed for comparison purposes.

<table>
<thead>
<tr>
<th>CLUT name</th>
<th>Hypressen (17³)</th>
<th>Expired69 (17³)</th>
<th>Amstragram (17³)</th>
<th>Prussian Blue (17³)</th>
<th>Street (17³)</th>
<th>Cinematic Mexico (17³)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thinning approach</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td># Keypoints</td>
<td>732</td>
<td>596</td>
<td>359</td>
<td>816</td>
<td>704</td>
<td>521</td>
</tr>
<tr>
<td>Compression time</td>
<td>80m 26s</td>
<td>118m 37s</td>
<td>82m 50s</td>
<td>127m 15s</td>
<td>104m 13s</td>
<td>112m 50s</td>
</tr>
<tr>
<td>Keypoints in .png</td>
<td>3.35 Kb</td>
<td>2.48 Kb</td>
<td>1.79 Kb</td>
<td>3.39 Kb</td>
<td>3.13 Kb</td>
<td>3.6 Kb</td>
</tr>
<tr>
<td>Our approach</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td># Keypoints</td>
<td>417</td>
<td>373</td>
<td>271</td>
<td>553</td>
<td>338</td>
<td>352</td>
</tr>
<tr>
<td>Compression time</td>
<td>52s</td>
<td>45s</td>
<td>33s</td>
<td>50s</td>
<td>42s</td>
<td>44s</td>
</tr>
<tr>
<td>Keypoints in .png</td>
<td>2 Kb</td>
<td>1.59 Kb</td>
<td>1.37 Kb</td>
<td>2.35 Kb</td>
<td>1.64 Kb</td>
<td>1.78 Kb</td>
</tr>
<tr>
<td>%/Rate/thinning</td>
<td>40.3%</td>
<td>35.9%</td>
<td>23.5%</td>
<td>30.7%</td>
<td>47.6%</td>
<td>50.6%</td>
</tr>
</tbody>
</table>

Figure 14: Experimenting keypoint thinning algorithm [7] for 3D CLUT compression, and comparison with our proposed method.
One can see that the thinning algorithm works satisfactorily and returns a number of keypoints with an order comparable to what we get with our method (a few hundred). Our method always generates slightly smaller sets of keypoints. However, the paramount thing is the compression time, which is really not of the same order of magnitude (a few tens of seconds for our method, versus approximately a hundred minutes for the thinning approach). This timing difference would be obviously even more visible for larger CLUTs.

Actually, limiting as much as possible the number of CLUT reconstructions needed to converge to an acceptable set of keypoints is in our case much more important than finding the smallest set of keypoints, because the CLUT data to be processed naturally compresses very well. It is therefore irrelevant to significantly increase the algorithmic complexity for the compression algorithm. We also think that the very smooth general appearance of the CLUTs to be compressed, with only a few important inflection points to be taken into account, makes a bottom-up approach better suited than a top-down approach (which is not necessarily the case for natural images).

Our algorithm has not been compared to other bottom-up approaches to determining key points (e.g [13, 15]). Indeed, the stochastic aspect of these methods means that the number of 3D reconstructions will generally be greater. Keeping an algorithm that minimizes the number of reconstructions is of the utmost importance in order to make CLUT compression usable in practice.

### 3.3 Final compression results

The performance of our compression method has been evaluated on publicly available datasets (including [19, 21, 26]) for a total of 552 CLUTs at various resolutions (ranging from $17^3$ to $144^3$), encoding very diverse color transformations. The 35 Free LUTs [26] is a collection of CLUTs for video color grading, that are available as .cube files. The Hald CLUTs [21] is a collection of 517 CLUTs for film simulation profiles, that are available in the RGB color space, 8-bit per channel, in .png format. The PIXLS.US CLUTS [19] is a similar collection of 31 CLUTs, specifically proposed by the PIXLS.US community, dedicated to the use of free software for photographic retouching.

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 552 sets of keypoints indicates that the average number of keypoints is 888 (minimum: 10, maximum: 2047, standard deviation: 714), which is high enough to make our fast PDE-based reconstruction technique more suitable than RBFs for CLUT decompression.

The table in Fig. 15 provides individual compression measurements for a sample of 6 CLUTs taken from [19]. 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. uncompressed raw files or compressed files in .png, .cube.zip and .icc.zip formats. Computational time have been obtained on a standard 24-core PC. 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 smoothness (the keypoints being naturally located on the less differentiable areas of the CLUTs (Fig. 9). By limiting the reconstruction error average and maximal values of $\Delta E_{00}$, we ensure a perceptually indiscernible difference between the decompressed CLUT $\tilde{F}$ and the original one $F$. For the purpose of scientific replicability, our CLUT compression/decompression algorithms have been integrated into the G'MIC software [29], a full-featured open-source framework for image processing, used by thousands of people around the world.
<table>
<thead>
<tr>
<th>CLUT name</th>
<th>Hypressen</th>
<th>Expired69</th>
<th>Amstragram</th>
<th>Prussian Blue</th>
<th>Street</th>
<th>Cinematic Mexico</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resolution</td>
<td>17³</td>
<td>33³</td>
<td>48³</td>
<td>48³</td>
<td>64³</td>
<td>128³</td>
</tr>
<tr>
<td>Size uncomp. (8bits)</td>
<td>14.4 Kb</td>
<td>105 Kb</td>
<td>324 Kb</td>
<td>324 Kb</td>
<td>768 Kb</td>
<td>6 Mb</td>
</tr>
<tr>
<td>Size uncomp. (float)</td>
<td>57.6 Kb</td>
<td>421 Kb</td>
<td>1.3 Mb</td>
<td>1.3 Mb</td>
<td>3 Mb</td>
<td>24 Mb</td>
</tr>
<tr>
<td>Size of .cube.zip</td>
<td>130 Kb</td>
<td>712 Kb</td>
<td>604 Kb</td>
<td>598 Kb</td>
<td>1.3 Mb</td>
<td>7.3 Mb</td>
</tr>
<tr>
<td>Size of .png</td>
<td>8.5 Kb</td>
<td>30.4 Kb</td>
<td>83.4 Kb</td>
<td>78.7 Kb</td>
<td>151 Kb</td>
<td>947 Kb</td>
</tr>
<tr>
<td>Size of .icc.zip N³</td>
<td>49.7 Kb</td>
<td>359 Kb</td>
<td>1.1 Mb</td>
<td>1.1 Mb</td>
<td>2.5 Mb</td>
<td>19 Mb</td>
</tr>
<tr>
<td>Size of .icc.zip 17³</td>
<td>49.7 Kb</td>
<td>49.9 Kb</td>
<td>49.6 Kb</td>
<td>49.9 Kb</td>
<td>49.8 Kb</td>
<td>50 Kb</td>
</tr>
<tr>
<td>Size of .icc.zip 33³</td>
<td>358 Kb</td>
<td>356 Kb</td>
<td>360 Kb</td>
<td>358 Kb</td>
<td>357 Kb</td>
<td></td>
</tr>
<tr>
<td># of keypoints</td>
<td>417</td>
<td>300</td>
<td>300</td>
<td>506</td>
<td>393</td>
<td>651</td>
</tr>
<tr>
<td>PSNR</td>
<td>40.9 dB</td>
<td>42.3 dB</td>
<td>44.1 dB</td>
<td>42.2 dB</td>
<td>42.3 dB</td>
<td>43.3 dB</td>
</tr>
<tr>
<td>∆E²⁰⁰⁰(avg)</td>
<td>0.6</td>
<td>0.51</td>
<td>0.52</td>
<td>0.53</td>
<td>0.52</td>
<td>0.5</td>
</tr>
<tr>
<td>∆E²⁰⁰⁰(max)</td>
<td>1.25</td>
<td>1.25</td>
<td>1.25</td>
<td>1.25</td>
<td>1.25</td>
<td>1.25</td>
</tr>
<tr>
<td>Compression time</td>
<td>52.3s</td>
<td>2m 33s</td>
<td>15m 14s</td>
<td>9m 19s</td>
<td>15m 31s</td>
<td>210m 8s</td>
</tr>
<tr>
<td>Decompression time</td>
<td>37ms</td>
<td>137ms</td>
<td>227ms</td>
<td>247ms</td>
<td>462ms</td>
<td>5725ms</td>
</tr>
<tr>
<td>Keypoints in .png</td>
<td>2 Kb</td>
<td>1.5 Kb</td>
<td>1.7 Kb</td>
<td>2.6 Kb</td>
<td>2.2 Kb</td>
<td>3.6 Kb</td>
</tr>
<tr>
<td>%Rate/uncomp.8b</td>
<td>86.3%</td>
<td>98.5%</td>
<td>99.5%</td>
<td>99.2%</td>
<td>99.7%</td>
<td>99.9%</td>
</tr>
<tr>
<td>%Rate/.cube.zip</td>
<td>98.5%</td>
<td>99.8%</td>
<td>99.7%</td>
<td>99.6%</td>
<td>99.8%</td>
<td>≈ 100%</td>
</tr>
<tr>
<td>%Rate/.png</td>
<td>77%</td>
<td>95%</td>
<td>98%</td>
<td>96.8%</td>
<td>98.6%</td>
<td>99.6%</td>
</tr>
<tr>
<td>%Rate/.icc.zip N³</td>
<td>96%</td>
<td>99.6%</td>
<td>99.8%</td>
<td>99.8%</td>
<td>99.9%</td>
<td>≈ 100%</td>
</tr>
<tr>
<td>%Rate/.icc.zip 17³</td>
<td>96%</td>
<td>96.9%</td>
<td>96.6%</td>
<td>94.9%</td>
<td>95.7%</td>
<td>92.8%</td>
</tr>
<tr>
<td>%Rate/.icc.zip 33³</td>
<td>99.4%</td>
<td>99.6%</td>
<td>99.5%</td>
<td>99.3%</td>
<td>99.4%</td>
<td>99%</td>
</tr>
</tbody>
</table>

Figure 15: Results of our CLUT compression algorithm, on different CLUTs from [26] (with ∆max = 1.25 and ∆avg = 0.75).

4 Exemplar-based CLUT generation

We now describe a second application of our fast reconstruction method of CLUTs, which solves the following problem: an artist gives a particular colorimetric mood to one of his creations I₀ (photography or illustration). Using his custom color manipulation tools, he manages to generate a modified version of it, I_modified that he is satisfied with and that he archives. Later on, the same artist decides to give the same colorimetric mood to another of his images, I₁, but without exactly remembering all the steps taken to transform I₀ into I_modified and the precise setting of all parameters he had to set in his color retouching tools. Consequently he cannot reapply his color transformation identically to I₁.

A variation of the problem can be stated as follows: how can an artist ensure that the colorimetric mood he has just created can easily be reproduced by users of other image manipulation software (which therefore have probably different color retouching tools, or at least different parameter settings)?

Let us remind this is a different issue from color transfer between images: the purpose here is not to guess how to transfer colors from one image to another (this transformation has already been done by the artist), but to be able to extract a full-filled CLUT that reflects this transformation, and then apply it possibly to a whole series of images. Indeed, the transformation performed beforehand by the artist provides a pixel-to-pixel correspondence between the original and the transformed images. Of course, having this pair of example images
eases the construction of the CLUT (hence the term Exemplar-based to name this process).

Since a generic color transformation can be modeled by a CLUT, this problem can be solved by determining the 3D dense function $F : RGB \rightarrow RGB$ that verifies the following two constraints:

a. **Data attachment constraint:**

The example transformation $I_{0}^{\text{original}} \rightarrow I_{0}^{\text{modified}}$ must be preserved, which implies that the application of CLUT $F$ to image $I_{0}^{\text{original}}$ must result exactly in $I_{0}^{\text{modified}}$. It means that the values of $F$ are already known for all points of RGB which are actual colors of $I_{0}^{\text{original}}$:

$$\forall p \in \Omega, \quad F(I_{0(p)}^{\text{original}}) = I_{0(p)}^{\text{modified}} \quad (6)$$

b. **Density and regularity constraint:**

If one wants to apply $F$ to another image $I_{1}^{\text{original}}$, the values $F(X)$ must be defined for at least all colors $X$ existing in $I_{1}^{\text{original}}$ (so, ideally for all possible colors of RGB). Since CLUTs are most often piecewise continuous functions, colorimetric discontinuities in unknown parts of $F$ must be avoided. Therefore the known colors of $I_{0}^{\text{modified}}$ have to be continuously interpolated throughout the whole RGB space.

Both constraints can naturally be taken into account by generating $F$ with our anisotropic diffusion method (1) described in section 2.1, applied to the set $\mathcal{K}$ of the following keypoints:

$$\mathcal{K} = \left\{(I_{0(p)}^{\text{original}}, I_{0(p)}^{\text{modified}}) \mid p \in \Omega \right\}$$

where $\Omega$ is the spatial discretized definition domain of images $I_{0}^{\text{original}}$ and $I_{0}^{\text{modified}}$. Here, the number of keypoints $N = \text{card}(\Omega)$ is potentially quite high, which makes our fast multiscale PDE-based reconstruction method all the more valuable, compared to a RBF approach.

Hence, the interpolated CLUT $F$ exactly reproduces transformation $I_{0}^{\text{original}} \rightarrow I_{0}^{\text{modified}}$, and its application to $I_{1}^{\text{original}}$ generates consistent colors with respect to the colors of $I_{0}^{\text{modified}}$, so as to get a similar colorimetric mood, as shown in images $(c, d, h)$ of Fig. 16.

c. **Attachment term to colors missing in the original image:**

However, when image $I_{1}^{\text{original}}$ has tones that are not close to the colors of $I_{0}^{\text{original}}$ (for instance, the blue sky in image of Fig. 16g, last row) one may wish to control the degree of transformation of such colors in the CLUT under construction.

To this end, an additional parameter $\alpha$ is introduced, for attachment to missing colors to $I_{1}^{\text{original}}$, acting on these specific colors whose transformation is a priori not well defined. We then propose to estimate the global color transformation by a distance-blended CLUT, denoted by $F_{\alpha}$, and calculated as follows:

$$\forall X \in RGB, \quad F_{\alpha}(X) = \beta_{\alpha}^2(X) F(X) + (1 - \beta_{\alpha}^2) X$$

where

$$\beta_{\alpha}^2(X) = \exp \left( -\frac{d_K^2(X)}{2\alpha^2} \right) \quad \text{and} \quad d_K(X) = \min_{k \in 0...N} \|X - X_k\|$$
The term \( \beta^\alpha(X) \) allows to locally weight the influence of colors \( F(X) \) of the interpolated CLUT, with respect to the distance of color \( X \) to all the existing colors of image \( I_0 \) (Fig. 16, (e), (f), (i)). Taking into account \( \beta^\alpha(X) \) has a visible effect only when image \( I_0 \) has no similar colors present in \( I_1 \) (here, the blue sky). Otherwise, the application of interpolated and distance-blended CLUTs gives visually equivalent results (Fig. 16, (h), (i)).

This technique for generating CLUTs enables any purely colorimetric transformation to be extracted from a single pair of before/after images. It can then be easily reapplied later, or distributed in a standardized form, so that anyone can reproduce it on other images or software. Finally, it should be noted that it is still possible to compress \( F_\alpha \) afterwards, by the CLUT compression algorithm described in section 2.2.

5 Conclusion

The CLUT compression/decompression technique presented in this paper is surprisingly effective. This is mainly due to the adequacy of the proposed 3D diffusion model (1) to the type of data processed (smooth, volumetric, color-valued).

As a result, all the 552 CLUTs compressed by our method and integrated into G’MIC [29], could be integrated into any image editing software to offer photographers and illustrators the greatest diversity of color transformations, and this, for a minimal storage cost (2.5 Mb). In addition, our reconstruction method applied to exemplar-based CLUT generation can greatly help artists to build and distribute personalized CLUTs.

We are convinced that the integration of these algorithms into other image or video processing software would allow the distribution of CLUT-based color transformations at a much larger magnitude scale than current standards. At the present time, there is no image retouching software offering several hundreds of artistic nonparametric color transformations that can be applied on images or videos. In essence, our method allows to automatically re-parameter any kind of color transformations with a quite minimal set of parameters modeled as keypoints.

References


Figure 16: Exemplar-based CLUT generation: principles and comparison between interpolated and distance-blended CLUTs.