



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:

David Tschumperlé, Christine Porquet, Amal Mahboubi. Reconstruction of Smooth 3D Color Functions from Keypoints: Application to Lossy Compression and Exemplar-Based Generation of Color LUTs. SIAM Journal on Imaging Sciences, 2020, 13 (3), pp.1511-1535. 10.1137/19M1306798 . hal-02940557

HAL Id: hal-02940557

<https://normandie-univ.hal.science/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,
David.Tschumperle@ensicaen.fr

Christine Porquet,
Christine.Porquet@ensicaen.fr

Amal Mahboubi
Amal.Mahboubi@unicaen.fr

Normandie Univ, UNICAEN, ENSICAEN, CNRS, GREYC, F-14000 Caen, France
19th November 2020

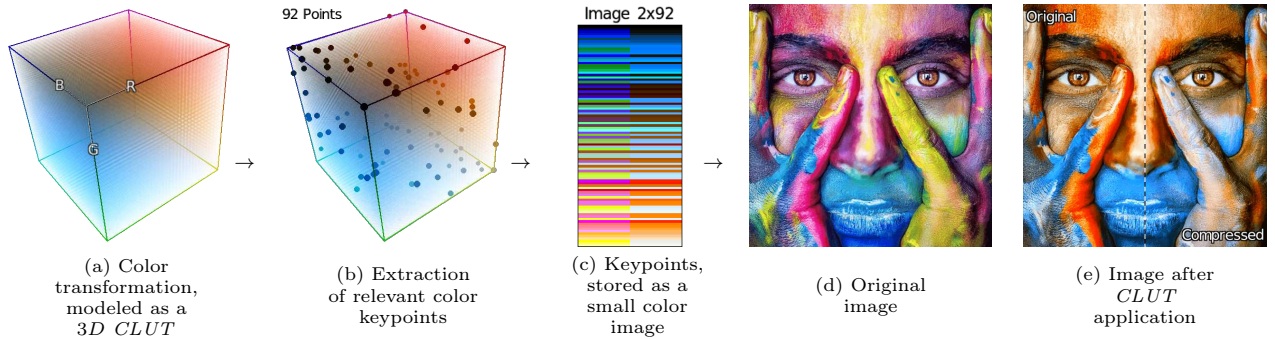


Figure 1: Our approach is able to generate a compressed representation of a generic, nonparametric, smooth $RGB \rightarrow RGB$ mapping, modeled as a 3D CLUT (a). We extract a set of relevant keypoints (b,c) that sparsely describes the CLUT structure, and provides a related CLUT decomposition algorithm 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

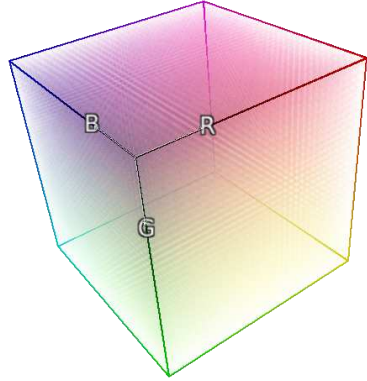
3D CLUTs (Color Look Up Tables) are popular digital models used in artistic image and video processing, for color grading, simulation of analog films, and more generally for the description and application of generic nonparametric color transformations. The relatively large size of these models leads 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).



(a) A *CLUT* is a 3D function $RGB \rightarrow RGB$, here visualized with semitransparent colored spheres for each point of RGB



(b) Original image



(c) Image after applying *CLUT*

Figure 2: 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 $\mathbf{I} : \Omega \rightarrow RGB$ (defined on a rectangular domain $\Omega \subset \mathbb{R}^2$) is done as follows:

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

where I_R , I_G and I_B are the RGB color components of \mathbf{I} . It should be noted that 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 $\mathbf{F}(\mathbf{x})$ 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 $\mathbf{F}(\mathbf{x})$ of RGB , unrolled as a 2D image (Fig. 4b). In both cases, the high number of color voxels composing the RGB cube leads

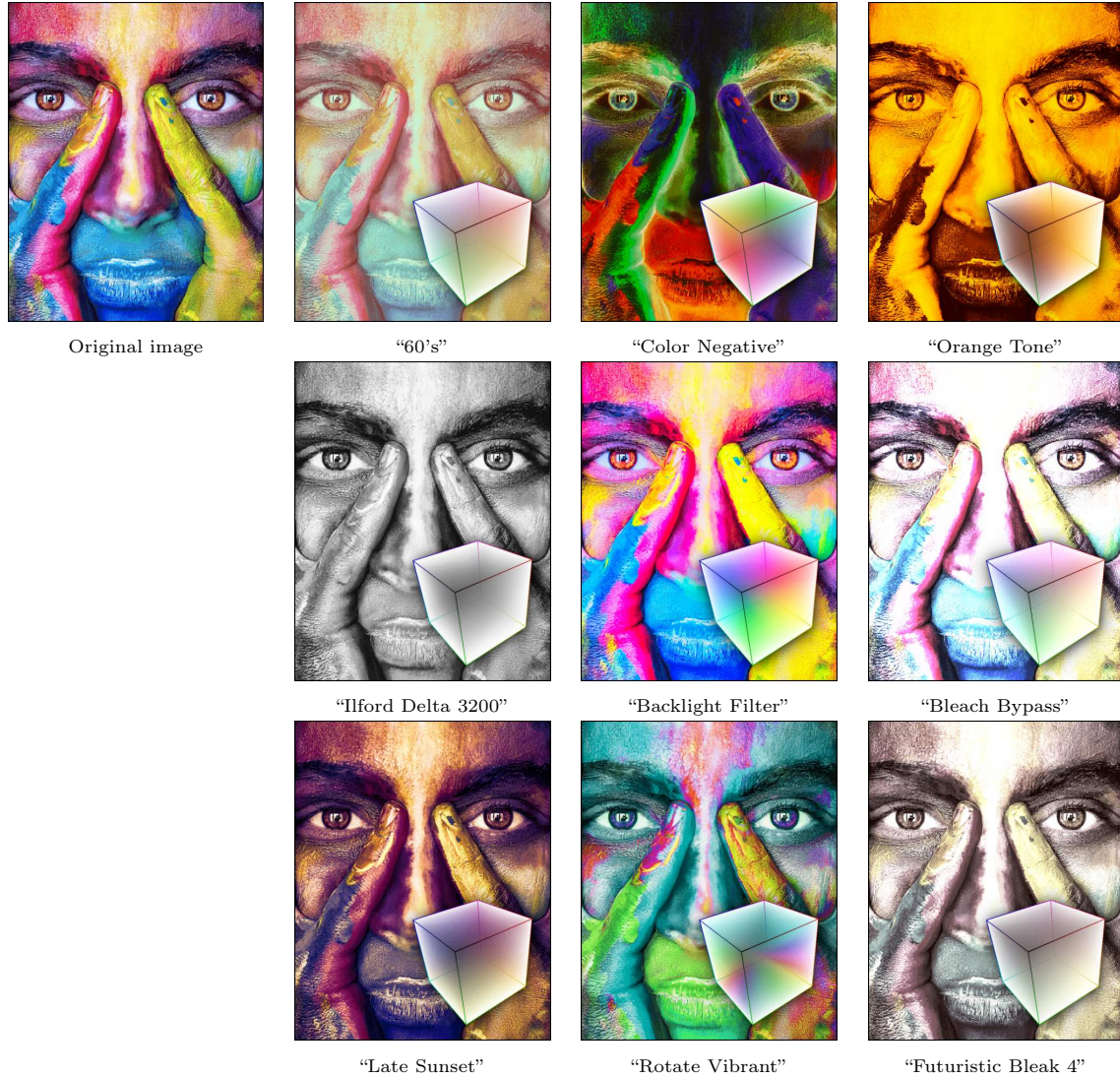


Figure 3: Generic nature of color transformations allowed by the use of 3D *CLUTs*.

to a quite large storage size for a single *CLUT*: For instance, a 256^3 -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 downsampled to a regularly subsampled *RGB* space (typically with resolutions $17^3, 33^3, 48^3, 64^3, \dots$) 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 a uncompressed array of raw data (32bits-float for each *RGB* component of the *CLUT* voxels).

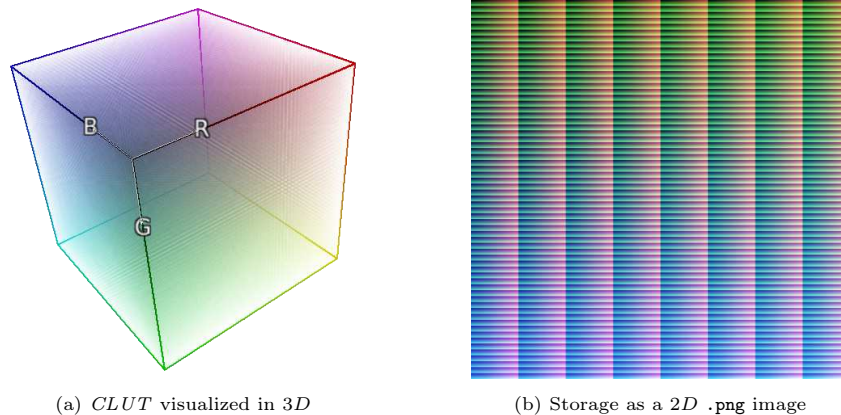


Figure 4: Storage of a downscaled *CLUT* as a *.png* file: The 64^3 colors of the *CLUT* are unrolled as a 512^2 image. Despite the apparent continuity of the 3D function, the resulting image exhibits lots of discontinuities, which make its harder to compress in 2D.

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 ($\approx 30\%$ 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 ($\geq 95\%$), 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* \mathbf{F} as input and generates a much smaller representation \mathbf{F}_c such that \mathbf{F} can be reconstructed from \mathbf{F}_c at its full resolution. The compression scheme is said to be *lossy* [24], as the reconstructed *CLUT* $\tilde{\mathbf{F}}$ slightly differs from \mathbf{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 \mathcal{K} of colored keypoints $\mathbf{K}_k \in RGB \times RGB$, defined as ordered pairs:

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

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 *RGB* color.

We assume that the set $\mathcal{K} = \{\mathbf{K}_1, \mathbf{K}_2, \dots, \mathbf{K}_N\}$ already provides a sparse representation of a given *CLUT* \mathbf{F} .

2.1.1 Diffusion 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. Using diffusion is indeed a natural way to ensure smoothness of the reconstructed *CLUT*. 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.*

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

We propose to reconstruct \mathbf{F} by solving the following anisotropic diffusion *PDE*:

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

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

At each point, the diffusion strength $m(\mathbf{x})$ 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 $\mathbf{F}_{t=0}$ as close as possible to a solution of (1). Actually, one can get a quite good estimate for $\mathbf{F}_{t=0}$ 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 [4]), then by smoothing it by an isotropic *3D* gaussian filter (Fig. 5b). A more efficient multiscale scheme for estimating $\mathbf{F}_{t=0}$ 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 \mathbf{x} of the *RGB* cube to its nearest keypoint. This filtering is done for all points \mathbf{x} of *RGB*, except for the known keypoints \mathbf{x}_k which keep their initial color \mathbf{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 \mathcal{K} composed of 6 colored keypoints.

2.1.2 Spatial discretization

Numerically, $d_{\mathcal{K}}$ is efficiently computed (in linear time) by a distance transform, such as the one proposed in [18]. 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 proposed:

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

where

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

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. 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.

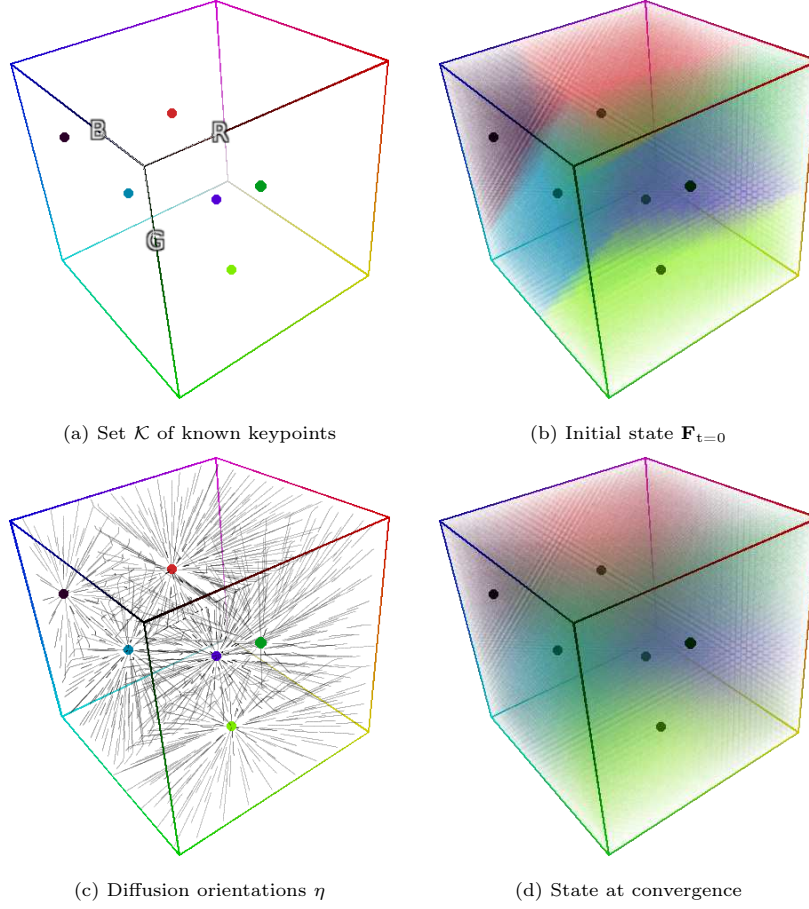


Figure 5: Reconstruction of a 3D *CLUT* \mathbf{F} from a set of keypoints \mathcal{K} using anisotropic diffusion PDE (1) (here, from 6 keypoints).

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:

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

which leads to:

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

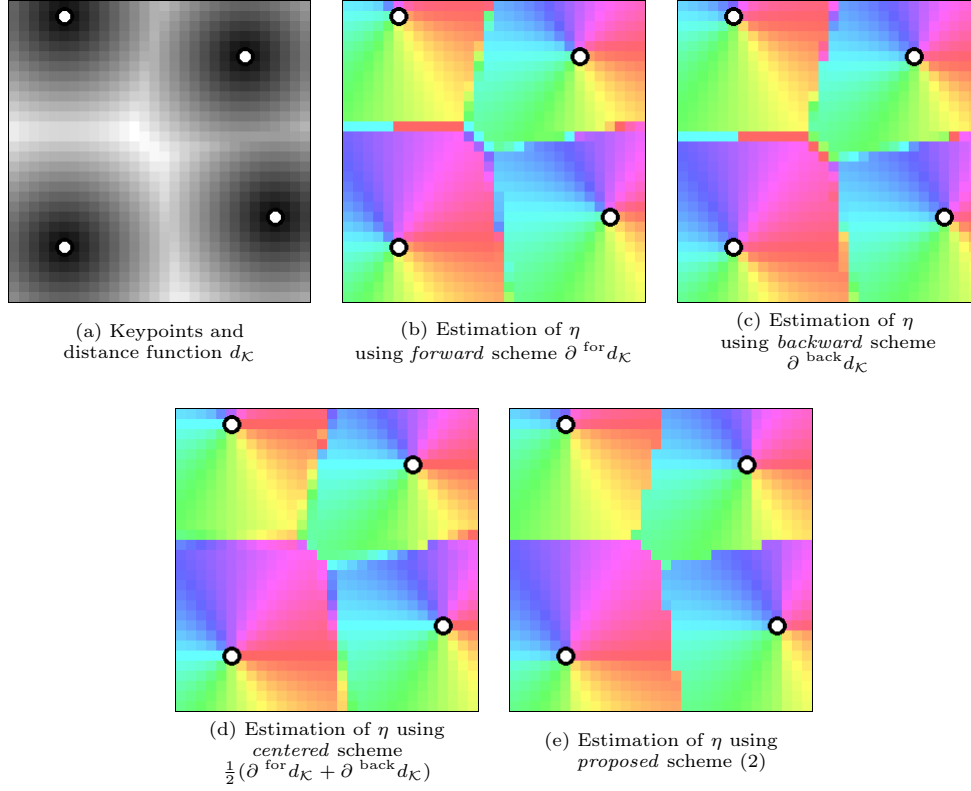


Figure 6: Influence of our numerical schemes for estimating the diffusion orientations η (shown here on a small 40×40 crop of a 2D distance function d_K). Hues displayed at each point represent the estimated orientations η .

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:

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

where $\mathbf{F}_{(\mathbf{x}+\eta)}^t$ and $\mathbf{F}_{(\mathbf{x}-\eta)}^t$ are accurately estimated using tricubic spatial interpolation. Starting from $\mathbf{F}_{t=0}$, 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 \mathbf{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* \mathbf{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 $\mathbf{F}_{t=0}$ by *watershed*-like propagation for computing the diffusion at resolution $(2^s)^3$, $\mathbf{F}_{t=0}$ 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 \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 $2^3, 4^3, 8^3 \dots$, until the desired resolution is reached (Fig. 7).

Note also that for each different resolution, the coordinates \mathbf{X}_k of the color keypoints are downscaled accordingly, and rounded to the nearest integers.

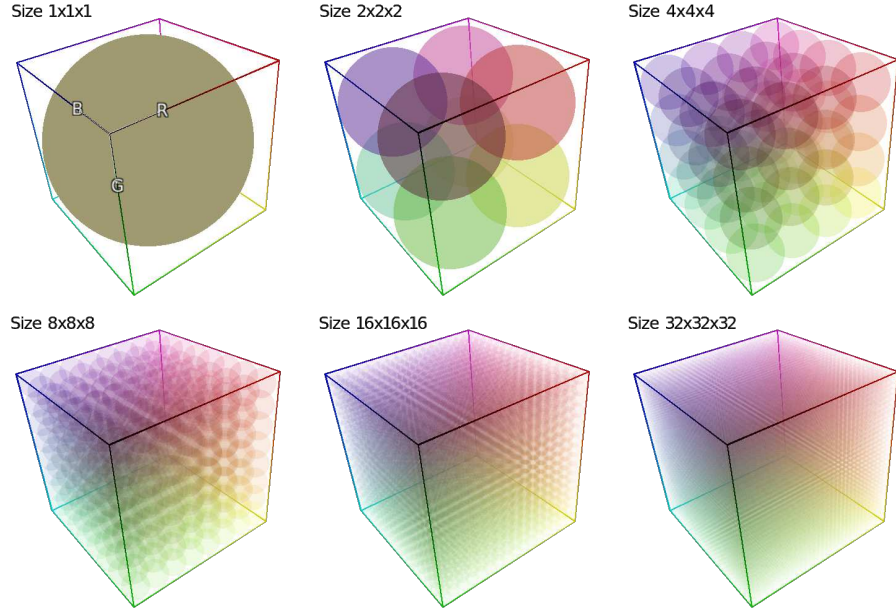


Figure 7: Multiscale reconstruction scheme: A reconstructed *CLUT* at resolution $(2^s)^3$ is linearly upscaled and used as an initialization for applying the diffusion scheme at a higher resolution $(2^{s+1})^3$.

2.2 A greedy compression algorithm

Now that the reconstruction of a dense *CLUT* \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} .

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 $E_N : RGB \rightarrow \mathbb{R}^+$ 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.

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

The set \mathcal{K} is populated until two reconstruction quality criteria are met, which are $E_{\max} \leq \Delta_{\max}$, 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

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

$\Delta_{\max} \in \mathbb{R}^+$ and $\Delta_{\text{avg}} \in \mathbb{R}^+$ are the two main parameters of the compression method, and are chosen in our experiments as $\Delta_{\max} = 8$ 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. $\mathcal{K} = \{(\mathbf{X}_k, \mathbf{F}(\mathbf{X}_k)) \mid k = 1 \dots 8\}$, for all \mathbf{X}_k whose coordinates in x , y and z are either 0 or 255.

b. Adding keypoints:

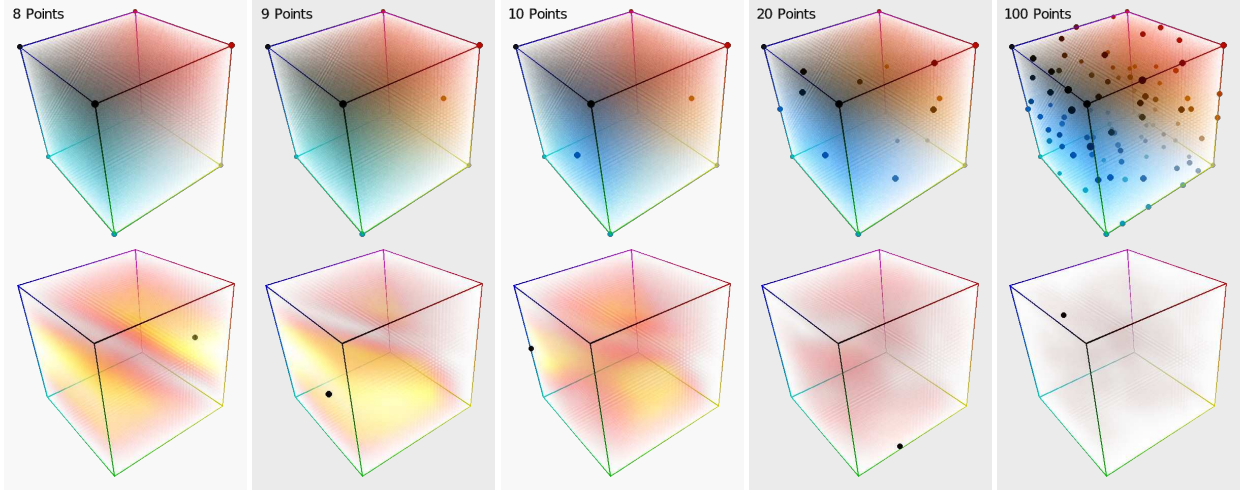
As long as $E_{\max} > \Delta_{\max}$ or $E_{\text{avg}} > \Delta_{\text{avg}}$, new keypoint $\mathbf{K}_{N+1} = (\mathbf{X}_{N+1}, \mathbf{F}_{N+1}(\mathbf{X}_{N+1}))$ is added to the set \mathcal{K} , located at coordinates $\mathbf{X}_{N+1} = \text{argmax}_{\mathbf{X}}(E_N)$ of the maximum reconstruction 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 Δ_{\max} and Δ_{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).

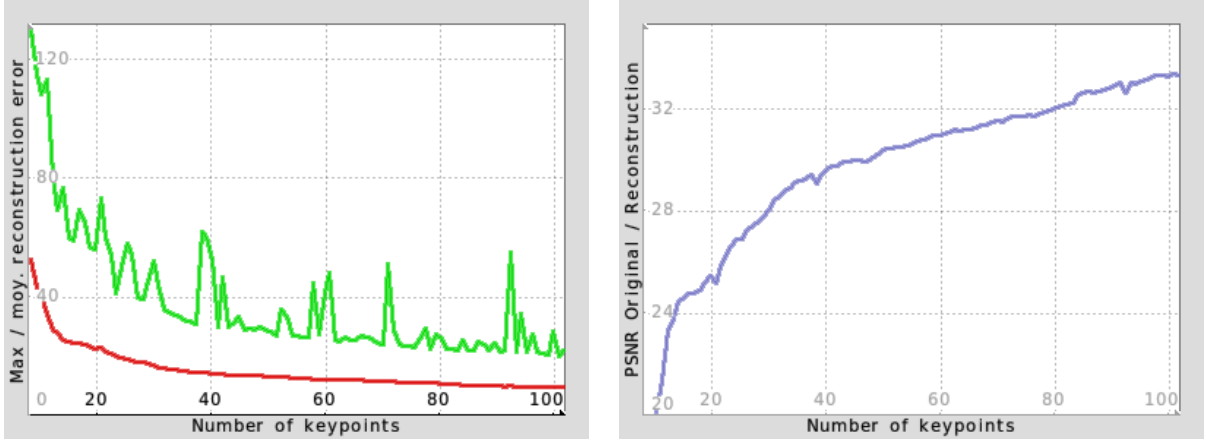
2.2.2 Removal of keypoints

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 $E_{\max} < \Delta_{\max} - \epsilon$ 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 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



(a) Iterative construction of the keypoint set \mathcal{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 \mathcal{K}).



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

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

\mathbf{K}_k allows to reconstruct a $CLUT \tilde{\mathbf{F}}_N$ with quality constraints that still hold. If this is the case, the keypoint \mathbf{K}_k is discarded from \mathcal{K} and the 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 $CLUTs$ 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.

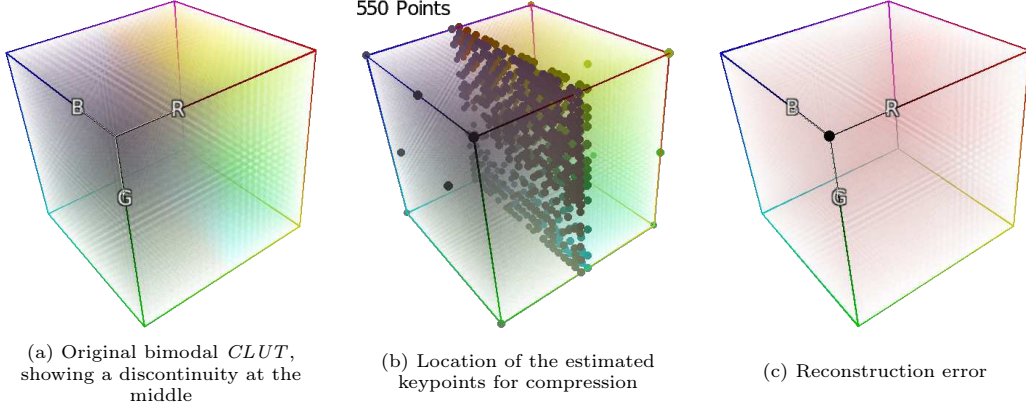


Figure 9: Dealing with discontinuous *CLUT*s. During compression, color keypoints are automatically inserted at either side of the discontinuity, to ensure a good *CLUT* reconstruction. No visible seam appears in the reconstruction error.

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 $E_{N(\mathbf{x})}$ 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

$$\forall \mathbf{X} \in RGB, \quad E_{N(\mathbf{x})} = \Delta E_{\{76 \text{ or } 00\}}(\mathbf{F}(\mathbf{x}), \tilde{\mathbf{F}}_{N(\mathbf{x})})$$

defined for each voxel of the *RGB* cube to be reconstructed, and based on measurements of the ΔE_{76} and ΔE_{00} perceptual color differences [16], computed from the original *CLUTs* \mathbf{F} and their reconstructed versions $\tilde{\mathbf{F}}$.

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 ($\Delta_{\max} = 1.25$ and $\Delta_{\text{avg}} = 0.75$).

To conclude with these experiments, we found it optimal to keep the *sRGB* color space for reconstruction, and ΔE_{00} as the error criterion, which allows to minimize the perceptual color difference between the original *CLUT* \mathbf{F} and its compressed version $\tilde{\mathbf{F}}$. 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

| CLUT name | Space used for reconstruction | Error measure minimized for compression | PSNR (in dB) | L_2 - $sRGB$ (avg/max) | ΔE_{76} (avg/max) | ΔE_{00} (avg/max) |
|---------------------------------|-------------------------------|---|--------------|---------------------------|---------------------------|---------------------------|
| Faded 47 | $sRGB$ | L_2 - $sRGB$ | 45.66 | 2.08 / 5.34 | 1.03 / 3.36 | 0.59 / 3.02 |
| | $sRGB$ | ΔE_{76} | 45.56 | 2.01 / 7.64 | 0.93 / 2.79 | 0.54 / 2.48 |
| | $sRGB$ | ΔE_{00} | 45.42 | 2.01 / 8.5 | 0.88 / 3.9 | 0.48 / 1.33 |
| | Linear RGB | L_2 - $sRGB$ | 43.22 | 2.9 / 6.58 | 1.12 / 4.78 | 0.7 / 3.2 |
| | Linear RGB | ΔE_{76} | 40.52 | 3.83 / 12.52 | 1.3 / 3.52 | 0.86 / 3.11 |
| | Linear RGB | ΔE_{00} | 42.63 | 3 / 9.92 | 1.11 / 4.99 | 0.66 / 1.35 |
| | $L^*a^*b^*$ | L_2 - $sRGB$ | 44.29 | 2.5 / 5.06 | 1.04 / 3.72 | 0.62 / 2.65 |
| | $L^*a^*b^*$ | ΔE_{76} | 42.75 | 2.86 / 9.41 | 1.07 / 3.07 | 0.68 / 2.67 |
| | $L^*a^*b^*$ | ΔE_{00} | 43.58 | 2.61 / 9.97 | 1.02 / 4.98 | 0.58 / 1.35 |
| Sprocket 231 | $sRGB$ | L_2 - $sRGB$ | 45.5 | 2.22 / 5.04 | 1.02 / 3.39 | 0.51 / 2.22 |
| | $sRGB$ | ΔE_{76} | 44.99 | 2.27 / 6.4 | 0.78 / 1.78 | 0.44 / 1.46 |
| | $sRGB$ | ΔE_{00} | 46.15 | 1.97 / 11.09 | 0.8 / 3.64 | 0.39 / 0.8 |
| | Linear RGB | L_2 - $sRGB$ | 42.58 | 3.15 / 5.63 | 1.23 / 4.6 | 0.67 / 2.28 |
| | Linear RGB | ΔE_{76} | 41.94 | 3.23 / 11.86 | 0.99 / 2.04 | 0.62 / 1.87 |
| | Linear RGB | ΔE_{00} | 42.35 | 3.13 / 12.85 | 1.13 / 4.23 | 0.59 / 1.14 |
| | $L^*a^*b^*$ | L_2 - $sRGB$ | 44.66 | 2.48 / 4.37 | 1.06 / 3 | 0.56 / 1.86 |
| | $L^*a^*b^*$ | ΔE_{76} | 43.25 | 2.72 / 11.59 | 0.96 / 1.75 | 0.54 / 2.26 |
| | $L^*a^*b^*$ | ΔE_{00} | 44.34 | 2.44 / 10.44 | 0.96 / 3.45 | 0.47 / 0.88 |
| Set of 31 CLUTs (median values) | $sRGB$ | L_2 - $sRGB$ | 43.63 | 2.64 / 6.54 | 1.16 / 4.79 | 0.61 / 3.93 |
| | $sRGB$ | ΔE_{76} | 42.09 | 2.94 / 16.82 | 1.07 / 2.76 | 0.68 / 2.94 |
| | $sRGB$ | ΔE_{00} | 40.4 | 3.28 / 23.78 | 1.24 / 6.39 | 0.64 / 1.73 |
| | Linear RGB | L_2 - $sRGB$ | 39.21 | 4.56 / 9.21 | 1.71 / 7.02 | 0.97 / 5.85 |
| | Linear RGB | ΔE_{76} | 37.76 | 5.12 / 21.73 | 1.52 / 3.85 | 0.89 / 3.8 |
| | Linear RGB | ΔE_{00} | 36.09 | 5.72 / 29.56 | 1.75 / 7.67 | 0.88 / 2.26 |
| | $L^*a^*b^*$ | L_2 - $sRGB$ | 40.12 | 4.05 / 8.8 | 1.66 / 6.67 | 0.92 / 5.45 |
| | $L^*a^*b^*$ | ΔE_{76} | 39.1 | 4.3 / 21.92 | 1.41 / 3.39 | 0.74 / 3.57 |
| | $L^*a^*b^*$ | ΔE_{00} | 37.53 | 4.8 / 29.43 | 1.66 / 7.88 | 0.76 / 1.83 |

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

P^3 CLUT is re-interpolated to its original size N^3 , by tricubic interpolation. Finally, the differences between the re-interpolated and the original CLUTs are computed using three different metrics ($PSNR$, $\Delta E_{00(avg)}$ and $\Delta E_{00(max)}$).

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. $\Delta_{\max} = 1.25$ 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.

| <i>CLUT</i> name | Hypresen | Expired69 | Amstragram | Prussian Blue | Street | Cinematic Mexico |
|---------------------------------|-----------------|-----------------|-----------------|-----------------|-----------------|------------------|
| Resolution | 17 ³ | 33 ³ | 48 ³ | 48 ³ | 64 ³ | 128 ³ |
| Downsampling to 17 ³ | | | | | | |
| Downsampling factor | $\times 1$ | $\times 1.9^3$ | $\times 2.8^3$ | $\times 2.8^3$ | $\times 3.8^3$ | $\times 7.5^3$ |
| PSNR | ∞ | 42.2 | 42.2 | 39 | 40 | 39.7 |
| $\Delta E_{00(\text{avg})}$ | 0 | 0.67 | 0.71 | 0.86 | 0.81 | 0.95 |
| $\Delta E_{00(\text{max})}$ | 0 | 2.48 | 2.64 | 3.7 | 3.1 | 4.05 |
| %Rate/uncomp. | 0% | 86.3% | 95.6% | 95.6% | 98.1% | 99.8% |
| Downsampling to 33 ³ | | | | | | |
| Downsampling factor | - | $\times 1$ | $\times 1.5^3$ | $\times 1.5^3$ | $\times 1.9^3$ | $\times 3.9^3$ |
| PSNR | - | ∞ | 50.8 | 49 | 48.1 | 46.2 |
| $\Delta E_{00(\text{avg})}$ | - | 0 | 0.34 | 0.32 | 0.37 | 0.51 |
| $\Delta E_{00(\text{max})}$ | - | 0 | 2.49 | 2.67 | 2.17 | 3.23 |
| %Rate/uncomp. | - | 0% | 67.5% | 67.5% | 86.3% | 98.3% |

Figure 11: Performances of *CLUT* compression by resampling.

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*:

$$\forall \mathbf{X} \in RGB, \quad \frac{\partial \mathbf{F}}{\partial t}(\mathbf{X}) = m(\mathbf{x}) \Delta \mathbf{F}(\mathbf{x}) \quad (4)$$

Fig. 12 shows an example of reconstruction of a function \mathbf{F} from a set of color keypoints \mathcal{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 \mathbf{F}^i of \mathbf{F} ($i = R, G$ or B) by:

$$\forall \mathbf{X} \in RGB, \quad \mathbf{F}^i_{(\mathbf{X})} = \sum_{k=1}^N w_k^i \phi(\|\mathbf{X} - \mathbf{X}_k\|), \quad (5)$$

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_k^i 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 (Fig. 12c).

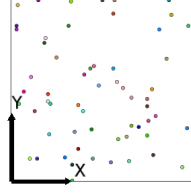
As far as we know, *CLUT* design using RBFs-based interpolation has been first proposed by the *Darktable* open-source project in 2016, within a specific image modification module of the software allowing to build user-personalized *CLUTs* from a set of 292-max color keypoints [6]. It might therefore 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.

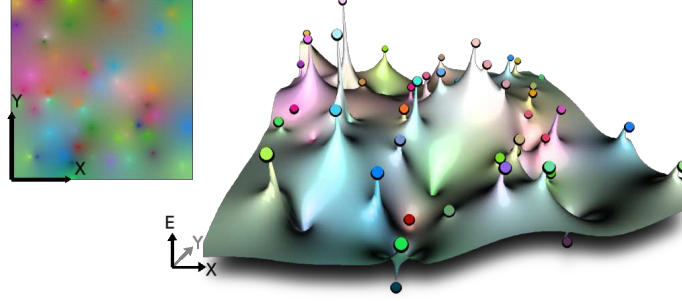
Fig. 13 illustrates the fact that for some *CLUTs*, the number of keypoints required to accurately represent the data is indeed smaller when using the *RBF* approach (first column). In practice, one should notice that the *RBF* reconstruction works remarkably well to compress *CLUTs* that do not exhibit 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

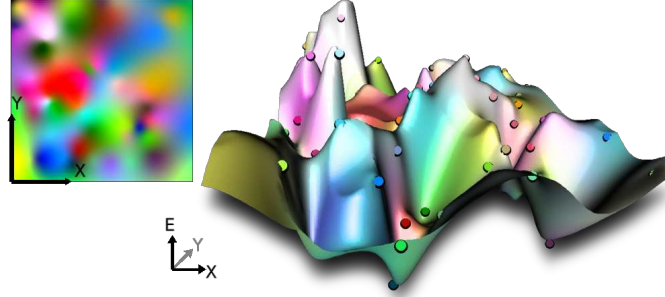
Our simple method to determine the set of color keypoints that represents a *CLUT*, described in section 2.2.1, is a straightforward greedy bottom-up approach: it tries to minimize the number of keypoints necessary to achieve a high enough quality for the reconstruction of the *CLUT*, *as well as* the 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



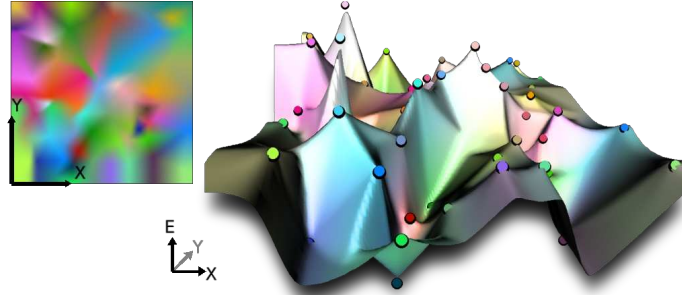
(a) Color keypoints \mathcal{K} in $2D$



(b) Reconstruction with Harmonic Functions (4), and its $3D$ elevation E



(c) Reconstruction with *RBFs* (5), and its $3D$ elevation E



(d) Reconstruction with our anisotropic PDE (1), and its $3D$ elevation E

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 $E = \sqrt{R^2 + G^2 + B^2}$.

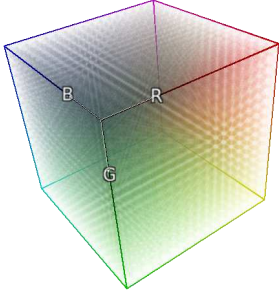
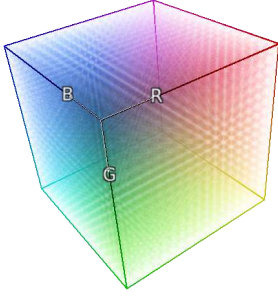
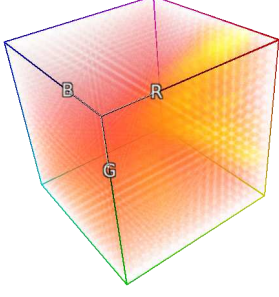
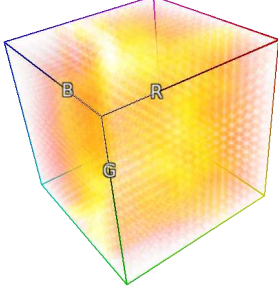
| <i>CLUT</i> name | Tension Green-1 (17^3) | Cinematic-1 (17^3) |
|---|---|--|
| 3D view |  |  |
| 3D view of the gradient norm |  |  |
| Keypoint construction with <i>RBFs</i> | 177 keypoints (obtained in 14s) | 936 keypoints (obtained in 30m 11s) |
| Keypoint construction with <i>PDEs</i> | 446 keypoints (obtained in 57s) | 823 keypoints (obtained in 1m 42s) |

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

algorithm will always be the same (for prescribed quality criteria).

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 $\Delta_{\max} = 1.25$ 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.

| <i>CLUT</i> name | Hypresen (17^3) | Expired69 (17^3) | Amstragram (17^3) | Prussian Blue (17^3) | Street (17^3) | Cinematic Mexico (17^3) |
|-------------------|------------------------|-------------------------|--------------------------|--------------------------------|----------------------|-----------------------------------|
| Thinning approach | | | | | | |
| # Keypoints | 732 | 596 | 359 | 816 | 704 | 521 |
| Compression time | 80m 26s | 118m 37s | 82m 50s | 127m 15s | 104m 13s | 112m 50s |
| Keypoints in .png | 3.35 Kb | 2.48 Kb | 1.79 Kb | 3.39 Kb | 3.13 Kb | 3.6 Kb |
| Our approach | | | | | | |
| # Keypoints | 417 | 373 | 271 | 553 | 338 | 352 |
| Compression time | 52s | 45s | 33s | 50s | 42s | 44s |
| Keypoints in .png | 2 Kb | 1.59 Kb | 1.37 Kb | 2.35 Kb | 1.64 Kb | 1.78 Kb |
| %Rate/thinning | 40.3% | 35.9% | 23.5% | 30.7% | 47.6% | 50.6% |

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 ΔE_{00} , we ensure a perceptually indiscernible difference between the decompressed *CLUT* $\tilde{\mathbf{F}}$ and the original one \mathbf{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.

| <i>CLUT</i> name | Hypresen | Expired69 | Amstragram | Prussian Blue | Street | Cinematic Mexico |
|----------------------------------|-----------------|-----------------|-----------------|-----------------|-----------------|------------------|
| Resolution | 17 ³ | 33 ³ | 48 ³ | 48 ³ | 64 ³ | 128 ³ |
| Size uncomp. (8bits) | 14.4 Kb | 105 Kb | 324 Kb | 324 Kb | 768 Kb | 6 Mb |
| Size uncomp. (float) | 57.6 Kb | 421 Kb | 1.3 Mb | 1.3 Mb | 3 Mb | 24 Mb |
| Size of .cube.zip | 130 Kb | 712 Kb | 604 Kb | 598 Kb | 1.3 Mb | 7.3 Mb |
| Size of .png | 8.5 Kb | 30.4 Kb | 83.4 Kb | 78.7 Kb | 151 Kb | 947 Kb |
| Size of .icc.zip N^3 | 49.7 Kb | 359 Kb | 1.1 Mb | 1.1 Mb | 2.5 Mb | 19 Mb |
| Size of .icc.zip 17 ³ | 49.7 Kb | 49.9 Kb | 49.6 Kb | 49.9 Kb | 49.8 Kb | 50 Kb |
| Size of .icc.zip 33 ³ | 358 Kb | 359 Kb | 356 Kb | 360 Kb | 358 Kb | 357 Kb |
| # of keypoints | 417 | 300 | 300 | 506 | 393 | 651 |
| PSNR | 40.9 dB | 42.3 dB | 44.1 dB | 42.2 dB | 42.3 dB | 43.3 dB |
| $\Delta E_{2000}(avg)$ | 0.6 | 0.51 | 0.52 | 0.53 | 0.52 | 0.5 |
| $\Delta E_{2000}(max)$ | 1.25 | 1.25 | 1.25 | 1.25 | 1.25 | 1.25 |
| Compression time | 52.3s | 2m 33s | 15m 14s | 9m 19s | 15m 31s | 210m 8s |
| Decompression time | 37ms | 137ms | 227ms | 247ms | 462ms | 5725ms |
| Keypoints in .png | 2 Kb | 1.5 Kb | 1.7 Kb | 2.6 Kb | 2.2 Kb | 3.6 Kb |
| %Rate/uncomp.8b | 86.3% | 98.5% | 99.5% | 99.2% | 99.7% | 99.9% |
| %Rate/.cube.zip | 98.5% | 99.8% | 99.7% | 99.6% | 99.8% | $\approx 100\%$ |
| %Rate/.png | 77% | 95% | 98% | 96.8% | 98.6% | 99.6% |
| %Rate/.icc.zip N^3 | 96% | 99.6% | 99.8% | 99.8% | 99.9% | $\approx 100\%$ |
| %Rate/.icc.zip 17 ³ | 96% | 96.9% | 96.6% | 94.9% | 95.7% | 92.8% |
| %Rate/.icc.zip 33 ³ | 99.4% | 99.6% | 99.5% | 99.3% | 99.4% | 99% |

Figure 15: Results of our *CLUT* compression algorithm, on different *CLUTs* from [26] (with $\Delta_{max} = 1.25$ and $\Delta_{avg} = 0.75$).

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 $\mathbf{I}_0^{\text{original}}$ (photography or illustration). Using his custom color manipulation tools, he manages to generate a modified version of it, $\mathbf{I}_0^{\text{modified}}$ 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, $\mathbf{I}_1^{\text{original}}$, but without exactly remembering all the steps taken to transform $\mathbf{I}_0^{\text{original}}$ into $\mathbf{I}_0^{\text{modified}}$ 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 $\mathbf{I}_1^{\text{original}}$.

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 $\mathbf{F} : RGB \rightarrow RGB$ that verifies the following two constraints:

a. Data attachment constraint:

The example transformation $\mathbf{I}_0^{\text{original}} \rightarrow \mathbf{I}_0^{\text{modified}}$ must be preserved, which implies that the application of *CLUT* \mathbf{F} to image $\mathbf{I}_0^{\text{original}}$ must result exactly in $\mathbf{I}_0^{\text{modified}}$. It means that the values of \mathbf{F} are already known for all points of *RGB* which are actual colors of $\mathbf{I}_0^{\text{original}}$:

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

b. Density and regularity constraint:

If one wants to apply \mathbf{F} to another image $\mathbf{I}_1^{\text{original}}$, the values $\mathbf{F}(\mathbf{x})$ must be defined for at least all colors \mathbf{x} existing in $\mathbf{I}_1^{\text{original}}$ (so, ideally for all possible colors of *RGB*). Since *CLUTs* are most often piecewise continuous functions, colorimetric discontinuities in unknown parts of \mathbf{F} must be avoided. Therefore the known colors of $\mathbf{I}_0^{\text{modified}}$ have to be continuously interpolated throughout the whole *RGB* space.

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

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

where Ω is the spatial discretized definition domain of images $\mathbf{I}_{0(\mathbf{p})}^{\text{original}}$ and $\mathbf{I}_{0(\mathbf{p})}^{\text{modified}}$. 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* \mathbf{F} exactly reproduces transformation $\mathbf{I}_0^{\text{original}} \rightarrow \mathbf{I}_0^{\text{modified}}$, and its application to $\mathbf{I}_1^{\text{original}}$ generates consistent colors with respect to the colors of $\mathbf{I}_0^{\text{modified}}$, 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 $\mathbf{I}_1^{\text{original}}$ has tones that are not close to the colors of $\mathbf{I}_0^{\text{original}}$ (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 α is introduced, for *attachment to missing colors* to $\mathbf{I}_1^{\text{original}}$, 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 \mathbf{F}_α , and calculated as follows :

$$\forall \mathbf{x} \in RGB, \quad \mathbf{F}_\alpha(\mathbf{x}) = \beta_{(\mathbf{x})}^\alpha \mathbf{F}(\mathbf{x}) + (1 - \beta_{(\mathbf{x})}^\alpha) \mathbf{x}$$

where

$$\beta_{(\mathbf{x})}^\alpha = \exp \left(-\frac{d_{\mathcal{K}}^2(\mathbf{x})}{2\alpha^2} \right) \quad \text{and} \quad d_{\mathcal{K}}(\mathbf{x}) = \min_{k \in 0 \dots N} \|\mathbf{x} - \mathbf{x}_k\|$$

The term $\beta_{(\mathbf{x})}^\alpha$ allows to locally weight the influence of colors $\mathbf{F}_{(\mathbf{x})}$ of the interpolated *CLUT*, with respect to the distance of color \mathbf{X} to all the existing colors of image $\mathbf{I}_0^{\text{original}}$ (Fig. 16, (e), (f), (i)). Taking into account $\beta_{(\mathbf{x})}^\alpha$ has a visible effect only when image $\mathbf{I}_1^{\text{original}}$ has no similar colors present in $\mathbf{I}_0^{\text{original}}$ (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 \mathbf{F}_α 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

- [1] Ken Anjyo, John P Lewis, and Frédéric Pighin. Scattered data interpolation for computer graphics. In *ACM SIGGRAPH 2014 Courses*, page 27, New York, NY, USA, 2014. ACM, ACM.
- [2] Aravindh Balaji, Gaurav Sharma, Mark Shaw, and Randall Guay. Preprocessing Methods for Improved Lossless Compression of Color Look-Up Tables. *Jour. of Imag. Science and Tech.*, 07 2008.
- [3] Aravindh Balaji, Gaurav Sharma, Mark Q Shaw, and Randall Guay. Hierarchical compression of color look up tables. In *Color and Imaging Conference*, number 1, pages 261–266. Society for Imaging Science and Technology, 2007.
- [4] Serge Beucher and Fernand Meyer. The Morphological Approach to Segmentation: The Watershed Transformation. *Optical Engineering-N.Y.*, 34:433–433, 1992.
- [5] CIE. *ISO/CIE11664-6:2013(E): Joint ISO/CIE Standard: Colorimetry –Part 6: CIEDE2000 Colour-Difference Formula*. pub-ISO, 2013.

- [6] Darktable. Colour manipulation with the colour checker lut module. <https://www.darktable.org/2016/05/colour-manipulation-with-the-colour-checker-lut-module/>, 2016.
- [7] Laurent Demaret, Nira Dyn, and Armin Iske. Image compression by linear splines over adaptive triangulations. *Signal Processing*, 86(7):1604 – 1616, 2006.
- [8] 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.
- [9] Jean Duchon. Splines minimizing rotation-invariant semi-norms in sobolev spaces. In *Constructive theory of functions of several variables*, pages 85–100. Springer, 1977.
- [10] Steenberg Eskil. Explanation of what a 3D CLUT is (accessed 2019-12-13). <http://www.quelsolaar.com/technology/clut.html>, 2019.
- [11] Oriel Frigo, Neus Sabater, Vincent Demoulin, and Pierre Hellier. Optimal transportation for example-guided color transfer. In *Asian Conf. on Comp. Vision*, pages 655–670. Springer, 2014.
- [12] 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.
- [13] Laurent Hoeltgen, Markus Mainberger, Sebastian Hoffmann, Joachim Weickert, Ching Hoo Tang, Simon Setzer, Daniel Johannsen, Frank Neumann, and Benjamin Doerr. Optimising spatial and tonal data for pde-based inpainting, 2015.
- [14] International Color Consortium. Specification ICC. 1: 2004-10. *Image technology colour management—Architecture, profile format, and data structure*, 2004.
- [15] Lena Karos, Pinak Bheed, Pascal Peter, and Joachim Weickert. Optimising data for exemplar-based inpainting. In Jacques Blanc-Talon, David Helbert, Wilfried Philips, Dan Popescu, and Paul Scheunders, editors, *Advanced Concepts for Intelligent Vision Systems*, pages 547–558, Cham, 2018. Springer International Publishing.
- [16] M Ronnier Luo, Guihua Cui, and Bryan Rigg. The development of the CIE 2000 colour-difference formula: CIEDE2000. *Color Research & Application*, 26(5):340–350, 2001.
- [17] David Marwood, Pascal Massimino, Michele Covell, and Shumeet Baluja. Representing images in 200 bytes: Compression via triangulation, 2018.
- [18] 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, Boston, MA, 2002.
- [19] PIXLS.US. PIXLS.US Color LUTs. <https://discuss.pixls.us/t/help-to-create-a-set-of-pixls-us-color-luts/>, 2019.
- [20] Julien Rabin, Sira Ferradans, and Nicolas Papadakis. Adaptive Color Transfer With Relaxed Optimal Transport. In *IEEE international Conference on Image Processing*, Paris, France, October 2014.
- [21] RawTherapee. Film Simulation Pack (accessed 2019-12-13). https://rawpedia.rawtherapee.com/Film_Simulation, 2019.

- [22] E. Reinhard, M. Adhikhmin, B. Gooch, and P. Shirley. Color transfer between images. *IEEE Computer Graphics and Applications*, 21(5):34–41, July 2001.
- [23] C. E. Rodríguez-Pardo and G. Sharma. Color control functions for multiprimary displays—i: Robustness analysis and optimization formulations. *IEEE Transactions on Image Processing*, 29:1152–1163, 2020.
- [24] David Salomon and Giovanni Motta. *Handbook of Data Compression*. Springer Publishing Company, Incorporated, 5th edition, 2009.
- [25] Mark Shaw, Randall G Guay, Gaurav Sharma, and Aravindh BS Rajagopalan. Lossless compression of color look-up table via hierarchical differential encoding or cellular interpolative prediction, October 23 2012. US Patent 8,294,953.
- [26] Shutterstock. RocketStock, 35 Free LUTs for Color Grading (accessed 2019-06-24). <https://www.rocketstock.com/free-after-effects-templates/35-free-luts-for-color-grading-videos/>, 2019.
- [27] Chuohao Tang, Weibao Wang, Sean Collison, Mark Shaw, Jay Gondek, Amy Reibman, and Jan Allebach. ICC profile color table compression. In *Color and Imaging Conference, 24th Color and Imaging Conference*, volume 6, pages 260–265, 2016.
- [28] 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.
- [29] 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.
- [30] 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.
- [31] David Tschumperlé, Christine Porquet, and Amal Mahboubi. 3D Color CLUT Compression by Multi-scale Anisotropic Diffusion. In *CAIP 2019*, pages 3–14, Salerno, Italy, September 2019. CAIP 2019, Part II, LNCS 11679, pp. 3-14, 2019.
- [32] 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.

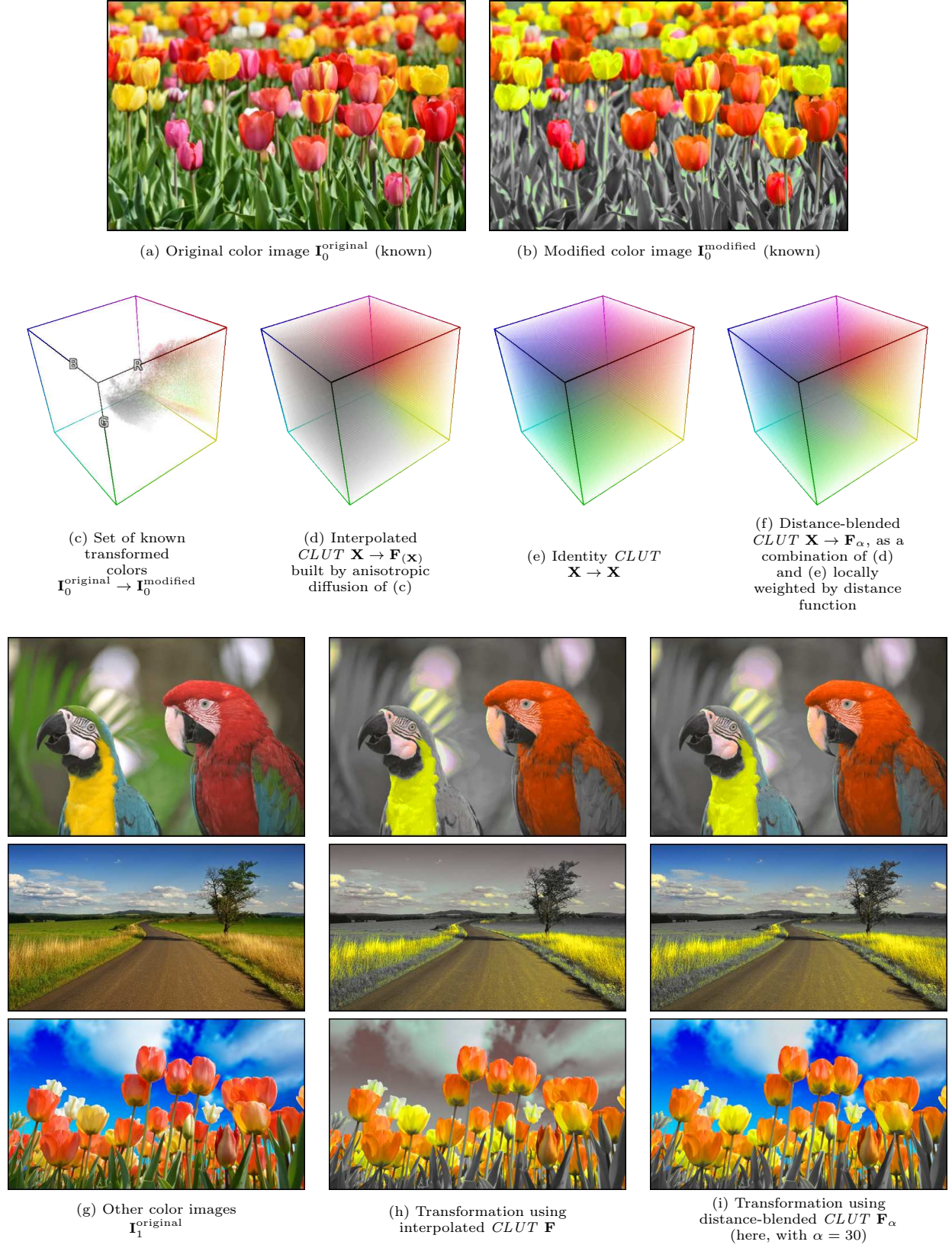


Figure 16: Exemplar-based $CLUT$ generation: principles and comparison between interpolated and distance-blended $CLUT$ s.