An Efficient 3D Color LUT Compression Algorithm Based on a Multi-Scale Anisotropic Diffusion Scheme
David Tschumperlé, Christine Porquet, Amal Mahboubi

To cite this version:
David Tschumperlé, Christine Porquet, Amal Mahboubi. An Efficient 3D Color LUT Compression Algorithm Based on a Multi-Scale Anisotropic Diffusion Scheme. Computer Analysis of Images and Patterns - 18th International Conference, CAIP 2019, Mario Vento and Gennaro Percannella, Sep 2019, Salerne, Italy. pp.3–14, 10.1007/978-3-030-29891-3_1. hal-02066484v3

HAL Id: hal-02066484
https://hal.archives-ouvertes.fr/hal-02066484v3
Submitted on 3 Apr 2019

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

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

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

Keywords: 3D CLUTs, generic color transformations, compression of smooth data, anisotropic diffusion.

1 Introduction

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

Let \(F : RGB \rightarrow RGB\) be a 3D CLUT. Applying \(F\) to a color image \(\mathbf{I} : \Omega \rightarrow RGB\) is done as follows:
\[
\forall \mathbf{p} \in \Omega, \quad \mathbf{I}^{\text{modified}}(\mathbf{p}) = F(I_R(\mathbf{p}), I_G(\mathbf{p}), I_B(\mathbf{p}))
\]
where \(I_R, I_G\) and \(I_B\) are the RGB color components of \(\mathbf{I}\). It should be noted that, most often, a CLUT is a volumic function that is continuous or, at worst, piecewise continuous (Fig.1). Fig.2 exhibits a small set of various colorimetric modifications done with CLUTs, taken from [2, 10]. It illustrates the large diversity of the effects that CLUTs allow, e.g. color fading, chromaticity boost, color inversion, hue shift, black-and-white conversion, contrast enhancement, etc.

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

![CLUT, visualized in 3D](image1.png)

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

Here, this issue is addressed: an efficient technique for CLUT compression is put forward, as well as the corresponding decompression method. Our algorithm takes a CLUT \(F\) as input and generates a smaller representation \(F_c\). The reconstruction algorithm operates on \(F_c\) to generate a reconstructed CLUT \(\tilde{F}\). Our compression scheme is said to be lossy\(^1\), as \(\tilde{F}\) is different from \(F\), but with an error that remains visually unnoticeable.
The paper is organized as follows: in Section 2, our

The idea of compressing/decompressing PDEs using anisotropic diffusion interpolation algorithm performing a dense reconstruction can be found in the literature. In [9], a lossless CLUT compression method is proposed; it is based on two different predictive coding schemes, the former being differential hierarchical coding and the latter cellular interpolative predictive coding. In both cases, a prior preprocessing step for data reorganization is needed. However, experimentations are only made on small-sized CLUTs (resolution $17^3$), and the lossless method leads to compression rates (around 30%) that are much less effective than what we get with our approach.

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

The paper is organized as follows: in Section 2 our CLUT re-

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

First, let us assume we have a set $\mathcal{K} = \{ \mathbf{K}_k \in RGB \times RGB \mid k = 1 \ldots N \}$ of $N$ color keypoints, located in the $RGB$ cube, such as $\mathcal{K}$ provides a sparse representation of a CLUT $\mathbf{F} : RGB \rightarrow RGB$.

The $k$th keypoint of $\mathcal{K}$ is defined by vector

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

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

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

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

$\mathbf{F}$ is reconstructed 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 $m(\mathbf{X}) = \begin{cases} 0 & \text{if } \exists k, \mathbf{X} = \mathbf{X}_k \\ 1 & \text{otherwise} \end{cases}$

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). A quite good estimate for $\mathbf{F}_{t=0}$ is actually obtained by propagating the colors $\mathbf{C}_k$.
inside the Voronoi cells associated to the set of points $X_k$ (for instance by watershed-like propagation $[5]$), then by smoothing it by an isotropic 3D gaussian filter (Fig.4). A more efficient multi-scale scheme for estimating $F_{\text{centered}}$ is detailed hereafter.

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

**Spatial discretization:** Numerically, $d_K$ is efficiently computed (in linear time) by a distance transform, such as the one proposed in $[9]$. 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 put forward:

$$\nabla d_K(X) = \begin{pmatrix} \maxabs(\partial_x^f d_K, \partial_x^b d_K) \\ \maxabs(\partial_y^f d_K, \partial_y^b d_K) \\ \maxabs(\partial_z^f d_K, \partial_z^b d_K) \end{pmatrix}$$

(2)

where

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

and

$$\partial_x^f d_K = d_K(x+1, y, z) - d_K(x, y, z)$$

$$\partial_x^b d_K = d_K(x-1, y, z) - d_K(x, y, z)$$

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.5 below.

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

$$F_{t+dt}^t = m(X) \left[ F_t^t(X+\eta) + F_t^t(X-\eta) - 2 F_t^{t+dt}(X) \right]$$

which leads to:

$$F_{t+dt}^{t+dt} = \frac{F_t^t(X) + dt m(X) [F_t^t(X+\eta) + F_t^t(X-\eta)]}{1 + 2 dt m(X)}$$

A major advantage of using such a semi-implicit scheme to implement the evolution of (1) is that you can choose $dt$ arbitrarily large, without loss of stability or significant decrease in quality on the diffusion process (as studied in $[6, 15]$). Therefore, we get the following simplified temporal discretization scheme:
\[
\begin{cases}
F^t(x) = F(x) & \text{if } m(x) = 0 \\
F^{t+dt}(x) = \frac{1}{2} \left[ F(x+\eta) + F(x-\eta) \right] & \text{otherwise}
\end{cases}
\]

where \(F(x+\eta)\) and \(F(x-\eta)\) are accurately estimated using tricubic spatial interpolation.

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

**Multi-scale resolution**: As with most numerical schemes involving diffusion PDEs \([12]\), it can be observed that the number of iterations of (3) required to converge towards a stable solution of (1) increases quadratically with the 3D resolution of the CLUT \(F\) to be reconstructed. In order to limit this number of iterations for high resolutions of CLUTs, we therefore suggest to solve (1) by a multi-scale ascending technique.

Rather than initializing \(F_{t=0}\) by watershed-like propagation for computing the diffusion at resolution \((2^4)^3\), \(F_{t=0}\) is estimated as a trilinear upscaling of the CLUT reconstructed at half resolution \((2^{4+1})^3\). The latter is closer to the stable state of the PDE (1) at resolution \((2^4)^3\), and the number of necessary iterations of (3) required 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 upsampled results obtained at resolutions \(2^4\), \(4^4\), \(8^3\) \ldots, until the desired resolution is reached (Fig. 6).

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

### 3 Generation of keypoints

Now that the reconstruction of a dense CLUT \(F\) 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 \(F\).

The compression algorithm detailed hereafter generates a set \(K\) of \(N\) keypoints representing a given input CLUT \(F\), such that the CLUT \(F_N\) reconstructed from \(K\) is close enough to \(F\), in the sense of two reconstruction quality criteria (which are set as parameters of the method). These quality criteria are chosen.
as: \( \Delta_{\text{max}} = 8 \), the maximum reconstruction error allowed at one point of \( RGB \), and \( \Delta_{\text{avg}} = 2 \), the average reconstruction error for the entire \( CLUT \) \( F \).

The algorithm consists of three distinct steps:

1. **Initialization**: The set \( K \) 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., \( K = \{(X_k, F(X_k)) \mid k = 1 \ldots 8\} \), for all \( X_k \) whose coordinates in \( x, y \) and \( z \) are either 0 or 255.

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

\[
E_N(X) = \|F(X) - \tilde{F}_N(X)\|
\]

respectively denote the maximum error and the average reconstruction error. As long as \( E_{\text{max}} > \Delta_{\text{max}} \) or \( E_{\text{avg}} > \Delta_{\text{avg}} \), a new keypoint \( F_{N+1} = (X_{N+1}, F_{N+1}(X_{N+1})) \) is added to \( K \), located at coordinates \( X_{N+1} = \arg\max_X \{E_N(X)\} \) with 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. 7).

3. **Deleting keypoints**: Sometimes, the addition of the last keypoint at step 2 leads to a \( CLUT \) reconstructed with a higher quality than expected, i.e., with \( E_{\text{max}} < \Delta_{\text{max}} - \epsilon \) and \( E_{\text{avg}} < \Delta_{\text{avg}} - \epsilon \) and a non negligible \( \epsilon > 0 \). In this case, there is usually a subset of \( K \) that also verifies the 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 \( K \). This is simply achieved by iteratively going through all the keypoints \( K_k \) of \( K \) (in the order of their insertion, \( k = 1 \ldots N \)), and checking whether the deletion of the \( k \)-th keypoint \( K_k \) allows to reconstruct a \( CLUT \) \( F_N \) with quality constraints that still hold. If this is the case, the keypoint \( K_k \) is discarded from \( K \) and the algorithm is resumed from where we left it. According to the degree of continuity of the \( CLUT \), this third step sometimes allows to withdraw up to 25% of keypoints in \( K \) (it also happens that no keypoint can be removed this way).

At the end of these three steps, we get a set or keypoints \( K \) representing a compressed lossy version of a \( CLUT \) \( F \), such that a minimum quality of reconstruction is guaranteed.

### 4 Results

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

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

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

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

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

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

5 Conclusions

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

Références