



HAL
open science

Example-Based Colour Transfer for 3D Point Clouds

Ific Goudé, Rémi Cozot, Olivier Le Meur, Kadi Bouatouch

► **To cite this version:**

Ific Goudé, Rémi Cozot, Olivier Le Meur, Kadi Bouatouch. Example-Based Colour Transfer for 3D Point Clouds. Computer Graphics Forum, 2021, 40 (6), pp.428-446. 10.1111/cgf.14388. hal-03396448

HAL Id: hal-03396448

<https://hal.science/hal-03396448v1>

Submitted on 22 Oct 2021

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.

Example-based color transfer for 3D point clouds

Ific Goudé¹, Rémi Cozot², Olivier Le Meur¹, and Kadi Bouatouch¹

¹Univ Rennes, CNRS, IRISA, Rennes, France

²Littoral Opal Coast University, Calais, France

9 August 2021

Abstract

Example-based color transfer between images, which has raised a lot of interest in the past decades, consists of transferring the color of an image to another one. Many methods based on color distributions have been proposed, and more recently, the efficiency of neural networks has been demonstrated again for color transfer problems. In this paper, we propose a new pipeline with methods adapted from the image domain to automatically transfer the color from a target point cloud to an input point cloud. These color transfer methods are based on color distributions and account for the geometry of the point clouds to produce a coherent result. The proposed methods rely on simple statistical analysis, are effective, and succeed in transferring the color style from one point cloud to another. The qualitative results of the color transfers are evaluated and compared with existing methods.

1 Introduction

Point clouds are now studied for decades and start to be a regular format used in research and industry. The emergence of new scanners, like Lidar technologies, helps a lot in making point cloud data format necessary. It is useful for capturing the surrounding

environment of autonomous vehicles, scanning sculptures to preserve the cultural heritage [35, 22], obtaining 3D models of buildings for renovation and construction [33, 9], and so on.

In computer graphics, 3D models are represented either by point clouds or by meshes. As meshes are made of a set of triangles connected through vertices, removing this connectivity results in a set of points in a 3D space. Point clouds are a more general representation of 3D models.

Indeed, an increasing number of 3D point clouds are generated by scanning real-world objects. Besides, several papers have tackled point cloud rendering, such as Levoy and Whitted [16], but only one, to the best of our knowledge, deals with color transfer between point clouds [5].

On the other hand, color transfer between images is a well-known process. Changing the color style of an input image according to the color style of a target one is the objective.

Many different approaches can be used to solve color transfer problems (distribution-based color transfer [27, 23, 11], histogram matching [24, 21], transport optimization [3], etc.) More recently, the use of Convolutional Neural Networks (CNNs) has proved its efficiency regarding image color transfer [6, 13].

In this paper, we focus on color transfer between 3D point clouds. To the best of our knowledge, the only existing attempt to perform a color transfer between 3D point clouds [5] is a CNN-based method originally used for point clouds classification [25]. As will



Figure 1: Rendering a church point cloud (input) with the color style of a fantasy house point cloud (target) using our MGD_N color transfer method.

be seen in Section 4.2, this method does not provide interesting results in terms of visual quality and metric values. This is why we propose new color transfer methods which account for the geometry of the 3D point clouds.

Our methods rely on the matching of color distribution variances and assume that the color channels follow Gaussian distributions. For our purpose, we want a color transfer coherent with the geometry of the point clouds by accounting for the normals to the surfaces. Before performing our color transfer, the normals of the input and the target point clouds are projected into the best fitting 3D coordinate system by a Principal Component Analysis (PCA) and decomposed into 6 directions. This pre-process improves the robustness of our method and allows a color transfer consistent with the geometry of the point clouds as detailed in Section 3.2.

The paper is organized as follows. Section 2 reviews previous work on image style transfer and point cloud rendering. Then, we present our color transfer pipeline for point clouds (considering both the color distributions and the point clouds geometry) and detail our two distribution-based methods in Section 3. The results of our methods and different applications are presented in Section 4. Thereafter, in Section 4.2,

we evaluate the performance of our methods and compare them with other techniques that consider or not the geometry. To conclude, the limits of our methods and future work are discussed in Section 5.

2 Related work

Color style transfer for 2D images. Over the past years, image style transfer aroused great interest [2]. Reinhard *et al.* [27] proposed a color transfer between images based on a simple statistical analysis. Color signals are supposed to follow Gaussian distributions and to be independent as they are first converted into the decorrelated $l\alpha\beta$ color space. The transfer is then achieved by matching the distributions of the target colors with the distributions of the input colors, as detailed in Section 3.3.1.

Later, Pitié and Kokaram [23] relaxed the independence assumption and supposed that the color signals follow a Multivariate Gaussian Distribution (MGD). The transfer is achieved using a linear transformation defined by the Monge-Kantorovich closed-form matrix. The authors experimented with several color spaces and obtained better results when using the *CIELAB* space. This transformation is detailed in Section 3.3.2.

Hristova *et al.* [10, 11, 12] went further by considering the color channels as a Multivariate Generalized Gaussian Distribution (MGGD) or as a beta distribution. Relaxing the shape parameter of the MGGD allowed them to improve the style transfer between images and even proposed to perform an n-dimensionality style transfer. A 5 dimensions transfer of image colors and gradients is given as an example in their work.

More recently, CNNs have proved their efficiency in addressing style transfer problems. Gatys *et al.* [6] used a CNN optimized for object recognition to separate the semantic content of images from the style. In their method, high-level features, corresponding to the content, are extracted from the input image while texture information of the target image is given by a Gram matrix. An iterative process of gradient descent gives the final stylized image that results from a linear combination of the content and the style features. Johnson *et al.* [13] achieved similar qualitative results with a faster method. Furthermore, CNN-based style transfer has been extended to different domains, like fluids simulations [15] or interactive videos [32].

Among all these existing color transfer methods, we have extended two of them [27, 23] to color transfer between point clouds. The choice of these two methods has been motivated by their efficiency and their intensive use by the computer graphics community.

Style-based rendering of 3D point clouds.

While many research works seek to improve the quality of point clouds rendering (holes filling [26, 14, 19], reducing edges aliasing [7], drawing curved shapes instead of circles as a primitive [31]), only a few attempts deal with point clouds stylization.

Xu *et al.* [34] proposed to render point clouds silhouette to get a cartoon style. Their method is based on two rendering passes. Points are firstly rendered in black color with a large radius while the second pass performs a rendering of the points with their appropriate colors and radius. Flat surfaces are then filled in with the right content while the edges are surrounded by a black silhouette.

A few years later, Rosenthal and Linsen [28] proposed

to improve point clouds rendering in the image space. A point cloud is rendered before proceeding to holes filling, edge detection, and anti-aliasing in the image space. Silhouette rendering is then a direct application of edge detection.

Recently, Sabbadin *et al.* [30] proposed a method for relighting point clouds with High Dynamic Range (HDR) environments in real-time. Considering a captured point cloud with a Limited Dynamic Range as a relighting environment, they expand its dynamic range thanks to a single HDR image to obtain an HDR point cloud. Then, this HDR point cloud is used to relight virtual objects in real-time, realistically simulating the global illumination due to the environment.

Color style transfer for 3D point clouds.

After a first attempt in transferring textures style for 3D models [18], a first study finally focused on point clouds style transfer. Cao *et al.* [5] proposed *PSNet*, a color transfer network for point cloud stylization. Based on a network trained for point clouds classification and segmentation [25], they used the ability of this network to capture high-level features [8] to transfer either the color or the geometry (or both) of a target point cloud to an input point cloud. Regarding color transfer, they also add the possibility to consider a simple image as a target where each pixel corresponds to a 3D point. As in [6], Cao *et al.* used the Gram matrix of features map to represent the color style of the target.

Nevertheless, *PSNet* is efficient for point clouds similar to those used by its training dataset [4], which consists of single objects across 16 categories (lamp, chair, table, etc.), while our methods can be used regardless of the nature of the point clouds. Furthermore, while a deep learning approach would rely on high-level features such as semantic meanings, the two methods we propose are based on correlations between colors and normals to perform the color transfer. We show later, by qualitative and quantitative measures, that our methods produce better results than *PSNet*. In the following section, we present our simple, efficient, and formally defined color transfer technique for point clouds.

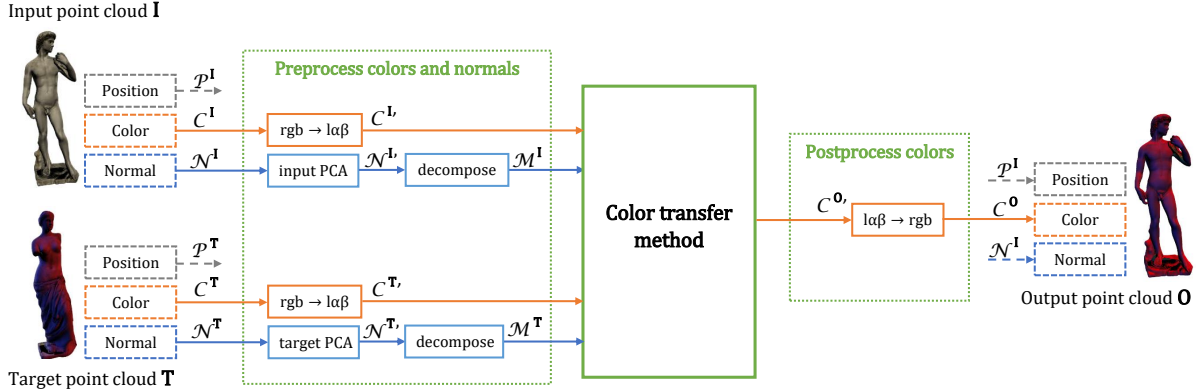


Figure 2: Our point clouds color transfer consists of three steps: First, normals and colors of both input \mathbf{I} and target \mathbf{T} point clouds are transformed. Then, a color transfer method relying on a statistical analysis of color distributions and accounting for the point clouds geometry is performed. Finally, the output color is converted back to RGB and applied to the output point cloud \mathbf{O} .

3 Point clouds color transfer

We are interested in transferring color style from a 3D model to another. Actually, 3D models are generally represented by meshes, a set of triangles connected through vertices. However, we decided to apply the color transfer to 3D point clouds, which is becoming a largely used standard because of its more general and compact format.

Furthermore, we assume that color transfer between 3D point clouds depends on their geometry. There are at least two features that characterize the geometry: positions and normals. We have chosen to rely on normals because of their robustness. Indeed, normals are efficient as they characterize the local shape of a surface. Besides, unlike positions, normals are invariant under scaling and translation.

Before elaborating on the proposed pipeline, we formally define the term of 3D point clouds. A point cloud is a set of data points in a 3-dimensional space. A point is generally defined by its position in space (x, y, z) , its color (r, g, b) and the direction of its normal to the surface (n_x, n_y, n_z) . The main goal is to transfer the color style of a target point cloud \mathbf{T} to an input point cloud \mathbf{I} .

Let $\mathcal{P}^{\mathbf{I}}$, $\mathcal{C}^{\mathbf{I}}$ and $\mathcal{N}^{\mathbf{I}}$ be the set of positions, colors

and normals vectors of the input point cloud respectively. $\mathcal{P}^{\mathbf{I}} = (\mathbf{p}_1^{\mathbf{I}}, \dots, \mathbf{p}_N^{\mathbf{I}})$, with $\mathbf{p}_i^{\mathbf{I}} = (x_i^{\mathbf{I}}, y_i^{\mathbf{I}}, z_i^{\mathbf{I}})$; $\mathcal{C}^{\mathbf{I}} = (\mathbf{c}_1^{\mathbf{I}}, \dots, \mathbf{c}_N^{\mathbf{I}})$, with $\mathbf{c}_i^{\mathbf{I}} = (r_i^{\mathbf{I}}, g_i^{\mathbf{I}}, b_i^{\mathbf{I}})$; and $\mathcal{N}^{\mathbf{I}} = (\mathbf{n}_1^{\mathbf{I}}, \dots, \mathbf{n}_N^{\mathbf{I}})$, with $\mathbf{n}_i^{\mathbf{I}} = (n_{xi}^{\mathbf{I}}, n_{yi}^{\mathbf{I}}, n_{zi}^{\mathbf{I}})$ for $i \in \{1, \dots, N\}$ where N is the number of points of the input point cloud.

In the same way, let $\mathcal{P}^{\mathbf{T}}$, $\mathcal{C}^{\mathbf{T}}$ and $\mathcal{N}^{\mathbf{T}}$ be respectively the set of positions, colors and normals vectors of the target point cloud. $\mathcal{P}^{\mathbf{T}} = (\mathbf{p}_1^{\mathbf{T}}, \dots, \mathbf{p}_M^{\mathbf{T}})$, with $\mathbf{p}_j^{\mathbf{T}} = (x_j^{\mathbf{T}}, y_j^{\mathbf{T}}, z_j^{\mathbf{T}})$; $\mathcal{C}^{\mathbf{T}} = (\mathbf{c}_1^{\mathbf{T}}, \dots, \mathbf{c}_M^{\mathbf{T}})$, with $\mathbf{c}_j^{\mathbf{T}} = (r_j^{\mathbf{T}}, g_j^{\mathbf{T}}, b_j^{\mathbf{T}})$; and $\mathcal{N}^{\mathbf{T}} = (\mathbf{n}_1^{\mathbf{T}}, \dots, \mathbf{n}_M^{\mathbf{T}})$, with $\mathbf{n}_j^{\mathbf{T}} = (n_{xj}^{\mathbf{T}}, n_{yj}^{\mathbf{T}}, n_{zj}^{\mathbf{T}})$ for $j \in \{1, \dots, M\}$ where M is the number of points of the target style point cloud. Note that, the exponent T is an index standing for target rather than for transposed matrix, and the number of points N in the input point cloud is not necessarily the same as the number of points M in the target point cloud.

3.1 Method overview

In this paper, we propose two point clouds color transfer methods. As they account for the point clouds geometry, the color transferred to a point depends on the direction of its normal. Our two methods, defined in Section 3.3, are based on a statistical

model applied to the color distributions of the point clouds. While our first method is a geometric approach, our second method relies on the correlations between color channels and normal directions.

The general flowchart of our color transfer is presented in Figure 2 and consists of three steps.

First, we project both colors and normals vectors into new spaces where the components are less correlated, which makes the color transfer more robust. More precisely, for both input and target point clouds, the color (orange boxes in Figure 2) of each point is converted from the RGB space to the decorrelated $l\alpha\beta$ space [29], while the normal (blue boxes in Figure 2) is projected to the best fitting 3D coordinate system by PCA before being decomposed to get a 6-dimensional vector to separate its positive and negative components. A 3D coordinate system is a set of 3D basis vectors called from now on **basis**.

Second, we want the shape of the input’s color distribution to fit with the shape of the target’s one, depending on the direction of the points normal. To do so, we propose two methods that make different assumptions regarding the shape of the distributions of the point clouds colors. The first method supposes that the color channels are independent and follow Gaussian distributions. We propose to perform six different color transfers depending on the direction of the point’s normal as detailed in Section 3.3.1. The second supposes that points normals and colors follow a Multivariate Gaussian Distribution. In that case, we give the input distribution the aspect of the target’s one thanks to a linear transformation, as proposed by Pitié and Kokaram [23]. To do so, the colors and the normals of point clouds are concatenated to obtain a 9-dimensional probability distribution for both input and target point clouds. A linear transformation is computed between the covariance matrices of these two 9D distributions as explained in Section 3.3.2.

Finally, regardless of the used method, we convert the colors of the output back to the RGB color space to obtain the resulting point cloud.

All the steps of our pipeline are detailed in the following sections.

3.2 Making point clouds components decorrelated

To make our color transfer more robust, we aim to decorrelate the color channels from each other, as well as normal directions.

Regarding color, we used the perceptually decorrelated $l\alpha\beta$ color space [29]. The calculus of the color space transform stems from [27].

On the other hand, the point clouds normals are not necessarily well aligned with the origin axes, i.e. they are strongly correlated. So, they are firstly projected into the best fitting basis using a PCA, as explained in the following section. Then, the normal vectors are assigned 6 components corresponding to the 6 axes directions of the basis by separating the positive from the negative directions, as detailed in Section 3.2.2.

3.2.1 Transforming normals basis by PCA

To make our methods more efficient in determining correlations between color channels and normal directions, we project the normals into the best fitting basis by PCA before proceeding to the color transfer. A PCA is computed by performing an eigenvalue decomposition on the covariance matrix of the normals and used to project the normals into a new orthogonal basis that minimizes the distance from normals to origin axes. In other words, a PCA defines a more representative basis for the set of normals. Two PCA are applied: one to the input point cloud and another to the target one.

Figure 3 shows the result of our color transfer (using our MGD_N method as explained later) applied to an input point cloud that is not well aligned with the original basis, i.e. the coordinate system in which the point cloud is defined. By alignment, we mean the normals of the point cloud are aligned with the origin axes of the basis. In our example, while the target (Figure 3a) is aligned with its original basis, the input (Figure 3b) is not aligned with its original basis. However, we expect the right, up, and forward faces of the input point cloud to be colored in red, yellow, and blue, in accordance with the faces’ color of the target point cloud.

As the input point cloud is not well aligned with the

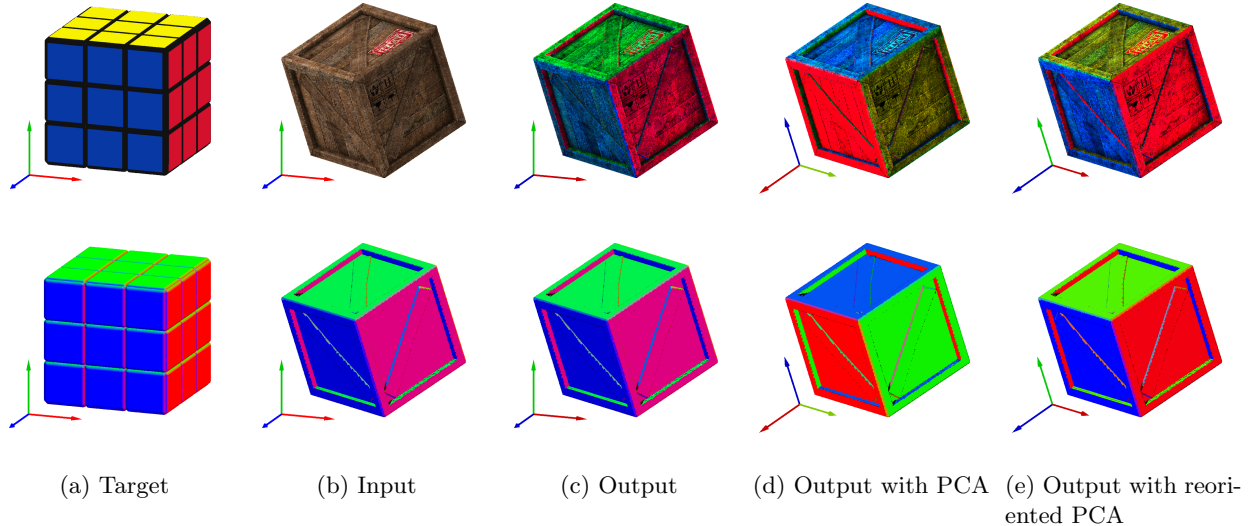


Figure 3: Red, green, and blue arrows correspond to x , y , and z axes of the basis respectively. First row: the colored point clouds. Second row: the corresponding point clouds normals (normals components x , y and z are colored in red, green, and blue respectively). The target point cloud (3a) is well aligned with the original basis: its normals are oriented toward the x direction for the right face (red), the y direction for the up face (green), and the z direction for the forward face (blue). As the input (3b) is not well aligned with the original basis (its normals are in between x , y and z axes), the color transfer fails in reproducing the colors of the target on each face of the output (3c). Transforming the normals by PCA (3d) allows a more coherent color transfer depending on the normals' direction, but the global orientation is not preserved (axes are inverted). Automatically mapping each axis given by the PCA with the closest axis of the original basis results in the expected color transfer (3e).

original basis, the faces' colors of the output (Figure 3c) result from the blending of different colors from the target point cloud (Figure 3a). Instead of appearing red, the right face of the output appears purple, a blending between the blue and red faces of the target, as well as the up face, appears green, a blending between the yellow and blue faces of the target.

Projecting the input normals into a more representative basis using a PCA results in well-transferred colors for each face (Figure 3d). Nevertheless, while the directions of the original basis are fixed (in our case: x right, y up, and z forward; respectively red, green, and blue arrows), the transformation of the normals produced by the PCA does not conserve the original global orientation (axes may be inverted).

Therefore, as the color transfer depends on the direction of the normals, the color of the target in a particular direction is transferred to the input in the same direction. For example, target points with normals oriented toward the up (y axis) are colored in yellow. This color is transferred to the points with normals oriented toward the new y axis defined by the PCA of the output (the forward direction in Figure 3d). Hence, the direction of the normals should match between the two point clouds.

For this reason, we associate each axis, resulting from the application of PCA, with the closest axis of the original basis using a cosine distance (Figure 3e). Indeed, as axes resulting from the PCA can be positively or negatively oriented, we retain the orientation which maximizes the cosine distance to an axis

of the original basis (x , y , or z). As a PCA defines an orthogonal basis, the mapping ensures that each axis defined by the PCA matches exactly with one axis of the original basis. Rather than using PCA, users can also manually associate the axes, or apply a rotation to the point clouds, to obtain specific results.

In conclusion, once the normals are transformed by PCA and reoriented to match with the original basis, we decompose the normal vectors into the 6 directions of the basis to separate the positive from negative components. This decomposition is detailed in the following.

3.2.2 Decomposing normals

Because our methods rely on correlations between colors and normals, we decompose the normal components by separating its positive and negative values. This process increases the robustness of the statistical models we used and provides a color transfer consistent with the geometry of the point clouds. The proposed decomposition is simple and consists of a 6-dimensional normal vector defined as follows. Given a 3-dimensional normal vector $\mathbf{N} = (x, y, z)$ with values in the range $(-1,1)$, a 6-dimensional normal vector \mathbf{M} will be equal to:

$$\mathbf{M} = (|min(x, 0)|, |min(y, 0)|, |min(z, 0)|, |max(x, 0)|, |max(y, 0)|, |max(z, 0)|), \quad (1)$$

with values in the range $(0,1)$. Following that, all input and target normal vectors are decomposed to get 6-dimensional vectors.

Figure 4 visually compares the color transfer considering 3-dimensional normal vectors with the one considering 6-dimensional normal vectors \mathcal{N}' and \mathcal{M} in Figure 2 respectively). In addition, the correlation coefficient values from covariance matrices between colors (in RGB color space) and normals (3D or 6D) are given in Table 1. For these results we used our $MGD_{\mathcal{N}}$ method as explained later. As an example, the faces of the Rubik’s cube, oriented in the x positive and x negative directions, are both composed of red color (red for $+x$ and orange for $-x$). If we consider 3D normal vectors, the correlation coefficient value between the red color and the x axis

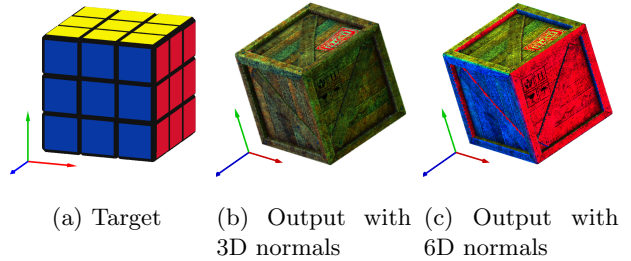


Figure 4: Correlations between colors and normals are better captured when decomposing the normals into 6 directions as shown in Table 1. Decomposing normals results in a more coherent color transfer.

is low (-0.023), which means there is no correlation between these two components, i.e. no red color will be transferred on these faces (see Figure 4b). On the other hand, when the normals are 6D vectors, the correlation coefficients (see covariance matrices in Section 3.3.2) between red color and $+x$ direction and $-x$ direction are both positive (0.246 and 0.159 respectively). When considering 6D normal vectors, the color transfer preserves the red color for both faces, which results in an efficient color transfer (see Figure 4c and Table 1).

To sum up, our color transfer methods rely on the correlations between color terms (in $l\alpha\beta$) and geometric terms (normals transformed by PCA and decomposed into 6 components). In the following section, we detail our two methods of color transfer.

3.3 Color transfer methods

Our two methods are extensions of color transfer techniques, applied to the geometry of point clouds, that rely on a statistical analysis of color distributions considering the normals of the point clouds. The first one (Section 3.3.1) assumes that the color channels are independent and follow Gaussian distributions. The second (Section 3.3.2) assumes that the 9-dimensional concatenated vectors of colors and normals follow a Multivariate Gaussian Distribution. The two following sections are organized as follows: First, we present the existing color transfer method that we have applied to point cloud colors. Then, we










		Correlation coefficient			Color
		red	green	blue	
3D normals \mathcal{N}'	x	-0.023	-0.160	0.080	
	y	0.072	0.054	-0.530	
	z	0.016	0.295	-0.262	
6D normals \mathcal{M}	-x	0.246	-0.112	0.283	
	-y	0.131	0.209	0.532	
	-z	-0.352	0.224	-0.101	
	+x	0.159	-0.364	-0.152	
	+y	0.246	0.297	-0.313	
	+z	-0.377	-0.249	0.318	

Table 1: Correlation coefficient values between RGB colors and 3D normals or 6D normals for the target point cloud (Figure 4a). Colors in the last column result from the three RGB correlation coefficients normalized between 0 and 1. The color obtained for each direction of the 6D normals corresponds to the color of the Rubik’s cube’s face in this direction.

go further by improving this method by leveraging the point cloud geometry. puted as follows:

$$\begin{aligned}
l_i^O &= (l_i^I - \mu_l^I) \times \frac{\sigma_l^T}{\sigma_l^I} + \mu_l^T, \\
\alpha_i^O &= (\alpha_i^I - \mu_\alpha^I) \times \frac{\sigma_\alpha^T}{\sigma_\alpha^I} + \mu_\alpha^T, \\
\beta_i^O &= (\beta_i^I - \mu_\beta^I) \times \frac{\sigma_\beta^T}{\sigma_\beta^I} + \mu_\beta^T.
\end{aligned} \tag{2}$$

3.3.1 First method: Independent Gaussian Distributions

Color based Independent Gaussian Distributions. Our first method relies on the color transfer approach proposed by Reinhard *et al.* [27], called *IGD* (Independent Gaussian Distributions) from now on. Let us summarize this approach before presenting our first method. The goal of this *IGD* method is to transform the shapes of each color channel distribution of the input point cloud to match as much as possible with the corresponding target color channel distribution. Assuming that the color channels are independent, Reinhard *et al.* suppose that each color channel follows a Gaussian distribution.

For this purpose, the mean μ and the standard deviation σ of each component of the $l\alpha\beta$ channels are computed, for both input and target point clouds. Then, the resulting output color channels are com-

puted as follows: Finally, we defined the output color $\mathbf{c}_{i'}^O = (l_i^O, \alpha_i^O, \beta_i^O)$. At this stage, the color transfer applied to the point clouds does not account for geometry.

Color and geometry based Independent Gaussian Distributions. Hereafter, we describe our first method, called from now on *IGD_N*. We perform the color transfer depending on both color and geometry. To do so, we compute the means and the standard deviations of each of the three $l\alpha\beta$ component weighted by the six components of the 6D normal vector in Equation 1.

It results in 18 means μ_{kc} and 18 standard deviations σ_{kc} for both the input and target point clouds, where the index c corresponds to one of the three color component $l\alpha\beta$, and the index k corresponds to one of the six component of the normal

$(-x, -y, -z, +x, +y, +z)$:

$$\begin{aligned}\mu_{kc}^I &= \frac{1}{\sum_{i=0}^N n_{ki}^I} \sum_{i=0}^N n_{ki}^I \cdot c_i^I, \\ \sigma_{kc}^I &= \frac{1}{\sum_{i=0}^N n_{ki}^I} \sum_{i=0}^N n_{ki}^I \cdot (c_i^I - \mu_{kc}^I)^2,\end{aligned}\quad (3)$$

$$\begin{aligned}\mu_{kc}^T &= \frac{1}{\sum_{j=0}^M n_{kj}^T} \sum_{j=0}^M n_{kj}^T \cdot c_j^T, \\ \sigma_{kc}^T &= \frac{1}{\sum_{j=0}^M n_{kj}^T} \sum_{j=0}^M n_{kj}^T \cdot (c_j^T - \mu_{kc}^T)^2.\end{aligned}\quad (4)$$

Once the 18 μ_{kc} and 18 σ_{kc} are computed for both the input and target point clouds, we perform the three color transfers of each input point for the six directions using Equation 2, resulting in six new $l_{ki}^O, \alpha_{ki}^O, \beta_{ki}^O$ colors per point:

$$\begin{aligned}\mathbf{l}_i &= (l_{-xi}^O, l_{-yi}^O, l_{-zi}^O, l_{+xi}^O, l_{+yi}^O, l_{+zi}^O), \\ \alpha_i &= (\alpha_{-xi}^O, \alpha_{-yi}^O, \alpha_{-zi}^O, \alpha_{+xi}^O, \alpha_{+yi}^O, \alpha_{+zi}^O), \\ \beta_i &= (\beta_{-xi}^O, \beta_{-yi}^O, \beta_{-zi}^O, \beta_{+xi}^O, \beta_{+yi}^O, \beta_{+zi}^O).\end{aligned}\quad (5)$$

Finally, the output $l_i^O, \alpha_i^O, \beta_i^O$ colors result from a linear combination between the color components and the values of the 6D normal vector:

$$\begin{aligned}l_i^O &= \langle \mathbf{m}_i^{\mathbf{I}}, \mathbf{l}_i \rangle, \\ \alpha_i^O &= \langle \mathbf{m}_i^{\mathbf{I}}, \alpha_i \rangle, \\ \beta_i^O &= \langle \mathbf{m}_i^{\mathbf{I}}, \beta_i \rangle.\end{aligned}\quad (6)$$

This scalar product ($\langle \cdot, \cdot \rangle$) between normals and color components ensures a smooth color transition for points with a normal direction in-between 2 or 3 axes. Finally, the output color $\mathbf{c}^{\mathbf{O}'} = (l_i^O, \alpha_i^O, \beta_i^O)$.

3.3.2 Second method: Multivariate Gaussian Distribution

Color based Multivariate Gaussian Distribution. Our second method relies on the color transfer approach proposed by Pitié and Kokaram [23], called *MGD* (Multivariate Gaussian Distribution) from now on. Let us summarize this approach before presenting our second method. The goal of this

transformation is to match an input MGD (Multivariate Gaussian Distribution) with a target MGD using the Monge-Kantorovich closed-form mapping. Without considering the geometry, given the means (μ) and covariances (Σ) of input and target $l\alpha\beta$ color distributions, the transformation matrix of Monge-Kantorovich mapping is computed as:

$$\mathbf{M} = \sum_{\mathcal{C}^{\mathbf{I}'}}^{-1/2} \left(\sum_{\mathcal{C}^{\mathbf{I}'}}^{1/2} \sum_{\mathcal{C}^{\mathbf{T}'}} \sum_{\mathcal{C}^{\mathbf{I}'}}^{1/2} \right)^{1/2} \sum_{\mathcal{C}^{\mathbf{I}'}}^{-1/2} \quad (7)$$

with $\mathcal{C}^{\mathbf{I}'}$ and $\mathcal{C}^{\mathbf{T}'}$ the sets of $l\alpha\beta$ color vectors for the input and target point clouds respectively, as defined in Section 3. Finally, the new set of color vectors for the output point cloud $\mathcal{C}^{\mathbf{O}'} = (\mathbf{c}^{\mathbf{O}'}_1, \dots, \mathbf{c}^{\mathbf{O}'}_{N'})$ are computed as:

$$\mathbf{c}^{\mathbf{O}'}_i = (\mathbf{c}^{\mathbf{I}'}_i - \mu_{\mathcal{C}^{\mathbf{I}'}}) \cdot \mathbf{M} + \mu_{\mathcal{C}^{\mathbf{T}'}}, \quad i \in \{1, \dots, N\}, \quad (8)$$

with $\mathbf{c}^{\mathbf{I}'}_i$, $\mu_{\mathcal{C}^{\mathbf{I}'}}$ and $\mu_{\mathcal{C}^{\mathbf{T}'}}$ are the input color in $l\alpha\beta$, the mean color of the input and the mean color of the target respectively.

Color and geometry based Multivariate Gaussian Distribution. Let us describe our second method, called from now on *MGD_N*. To consider the point clouds geometry, we extend the MGD method by concatenating colors and normals vectors to obtain sets of 9-dimensional vectors $f^{\mathbf{I}}$ and $f^{\mathbf{T}}$ for the input and target point clouds respectively where:

$$\begin{aligned}f_i^{\mathbf{I}} &= (l_i^I, \alpha_i^I, \beta_i^I, n_{-xi}^I, n_{-yi}^I, n_{-zi}^I, n_{+xi}^I, n_{+yi}^I, n_{+zi}^I), \\ f_j^{\mathbf{T}} &= (l_j^T, \alpha_j^T, \beta_j^T, n_{-xj}^T, n_{-yj}^T, n_{-zj}^T, n_{+xj}^T, n_{+yj}^T, n_{+zj}^T).\end{aligned}\quad (9)$$

Given the means $\mu_{f^{\mathbf{I}}}$, $\mu_{f^{\mathbf{T}}}$ and the covariance matrices $\Sigma_{f^{\mathbf{I}}}$, $\Sigma_{f^{\mathbf{T}}}$ of the input and target distributions $f^{\mathbf{I}}$ and $f^{\mathbf{T}}$ respectively, the transformation of the input distribution is given by:

$$\mathbf{M} = \sum_{f^{\mathbf{I}}}^{-1/2} \left(\sum_{f^{\mathbf{I}}}^{1/2} \sum_{f^{\mathbf{T}}} \sum_{f^{\mathbf{I}}}^{1/2} \right)^{1/2} \sum_{f^{\mathbf{I}}}^{-1/2}, \quad (11)$$

and the samples of the output distribution are computed as:

$$f_i^{\mathbf{O}'} = (f_i^{\mathbf{I}} - \mu_{f^{\mathbf{I}}}) \cdot \mathbf{M} + \mu_{f^{\mathbf{T}}}, \quad i \in \{1, \dots, N\}, \quad (12)$$

where $f^{\mathbf{O}}$ is the set of 9-dimensional output vectors. From this set of 9-dimensional output vectors, each vector $f_i^{\mathbf{O}}$ is composed of 3 color components and 6 normal components. By preserving only the 3 color components, we obtain the new set of output colors vectors $\mathcal{C}^{\mathbf{O}'} = (\mathbf{c}^{\mathbf{O}'_1}, \dots, \mathbf{c}^{\mathbf{O}'_{N'}})$, where each $\mathbf{c}^{\mathbf{O}'_i} = (l_i^{\mathbf{O}}, \alpha_i^{\mathbf{O}}, \beta_i^{\mathbf{O}})$.

3.4 Applying output colors to point cloud

The color transfer is finalized by applying the new color to the input point cloud. In the final step, the $l\alpha\beta$ output colors are converted back to the *RGB* colors following [27].

To sum up, each of our two methods consists of three steps:

1. The colors and normals of both the input and target point clouds are projected into more decorrelated spaces (PCA + decomposition)
2. The color transfer is performed between the input and target using one of our proposed method (*IGD_N* or *MGD_N*)
3. The output colors of the points are converted back to the *RGB* color space

4 Results

4.1 Qualitative comparison

In the following, we present several examples of point cloud color transfers performed by our two methods. Our proposed color transfers for point clouds can be used for building colorization as illustrated in Figure 5. In the first row, the color style of the fantasy house point cloud is transferred to the input church point cloud. In addition to changing the color of roofs and walls, our methods also change the lighting to fit better with the target color style. Even color variations on flat surfaces are transferred (yellow insets). In this example, *MGD_N* better reproduces the colors of the target. The performance of our *MGD_N* method regarding building colorization is confirmed

by the example of the second row.

Another application is the color transfer between two inner rooms (see Figure 6). While our two methods perform well in transferring the colors of the king’s room to the red saloon, the result produced with the *IGD_N* method is a bit more reddish than the result produced with the *MGD_N* method. In a second example, presented in Figure 7, the color of a room with the style of Van Gogh is transferred to a scan of a painted kitchen. The difference between the wood color of the floor and the blue color of the walls is well preserved in the resulting point clouds. Once again, the global color seems more coherent when using our *MGD_N* method.

The color transfer can also be done for exterior environments as illustrated in Figure 8. It can be useful where shooting a movie in a particular environment (the Mar Saba monastery in Cisjordan for example) and wanting to give it the color style of a different region (the Momoyama castle in Japan here). In this case, the result produced with our *MGD_N* method has more color variations, which better corresponds to the color style of the target. Another example is showed in Figure 9. Both input and target point clouds are scans of castles. Color variation between walls, path, and grass are well transferred. Even the shadows are smoothed to better match the target color style.

Finally, we used our methods for color transfer between furniture, like chairs, as illustrated in Figure 10. For the example of the first row, while the colors globally correspond to the target color style for both of our methods, the different parts of the chairs are not well separated. As our methods only rely on low-level features (i.e. normal directions and color distributions), in some cases the colors of the target may be transferred to unexpected parts of the input point cloud. As a result, we expect the color of the chair legs to be whiter and the color of the leather to have pink nuances. For the second example (second row), the result produced when using *IGD_N* encounters the same problem as in the previous example. The color of the bamboo of the target beach chair is transferred to the fabric of the input chair and blue nuances from the fabric of the beach chair are transferred to the legs of the input chair. When using



Figure 5: Color transfer for building colorization. Our MGD_N method better transfers the colors of the target point clouds.



Figure 6: Color transfer between inner rooms. IGD_N produces a slightly reddish result while MGD_N produces a faithful color transfer between the two rooms.



Figure 7: The input point cloud is a scan of a painted kitchen and the target is a bedroom with the color style of Van Gogh. The wood color of the floor and the blue color of the walls are well transferred with our two methods.

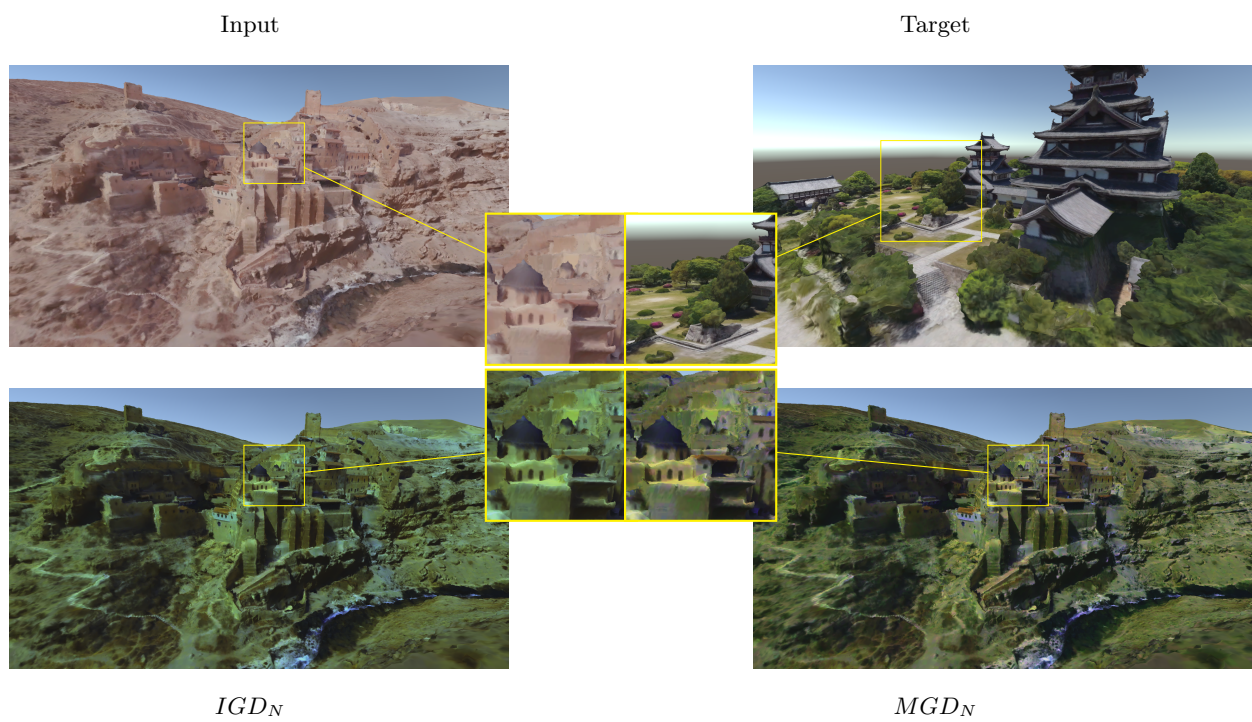


Figure 8: Color transfer between exterior environments. The color of the Momoyama castle in Japan is transferred to the Mar Saba monastery in Cisjordan.



Figure 9: The point clouds are scans of castles. While the input shows sunny weather, the target is a scan of a castle in Norway, with snowy weather. Our two methods succeed in reproducing the snowy weather.



Figure 10: Color transfer between chairs. As our methods only rely on low-level features (i.e. normal directions and color distributions), they cannot ensure a meaningful color transfer, resulting in mismatches (i.e. the fabric color transferred to the legs or inversely).

MGD_N , the colors are better separated, the fabric is composed of blue and white, and the legs tend to yellow color as expected.

To conclude, both proposed methods produce qualitative color transfer depending on the considered point clouds. However, it is difficult to know why one of the methods performed better than the other in some cases. For the sake of completeness, we evaluated several techniques for transferring color between point clouds.

4.2 Quantitative comparison with other methods

In this section, we evaluate the quality of the color transfer performed by the methods listed below:

- *PSNet*: CNN-based method proposed by *Cao et al.* [5]
- *IGD*: Independent Gaussian Distributions not accounting for normals (based on Reinhard *et al.* [27])
- *MGD*: Multivariate Gaussian Distribution not accounting for normals (based on Pitié and Kokaram [23])
- IGD_N : Independent Gaussian Distributions accounting for normals (our first method)
- MGD_N : Multivariate Gaussian Distribution accounting for normals (our second method).

Figure 11 illustrates a resulting color transfer for each method. The color distributions in $l\alpha\beta$ are displayed for the six directions of the basis.

We first discuss the processing time of those five color transfer methods and then, for contrasting the different methods, we propose several metrics that compare the color distributions of the target point cloud with the color distributions of the output point clouds. We expect the color distribution of the output to be close to the color distribution of the target. To measure the similarity between these two color distributions, we adapt objective metrics used in the field of image color transfer. The value provided by a metric will be called score from now on.

We propose three different approaches to calculate the score of similarity between the color distributions of two point clouds. First, the similarity score is calculated on the global color distribution of the point clouds (i.e. all the points of the cloud are considered), as it is done for images. Second, several viewpoint-dependent similarity scores are calculated on subsets of the points, the final similarity score is then the average of all the viewpoint scores as detailed later. A third similarity score can be computed in the case of color transfer for relighting and delighting. The expected result (i.e. the ground truth) can be simulated, so the color of each output point can be compared with the color of the corresponding point of the ground truth, as explained below.

In the following, as the evaluated point clouds are well aligned with their original basis, it was not necessary to make use of a PCA.

4.2.1 Processing time

As our methods are based on normals and colors means, standard deviations, and covariance matrices, the computation is very quick and has a linear complexity of $\mathcal{O}(N + M)$, with N and M the number of points for the input and target point clouds respectively. As an example, for point clouds (input and target) that have between 10^3 and 10^5 points, the color transfer produced by our *IGD*, *MGD*, and MGD_N methods takes 3.5 seconds maximum (7.5 seconds maximum for our IGD_N method). Furthermore, once the means and the standard deviations (Equations 2, 3 and 4 for *IGD* and IGD_N methods), or the Monge-Kantorovich closed-form matrix (Equations 7 and 11 for *MGD* and MGD_N methods) are calculated, the color transfer can be applied in real-time when rendering the output point cloud.

On the other hand, *PSNet* relies on CNNs and the result of the color transfer is calculated thanks to an iterative process (in our results, the maximum number of iteration was fixed to 2000 and the stop condition was a value difference under 10^{-7} between the previous and the current values of the loss function). Moreover, the operations made on the input and the target point clouds are different and the computation time is difficult to estimate. On average, for

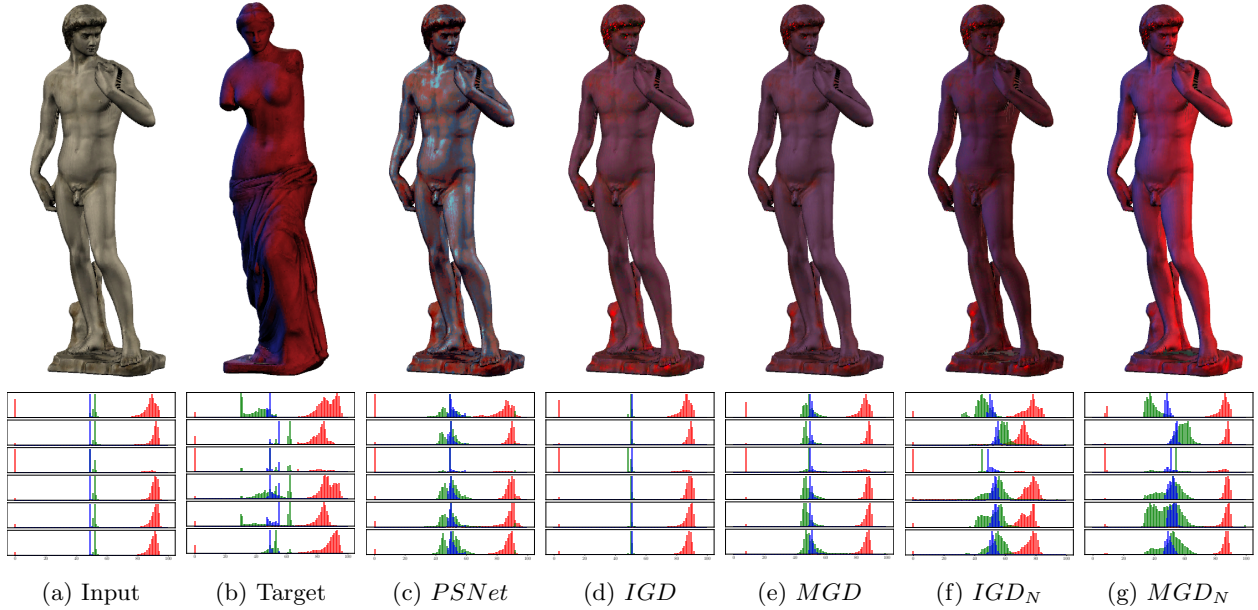


Figure 11: Point clouds color transfer evaluation. The second row shows the $l\alpha\beta$ distributions of the point clouds for each of the 6 directions, from top to bottom: $-x, +x, -y, +y, -z, +z$. The red, green, and blue colors correspond to l, α , and β distributions respectively.

the same sizes of point clouds (between 10^3 and 10^5 points), *PSNet* takes approximately 15 seconds to compute the color transfer and up to 100 seconds in the worst case. However, we think that, the more different the point clouds (in terms of color), the higher the computation time. All the processing times are reported in Figure 12 and have been obtained using a computer with the following specification: Intel(R) Core(TM) i7-7920HQ CPU 3.10GHz, processor x64, NVidia Quadro M2200 GPU, 32Go RAM.

4.2.2 Global color distributions similarity score

To compute the global similarity score between the output and the target color distributions, we use the metrics defined for image color transfer. As an example, after performing their style-aware color transfer, Hristova *et al.* [10] computed the Bhattacharya coefficient to measure the similarity between the target and the output color distributions. To evaluate

the color transfer in terms of color and luminance, they calculate the score for each component of the *CIELAB* color space. The final similarity score is obtained by averaging the similarity scores of the luminance channel and the two chroma channels.

Besides, several works performed a color transfer based on the Wasserstein distance, between the color distributions of two images, and obtained convincing results [3, 20]. The Wasserstein distance is a solution to the earth mover’s distance (EMD) problem and is computed between the output and the target color distributions. We decided to evaluate both Bhattacharya and Wasserstein distances between output and target point clouds color distributions in the *CIELAB* color space. The final similarity score is the average of the three components scores. Both metrics give a similarity score between 0 and 1, where 1 means identical distributions (for clarity we take 1 - Wasserstein coefficient).

Overall, for 14 pairs of input and target point clouds we have evaluated, when comparing the color trans-

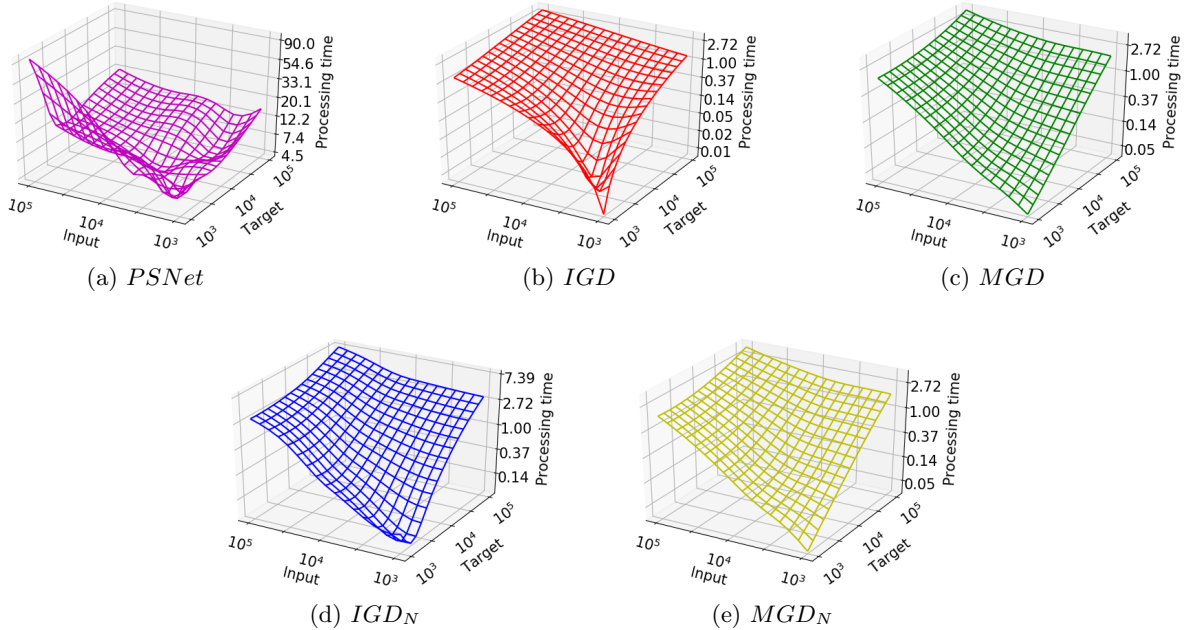


Figure 12: The processing time, first expressed in seconds, then represented in log-scale, for the five compared methods. Input and Target axes correspond to the number of points which varies between 10^3 and 10^5 points. *IGD*, *MGD*, and *MGD_N* methods takes between 0.03 and 3.5 seconds while *IGD_N* is slower (between 0.08 and 7.5 seconds). Our four methods have a linear complexity and the processing time only depends on the number of points: $\mathcal{O}(N + M)$. On the other hand, *PSNet* is much slower and takes approximately 15 seconds to produce the color transfer (about 100 seconds in the worst case), depending on the sizes and the colors of the point clouds.

fer produced by our methods against the others, our methods perform better for both Bhattacharya and Wasserstein distances as showed in Table 2. A Repeated Measures ANOVA has been performed and a significant difference ($p < 0.05$) between the scores obtained by the different methods has been found for both metrics. A posthoc test showed that the scores of *IGD_N* and *MGD_N* are significantly higher than those of *MGD* ($p < 0.05$) for the Bhattacharya metric. Regarding the Wasserstein metric, a posthoc test showed that the scores of *IGD_N* and *MGD_N* are significantly higher than those of *PSNet* and *IGD* ($p < 0.05$).

Nevertheless, those metrics suffer from a limitation when it comes to comparing color distributions of

point clouds as they globally consider all points of the clouds, without accounting for geometry. For example, regarding the color transfer presented in Figure 11, the score of *PSNet* is higher than the scores of our two methods for the Bhattacharya metric ($PSNet=0.943$, $IGD_N=0.930$ and $MGD_N=0.931$). While the visual quality of the resulting point clouds clearly shows that our two methods (Figures 11f and 11g) perform better than *PSNet* (Figure 11c), the color distribution of the target point cloud is globally closer to the color distribution of *PSNet* than the color distributions of our two methods. This first metric suffers from a lack of consideration for the geometry of the point clouds, resulting in scores that are not representative of the visual quality of the color

transfer.

To tackle this issue, we propose to calculate similarity scores between rendered images of the two point clouds when considering many viewpoints as explained in the following section. Proceeding this way, we ensure the score to be coherent with the color transfer as it depends on the point clouds geometry visualized from a particular point of view.

4.2.3 Per viewpoint color distributions similarity score

To improve the relevance of the comparison, we calculate several similarity scores depending on the viewpoint the point clouds are visualized from. The point clouds are rendered using splatting techniques with a point radius large enough to ensure no holes in the projection of the 3D models and the background is not considered in the color distribution. Furthermore, the images are rendered from viewpoints uniformly distributed on a sphere surrounding the point clouds. Following the suggestion of Alexiou *et al.* [1] in their work that exploits user interactivity to assess point clouds quality, the cameras are uniformly placed on the vertices of a geodesic sphere as illustrated in Figure 13. This placement uniformly considers all the point cloud areas when computing the average scores. We decided to render 642 views per point cloud to obtain a dense covering, where each rendered image is 720×720 pixels.

The Bhattacharya and Wasserstein scores are computed on the color distributions of each rendered view of the target and output point clouds. The resulting scores show that the color distributions of the two point clouds are similar when visualized from a particular viewpoint. Besides, we can display these

scores on a heat-map projected onto the sphere surrounding the point clouds as showed in Figure 14. Each point of the sphere corresponds to a viewpoint, the color of a point corresponds to the similarity score (Wasserstein in this example) where blue means not similar and red means similar. We see that, with our methods (Figures 14d and 14e), on average, the similarity scores are higher than those obtained with other methods.

Finally, a unique score is computed by averaging all the viewpoint-dependent similarity scores. This unique score corresponds to how the color distributions are similar when visualizing the point clouds from every viewpoint, on average. Once again, our methods obtain the best overall scores for both Bhattacharya and Wasserstein metrics, as shown in Table 3. A Repeated Measures ANOVA still shows a significant difference ($p < 0.05$) between the scores obtained with the different methods for both metrics. A posthoc test showed that the scores of IGD_N and MGD_N are significantly higher than those of $PSNet$ ($p < 0.05$), scores of IGD_N are significantly higher

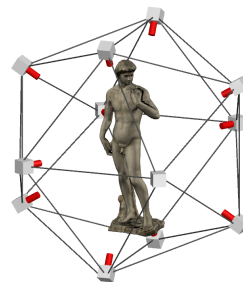
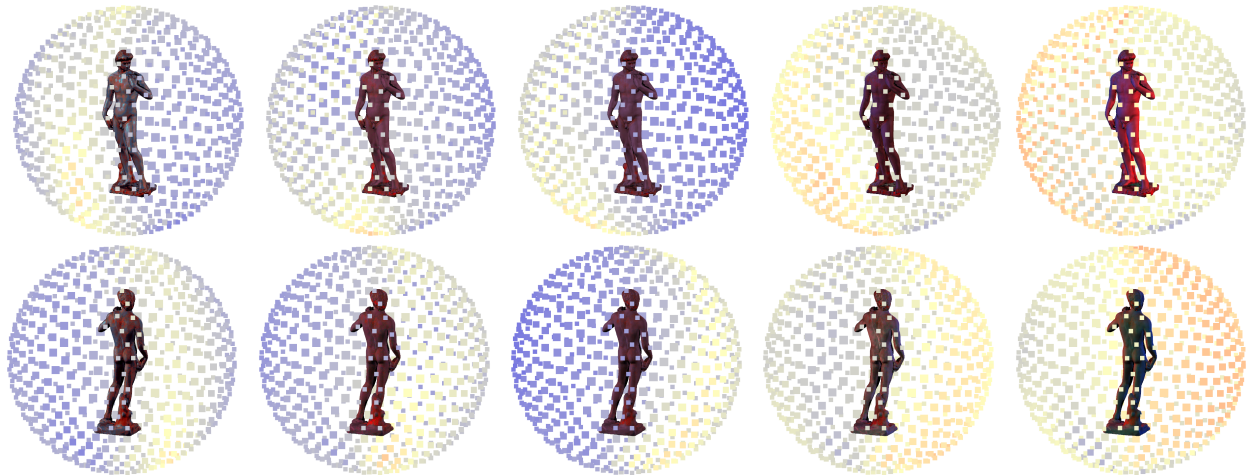


Figure 13: Cameras are uniformly placed on the geodesic sphere surrounding the point cloud.

	<i>PSNet</i>	<i>IGD</i>	<i>MGD</i>	<i>IGD_N</i>	<i>MGD_N</i>
Bhattacharya	0.84 ±0.136	0.87 ±0.103	0.86 ±0.105	0.89 ±0.080	0.90 ±0.076
1-Wasserstein	0.43 ±0.146	0.45 ±0.129	0.48 ±0.174	0.50 ±0.127	0.52 ±0.126

Table 2: Means and standard deviations of scores obtained with the 5 evaluated methods, for the metrics applied globally to all points of the point clouds, and for the color transfer of 14 different point clouds.



(a) *PSNet*
min=0.271;
max=0.561
mean=0.378;
std=0.053

(b) *IGD*
min=0.294;
max=0.596
mean=0.380;
std=0.055

(c) *MGD*
min=0.231;
max=0.553
mean=0.353;
std=0.081

(d) *IGD_N*
min=**0.361**;
max=0.608
mean=0.461;
std=0.053

(e) *MGD_N*
min=0.349;
max=**0.651**
mean=**0.499**;
std=0.063

Figure 14: Each point of the surrounding sphere corresponds to a viewpoint. The color of the point depends on the Wasserstein score in this example. Blue color means not similar (low score) and red color means similar (high score). The color distributions of our methods (14d and 14e) are globally more similar to the target than the other methods.

than those of *IGD*, and scores of *MGD_N* are significantly higher than those of *MGD* for the Bhattacharya metric. Regarding the Wasserstein metric, a posthoc test showed the scores of *MGD*, *IGD_N* and *MGD_N* to be significantly higher than those of *PSNet* ($p < 0.05$), and scores of *IGD_N* and *MGD_N* significantly higher than those of *IGD*.

This viewpoint-dependent metric is then more coherent than the score computed over the global color

distributions. Using the same example as previously (Figure 11), the Bhattacharya score of our two methods is now higher than the score of *PSNet* ($PSNet=0.874$, $IGD_N=0.894$ and $MGD_N=0.882$). In this case, the viewpoint-dependent metric is more representative of the color transfer quality than the global metric presented in Section 4.2.2.

Figure 15 shows an example where Bhattacharya and Wasserstein viewpoint dependent scores are repre-

	<i>PSNet</i>	<i>IGD</i>	<i>MGD</i>	<i>IGD_N</i>	<i>MGD_N</i>
Battacharya	0.81 ±0.147	0.84 ±0.115	0.84 ±0.112	0.87 ±0.096	0.88 ±0.081
1-Wasserstein	0.41 ±0.145	0.44 ±0.136	0.46 ±0.165	0.47 ±0.122	0.50 ±0.118

Table 3: Means and standard deviations of scores obtained by the 5 evaluated methods, for the averaged viewpoint-dependent metrics, and for the color transfer of 14 different point clouds.

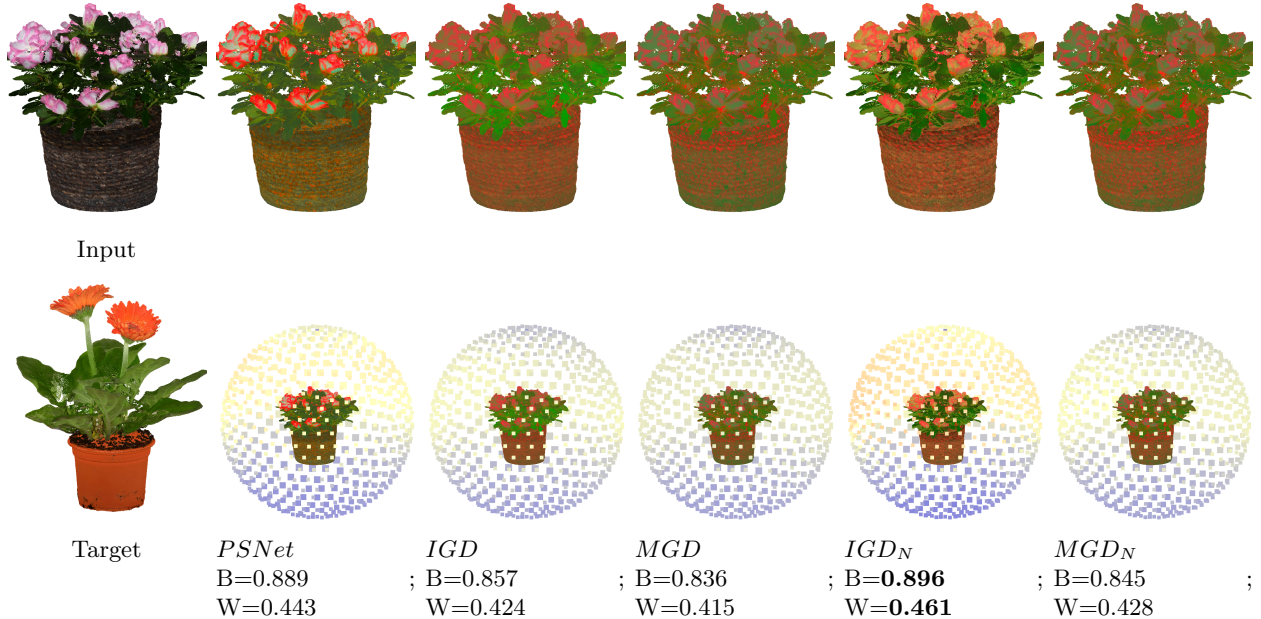


Figure 15: An example where MGD_N fails in providing a qualitative color transfer while IGD_N provides a faithful and coherent color transfer. The surrounding spheres are colored with the Wasserstein scores while the means of the Bhattacharya (B) and the Wasserstein (W) viewpoint-dependent scores are reported on the figure. Both scores are coherent with the visual quality of the results.

representative of the visual quality of the color transfer. Indeed, with MGD_N the orange color of the target flowers is not preserved and the variety of green colors are not rendered on the leaves. Contrarily, IGD_N provides a faithful and coherent color transfer as well as $PSNet$. The efficiency of $PSNet$ in this example may come from the clear separations in colors and positions, which helps to find meaning between all parts of the models (i.e. flowers, leaves, and pots).

4.2.4 Point-to-point distance with ground truth for relighting and delighting use case

We finally propose a third metric that can only be used in the case of relighting and delighting. Relighting is the process of applying the lighting condition of a target point cloud to an input one. On the opposite, a delighting process is a high trend for



Figure 16: Simulating the expected result of relighting.

a few years and consists of removing the lighting on a model.

In this scenario, the ground truth can be simulated by rendering the input point cloud with the lighting conditions of the target point cloud as illustrated in Figure 16. The output point cloud is then compared with

the simulation (i.e. the ground truth). As a result, each point of the input cloud is assigned two colors: one resulting from the color transfer and the other resulting from the relighting (the ground truth). We compute the PSNR and the cosine distance applied to the pairs of colors (output and ground truth). Note that the two scores (PSNR and cosine distance) are computed for each color component in *CIELAB*. The L , a , and b scores are averaged to get the final scores.

Two examples of relighting (first row) and delighting (second row) are presented in Figure 17. For those examples, the cosine distance is more representative of the visual quality of the color transfer than the PSNR. All the metrics presented in Sections 4.2 are valid for color transfer used for relighting or delighting applications.

To conclude, according to the objective metrics, we show that accounting for the point cloud geometry improves significantly the color transfer, and even more when considering correlations between colors and normals. In the next section, we present few other applications that can be tackled by our color transfer methods for point clouds.

4.3 Other applications

While, with the presented results below, the input and target point clouds have similar geometric structures, we proposed few other applications where the geometric structures are not necessarily similar between the two point clouds. The first ideas of applications are concerned with aging and weathering phenomena. Several techniques exist for those purposes [17]. Aging consists in gradually deteriorating a model depending on a period or on a chosen color style as illustrated in Figure 18 (MGD_N method has been used to produce these results). Here, we introduce the possibility of applying the color style to the output depending on a coefficient τ :

$$\mathbf{new}_{\mathbf{c}_i}^{\mathbf{O}} = \tau \times \mathbf{c}_i^{\mathbf{O}} + (1 - \tau) \times \mathbf{c}_i^{\mathbf{I}}, \quad (13)$$

where $\mathbf{new}_{\mathbf{c}_i}^{\mathbf{O}}$ is a linear combination between the input color $\mathbf{c}_i^{\mathbf{I}}$ and the output color $\mathbf{c}_i^{\mathbf{O}}$ of the point i , which replaces the output color in the final rendering. In this example, the statue is more or less aged

thanks to the target color style of a mossy stump.

On the other hand, weather-induced aging refers to embedding a model at a particular season time with specific environmental conditions. In the example presented in Figure 19, we embed an archaic house in a snowy environment defined by a simple snowed rock as a target color style. The strong point of our method is to transfer color depending on the direction of the normals. In the target point cloud, the normals oriented toward the up direction correspond to snow and are white. In the outputs, the surfaces oriented toward the up direction are then whiter, corresponding to snow that better holds (see the door trim in the yellow insets).

Finally, as it has been presented in Section 4.2.4, relighting and delighting are two applications of our color transfer for point clouds. Our method is very efficient for this task as lighting directly depends on normals, as showed in Figure 20. In this example, IGD_N does not succeed in transferring the red color on the right side of the statue while MGD_N produces a color transfer very close to the expected result.

Furthermore, our method allows delighting point clouds by taking a neutral point cloud as a target color style, as illustrated in Figure 21. Contrary to the previous example of relighting, IGD_N color transfer results in a uniform grey color on the entire statue while color variations are still perceptible on the MGD_N resulting point cloud.

5 Conclusion

Point clouds are becoming a commonly used data format for many applications, such as cultural heritage preservation, scanning of the surrounding environment, or the generation of 3D assets for video games and computer graphics movies, to name a very few. Improving the rendering quality of such point clouds is essential and has been studied for years now.

In this paper, we have presented a new and innovative pipeline for facilitating the transfer of point clouds color. The proposed methods, which rely on the color distribution as well as the geometry, are simple, unsupervised, and efficient. Our color transfers provide very good results in terms of visual quality.

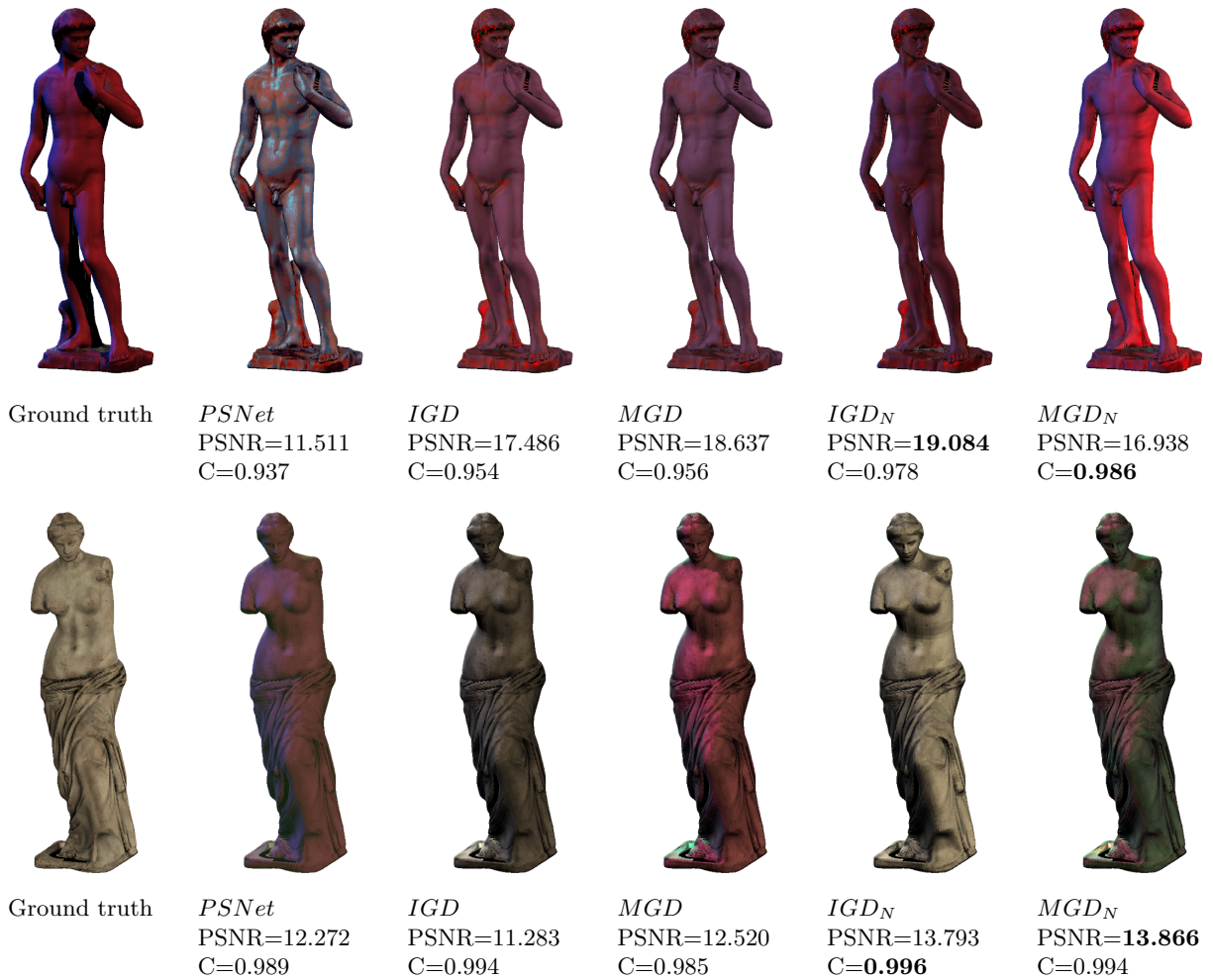


Figure 17: The scores of the point-to-point metric for relighting and delighting. The cosine distance (C) is coherent with the visual quality of the results while the PSNR is not very representative.

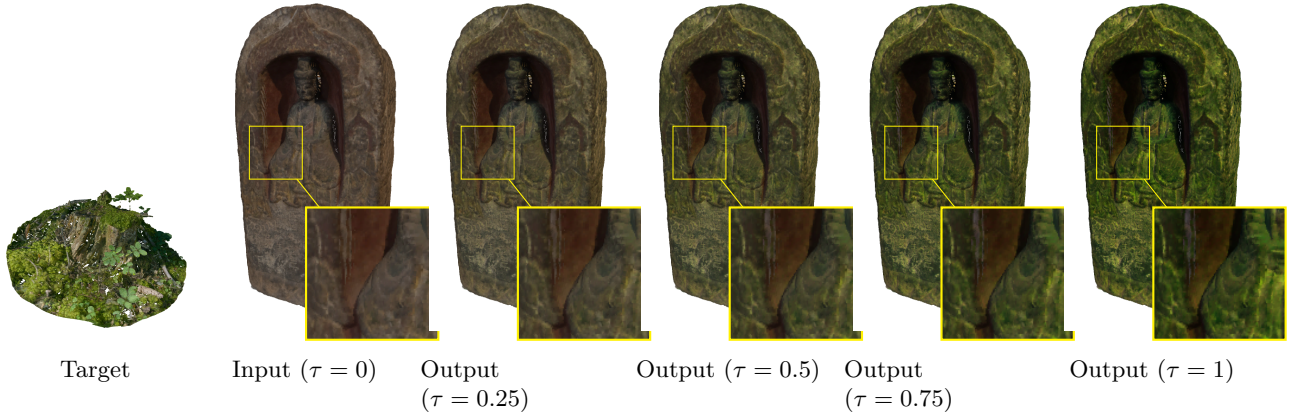


Figure 18: Gradually aging using MGD_N color transfer, with τ a coefficient of linear combination between input and output point clouds. The input point cloud gets more and more greenish on its exterior surfaces.

Furthermore, the proposed methods outperform the methods that do not account for the point clouds geometry as well as the *PSNet* method [5], in terms of visual quality and objective metrics. However, our methods still have some limitations. When different parts of the point clouds present color similarities as well as normals oriented toward the same direction, the color transfer may assign colors of the target point cloud to unexpected parts of the input point cloud as in Figures 10 and 15.

In future work, we aim to go further by considering new research avenues, such as more representative objective quality metrics (considering similarities between the geometry of the input and the target point clouds for example) and color distribution modeling adapted to 3D point clouds. The former is required to precisely determine the quality of the transfer whereas the latter could relax the Gaussian hypothesis of our method. We believe that a mixture of multivariate generalized Gaussian distribution could be a good candidate for further improving the results [11]. Beyond these two points, we also aim to investigate the potential of clustering-based methods as well as deep networks to automatically generate stylized point clouds by manipulating several visual features, such as color, gradient, texture, and so on.

Acknowledgments

All 3D assets have been uploaded from the Sketchfab library (<https://sketchfab.com>). This work has been supported by the ANR project ANR-17-CE23-0020. Thanks to editor and reviewers for their constructive commentaries.

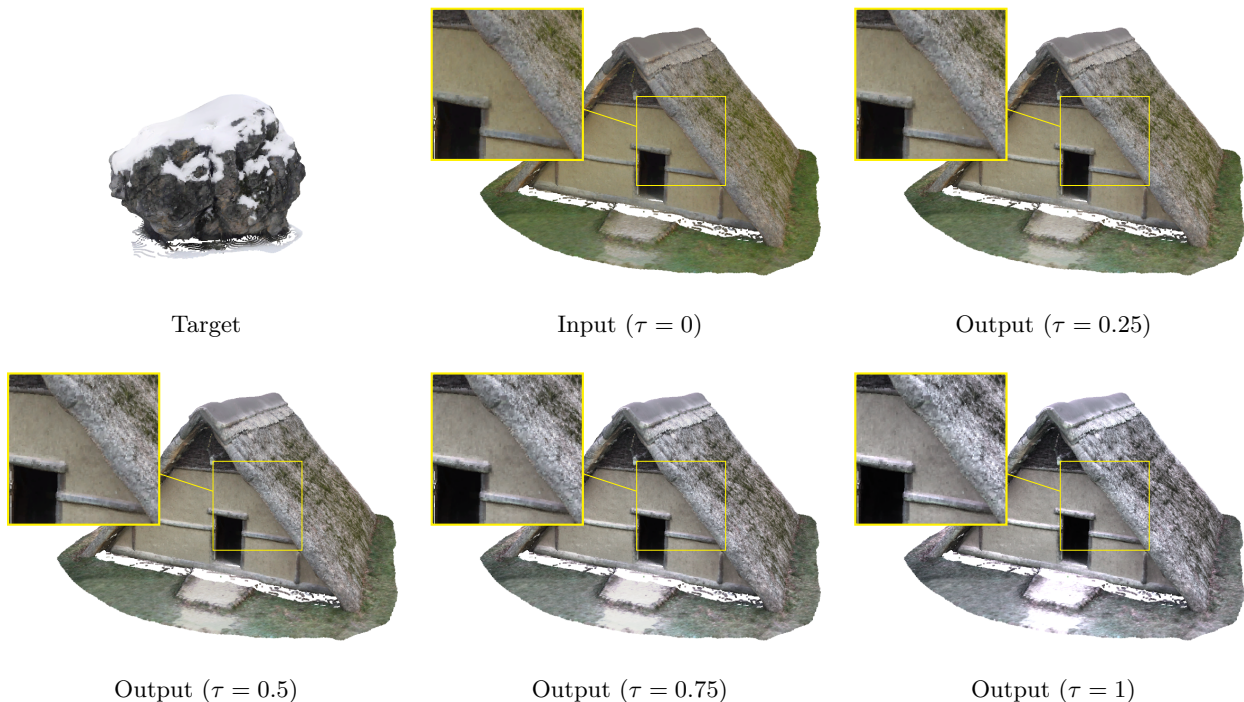


Figure 19: Gradually weather-induced aging using MGD_N color transfer, with τ a coefficient of linear combination between input and output point clouds. The input point cloud gets more and more snow-covered on its surfaces oriented toward the up direction.

References

- [1] Evangelos Alexiou and Touradj Ebrahimi. Exploiting user interactivity in quality assessment of point cloud imaging. In 2019 Eleventh International Conference on Quality of Multimedia Experience (QoMEX), pages 1–6. IEEE, 2019.
- [2] Alessandro Artusi, Francesco Banterle, Tunç Ozan Aydın, Daniele Panozzo, and Olga Sorkine-Hornung. Image content retargeting: maintaining color, tone, and spatial consistency. CRC Press, 2016.
- [3] Nicolas Bonneel, Gabriel Peyré, and Marco Cuturi. Wasserstein barycentric coordinates: histogram regression using optimal transport. ACM Trans. Graph., 35(4):71–1, 2016.
- [4] Xu Cao and Katashi Nagao. Point cloud colorization based on densely annotated 3d shape dataset. In International Conference on Multimedia Modeling, pages 436–446. Springer, 2019.
- [5] Xu Cao, Weimin Wang, Katashi Nagao, and Ryosuke Nakamura. Psnet: A style transfer network for point cloud stylization on geometry and color. In The IEEE Winter Conference on Applications of Computer Vision, pages 3337–3345, 2020.
- [6] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In Proceedings of the

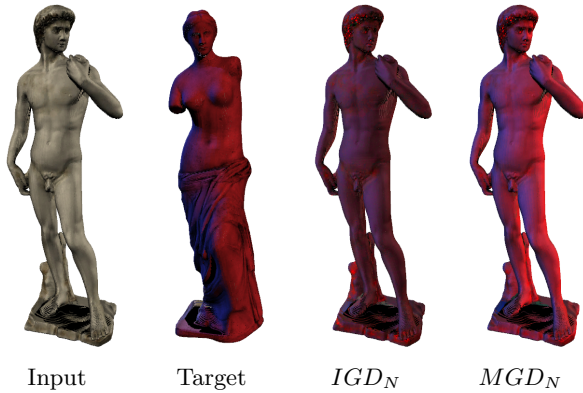


Figure 20: Point clouds color transfer used for re-lighting.

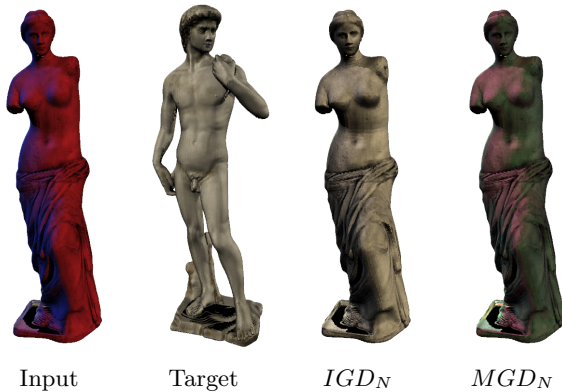


Figure 21: Point clouds color transfer used for de-lighting.

IEEE conference on computer vision and pattern recognition, pages 2414–2423, 2016.

- [7] Jeffrey P Grossman and William J Dally. Point sample rendering. In Rendering techniques' 98, pages 181–192. Springer, 1998.
- [8] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3d point clouds: A survey. arXiv preprint arXiv:1912.12033, 2019.
- [9] N. Hichri, Chiara Stefani, Livio De Luca, Philippe Veron, and Gael Hamon. From point

cloud to BIM: a survey of existing approaches. In XXIV International CIPA Symposium, strasbourg, France, 2013. Proceedings of the XXIV International CIPA Symposium.

- [10] Hristina Hristova, Olivier Le Meur, Rémi Cozot, and Kadi Bouatouch. Style-aware robust color transfer. In Computational Aesthetics, pages 67–77, 2015.
- [11] Hristina Hristova, Olivier Le Meur, Remi Cozot, and Kadi Bouatouch. Transformation of the multivariate generalized gaussian distribution for image editing. IEEE transactions on visualization and computer graphics, 24(10):2813–2826, 2017.
- [12] Hristina Hristova, Olivier Le Meur, Rémi Cozot, and Kadi Bouatouch. Transformation of the beta distribution for color transfer. In VISIGRAPP (GRAPP), 2018.
- [13] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In European conference on computer vision, pages 694–711. Springer, 2016.
- [14] Yongtae Jun. A piecewise hole filling algorithm in reverse engineering. Computer-aided design, 37(2):263–270, 2005.
- [15] Byungsoo Kim, Vinicius C. Azevedo, Markus Gross, and Barbara Solenthaler. Lagrangian Neural Style Transfer for Fluids. ACM Transactions on Graphics, 39(4), 2020.
- [16] Marc Levoy and Turner Whitted. The use of points as a display primitive. Citeseer, 1985.
- [17] Stéphane Mérillou and Djamchid Ghazanfar-pour. A survey of aging and weathering phenomena in computer graphics. Computers & Graphics, 32(2):159–174, 2008.
- [18] Tom Mertens, Jan Kautz, Jiawen Chen, Philippe Bekaert, and Frédo Durand. Texture transfer using geometry correlation. Rendering Techniques, 273(10.2312):273–284, 2006.

- [19] Carsten Moenning and Neil A Dodgson. Intrinsic point cloud simplification. Proc. 14th GrahCon, 14:23, 2004.
- [20] Youssef Mroueh. Wasserstein style transfer. arXiv preprint arXiv:1905.12828, 2019.
- [21] László Neumann and Attila Neumann. Color style transfer techniques using hue, lightness and saturation histogram matching. In Computational Aesthetics, pages 111–122. Cite-seer, 2005.
- [22] Massimiliano Pieraccini, Gabriele Guidi, and Carlo Atzeni. 3d digitizing of cultural heritage. Journal of Cultural Heritage, 2(1):63–70, 2001.
- [23] François Pitié and Anil Kokaram. The linear monge-kantorovitch linear colour mapping for example-based colour transfer. 2007.
- [24] Tania Pouli and Erik Reinhard. Progressive histogram reshaping for creative color transfer and tone reproduction. In Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering, pages 81–90, 2010.
- [25] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 652–660, 2017.
- [26] Yann Quinsat and Claire Lartigue. Filling holes in digitized point cloud using a morphing-based approach to preserve volume characteristics. The International Journal of Advanced Manufacturing Technology, 81(1-4):411–421, 2015.
- [27] Erik Reinhard, Michael Adhikhmin, Bruce Gooch, and Peter Shirley. Color transfer between images. IEEE Computer graphics and applications, 21(5):34–41, 2001.
- [28] Paul Rosenthal and Lars Linsen. Image-space point cloud rendering. In Proceedings of Computer Graphics International, pages 136–143, 2008.
- [29] Daniel L Ruderman, Thomas W Cronin, and Chuan-Chin Chiao. Statistics of cone responses to natural images: implications for visual coding. JOSA A, 15(8):2036–2045, 1998.
- [30] Manuele Sabbadin, Gianpaolo Palma, Francesco Banterle, Tamy Boubekeur, and Paolo Cignoni. High dynamic range point clouds for real-time relighting. In Computer Graphics Forum, volume 38, pages 513–525. Wiley Online Library, 2019.
- [31] Markus Schütz and Michael Wimmer. High-quality point-based rendering using fast single-pass interpolation. In 2015 Digital Heritage, volume 1, pages 369–372. IEEE, 2015.
- [32] Ondřej Texler, David Futschik, Michal Kučera, Ondřej Jamříška, Šárka Sochorová, Menglei Chai, Sergey Tulyakov, and Daniel Šykora. Interactive video stylization using few-shot patch-based training. arXiv preprint arXiv:2004.14489, 2020.
- [33] Chao Wang, Yong K Cho, and Changwan Kim. Automatic bim component extraction from point clouds of existing buildings for sustainability applications. Automation in Construction, 56:1–13, 2015.
- [34] Hui Xu, Minh X Nguyen, Xiaoru Yuan, and Baoquan Chen. Interactive silhouette rendering for point-based models. In Proceedings of the First Eurographics conference on Point-Based Graphics, pages 13–18. Eurographics Association, 2004.
- [35] Naci Yastikli. Documentation of cultural heritage using digital photogrammetry and laser scanning. Journal of Cultural Heritage, 8(4):423–427, 2007.