Vision, Modeling, and Visualization (2018)
F. Beck, C. Dachsbacher, and F. Sadlo (Eds.)

Interactive Interpolation of Metallic Effect Car Paints

Tim Golla and Reinhard Klein†

University of Bonn

(a) *Measured metallic car paints* **(b)** *Measured and interpolated metallic car paint materials*

Figure 1: *(a) Two measured metallic car paints. (b) Measured car paints and car paints generated with our approach: Interpolation of the brown paint towards the blue paint in 25% steps. All parameters are interpolated, including the metallic sparkling effect caused by metallic flakes.*

Abstract

Metallic car paints are visually complex materials that, among others effects, exhibit a view-dependent metallic sparkling, which is particularly difficult to recreate in computer graphics. While capturing real-world metallic paints is possible with specialized devices, creating these materials computationally poses a difficult problem. We present a method that allows for interactive interpolation between measured metallic automotive paints, which can be used to generate new realistic-looking metallic paint materials. By clustering the color information present in the measured bidirectional texture function (BTF) responsible for the metallic sparkling effect, we set up an optimal transport problem between metallic paints' appearances. The design of the problem facilitates efficiently finding a solution, based on which we generate a representation that allows for real-time generation of interpolated realistic materials. Interpolation happens smoothly, no flickering or other visual artifacts can be observed. The developed approach also enables to separately interpolate the larger-scale reflective properties, including the basic color hue, the local color hue, and the sparkling intensity of the metallic paint. Our method can be used intuitively in order to generate automotive paints with a novel appearance and explore the space of possible metallic paints spanned by given real-world measurements. The resulting materials are also well suited for real-time rendering in standard engines.

CCS Concepts

• ***Computing methodologies*** → ***Computer graphics; Rendering; Reflectance modeling;***

1. Introduction

Metallic paints are complex materials that are widely used in the car industry. They exhibit a sparkling effect that is caused by small effect particles – usually aluminium flakes – which is difficult to render in a convincing way. Metallic automotive paints convey a premium look and thus are the most popular kind of paints used for new cars. Realistic rendering of metallic paints in real-time is of great importance for the car industry. Typical applications are in the field of marketing, where real-time applications for customers are becoming more and more common, for example on the car manufacturer's websites as well as in the design area. Further fields are the game industry and even the movie industry where real-time rendering is used for quick prototypical viewing of virtual scenes.

† {golla,rk}@cs.uni-bonn.de

DOI: 10.2312/vmv.20181248

For designers not only realistic rendering but also the ability to generate new variants of the paints is very important. This imposes tremendous technical challenges. Finding the desired paint should be intuitive and efficient, which requires interactive editing of the appearance, since otherwise the exploration of the space of possible paints is very difficult.

Current models for car paints are either fully analytical, partially, or completely data-driven. Analytical models provide great freedom with the ability to generate more or less any kind of paint. They however have drawbacks in terms of usability: It is often tedious for non-expert users to generate new automotive paints' appearances because of the large number of parameters which are not intuitively usable. Testing parameters to realize the diversity of appearances is usually time-consuming. Furthermore, the high quality models are computationally heavy and thus not suited for real-time applications. Fully or partially data-driven models are easier to use, because real-world materials can be measured. They are often also computationally simpler than analytical models, making them better suited for real-time applications. It is however much harder to generate new materials from given measured real-world samples and specialized synthesis methods have to be used.

In this paper, we present a novel method for interactively synthesizing new metallic paints by inter- and extrapolating measured materials. For this purpose we build on a recently developed real-time capable car paint model, which among other parameters exploits the statistics of the measured materials. We formulate the correspondence computation between two of these statistical representations as an optimal transport problem, which can be solved efficiently. Based on the resulting correspondence map, a fast interactive inter- and extrapolation of measured paints is realized. The key contributions of our work can be summarized as follows:

- We present a simple and fast method for solving the interpolation problem for measured car paints. In comparison to the related works of Bonneel et al. [BVDPPH11] and Solomon et al. [SDGP*15], our method is easier to implement and handles a measured bidirectional texture function (BTF) representation.
- We show how to separately interpolate the larger-scale reflective properties, including the basic color hue, the local color hue and the sparkling intensity of the metallic paints' appearances in order to allow greater artistic freedom.
- We describe a representation suited for real-time editing and rendering of the metallic paints.

2. Related Work

We focus on the related work most closely related to this paper. For a more general overview of material acquisition and rendering, we refer the reader to the literature like the SIGGRAPH 2017 course *Material Capture and Representation with Applications in VR* by Guarnera et al. [GGH*17], the state of the art report by Guarnera et al. [GGG*16] and the textbook *Digital modeling of material appearance* by Dorsey et al [DRS10].

2.1. Metallic Car Paint Rendering

This subsection largely follows the overview given by Golla and Klein [GK17]. In 1990, first research on measurement and rendering of car paints were done by Takagi et al. [TTOO90]. Takagi continued to work in this field and published further research later [TWB05]. Further pioneers in this area were Dumont-Bècle et al. [DBFK*01], who presented a multi-texture approach. Ershov et al. [EKK99] presented a good-looking physically-based, analytical model. They further improved this in their later publications [EKM01, EĎKM04]. A disadvantage of their method is the large amount of parameters, which are difficult to select for non-expert users. A detailed introduction to the physics behind the appearance of metallic paints was given by Kitaguchi [Kit08].

In the approach of Ďurikovič and Martens [ĎM03] the geometry of metallic flakes is modeled explicitly in order to simulate their sparkling. This however leads to a huge number of polygons, which is not suitable for many applications, especially in the real-time context. Ngan et al. [NDM05] were able to show that the Cook-Torrance model [CT82] can represent the large-scale shininess of car paints well. A complete process from measuring to real-time rendering of car paints was presented by Günther et al. [GCG*05]. They rely on fitting analytical models to their measurements. To reproduce the sparkling, they used ideas from Ershov et al. [EKK99] and Ďurikovič and Martens [ĎM03] and procedurally generate a normal map which represents the flakes.

A combined model for metallic and pearlescent paints was presented by Rump et al. [RMS*08]. Later the AxF car paint model [ML15] was developed based on their research. They use a measured BTF for the metallic flakes, in contrast to the otherwise similar model by Günther et al. [GCG*05], where a procedural technique was used. Rump et al. also experimented with PCA-based compression for the BTF part. They achieved a moderate compression ratio of 1:4 on their data. In their follow-up paper [RSK09], they described a compression algorithm based on selecting representative image patches for the flake BTF. Depending on the dataset, they report compression rates of 1:18 to 1:46 on their data, The AxF format builds on this compressed representation. Golla and Klein [GK17] based their model on the AxF car paint model, but replaced the image-based BTF by a statistical representation based on the measurements. This way, they are able to achieve very good compression ratios.

Another approach was presented by Ďurikovič and Mihálik [ĎM13], who used an 8 bit texture to generate the sparkle effect. While yielding good results, the BTF approach seems to deliver a higher quality. A new BRDF model surpassing the Cook-Torrance model for car paints was introduced by Kurt et al. [KSKK10]. Another novel BRDF model for glossy surfaces was suggested by Löw et al. [LKYU12]. However, their models cannot account for the spatially varying metallic flakes.

Recently, high-quality simulation models for rendering of glints and metal surfaces were presented by Yan et al. [YHJ*14, YHMR16] and Jakob et al. [JHY*14]. These approaches deliver good results, but are computationally intensive and thus not ideal for real-time applications. Related is also the publication by Raymond et al. [RGB16], who render scratched metal surfaces, but no metallic paints. An approach for rendering metallic flakes was presented by Atanasov and Koylazov [AK16]. It is not real-time capable, however.

2.2. BTF Synthesis and Interpolation

The following publications all present methods for the synthesis of BTFs that look similar to a given – usually small – sample, but are parameterized on different – usually larger – surfaces. Tong et al. [TZL*02] described a method for BTF synthesis using so-called textons. They generate BTFs that look similar to samples from a database, but can be parameterized on arbitrary surfaces. Kawasaki et al. [KSOF05] presented a patch-based method for the same application. Meseth et al. [MMK03] presented a method for real-time rendering of BTFs, which relies on synthesis of similar-looking BTFs from small samples. Zhou et al. [ZDW*05] provided a method that synthesizes arbitrarily sized BTFs using a graph cut based algorithm. In a successive step, they add additional detail – imperfections, like scratches – to the BTF and generate seamless transitions. Haindl and Filip [HF04] described a probabilistic approach for BTF synthesis, which also provides good compression. Later, the same authors [HF07] presented a different method for strong compression and synthesis of BTFs, that look similar to the original data. Haindl et al. [HHCD05] presented a patch-based method for synthesizing BTFs. Liu et al. [LHZ*04] approximate a BTF sample by 4D point appearance functions. These, combined with 2D geometry maps can then be used for synthesis and real-time rendering. A good overview of BTF acquisition, synthesis and rendering was given by Müller et al [MMS*05].

Müller et al. [MSK07] presented a procedural method for editing BTFs. Their approach seems to be suited best for materials that exhibit a relatively coarse geometry like leather or corduroy. Kautz et al. [KBD07] presented an interactive approach for editing BTFs, which is well suited for materials like cloth, but not for the highly specular automotive paints. Ruiters et al. [RSK13] described a method for interpolating BTFs of different materials. Although their results look good, they report runtimes in the range of hours and require some manual markups. The major problem of these approaches is the correct interpolation of textures, which is hard to solve.

2.3. Optimal Transport in Computer Graphics

Optimal Transport problems and algorithms solving them have widespread use in many fields. We will only focus on applications in computer graphics. One of the best-known applications of optimal transport is the Wasserstein metric, also known as Earth Mover's Distance. Some of the best-known research in this context was published by Rubner et al. [RTG98, RTG00, RT01], who showed its usefulness for measuring the similarity of images. The earth mover's distance is frequently used in the computer vision area, as shown by Levina and Bickel [LB01], Ren et al. [RYZ11], Ling and Okada [LO07], Graumann and Darrell [GD04] and Pele and Werman [PW09].

Another widespread use of optimal transport, which is also closely related to our method, is color transfer for 2D images. Examples for this are the publications by Pitié et al. [PKD07], Rabin and Peyré [RP11], Ferradans et al. [FPPA14], Rabin et al. [RFP14], Frigo et al. [FSDH14] and Chizat et al. [CPSV16]. Rabin et al. [RPDB11] showed an application of optimal transport for 2D texture interpolation. Most closely related to our work are the publications by Bonneel et al. [BVDPPH11] and Solomon et al. [SDGP*15], who both employed optimal transport for BRDF interpolation. Bonneel et al. [BVDPPH11] described a method of mass transportation for BRDF interpolation. Solomon et al. [SDGP*15] employed convolutional Wasserstein distances for faster convergence and showed that their framework can also be used for BRDF interpolation.

3. The Statistical Car Paint Model

Our representation is based on the statistical car paint model presented by Golla and Klein [GK17], which itself is based on the car paint model defined in the AxF file format [ML15], which we will explain first.

3.1. The Basic Car Paint Model

The basic car paint model we use, which is also used in the AxF format [ML15], consists of a combination of multiple models. The model for a specific paint can be obtained by measurements, where the analytical parts are fitted to the measurements. The model consists of these components:

- A clear cloat layer, that changes incoming and outgoing directions $\mathbf{i}, \mathbf{o}$ to $\bar{\mathbf{i}}, \bar{\mathbf{o}}$, depending on the thickness and refractive index of this layer.
- A Lambertian BRDF $\frac{a}{\pi}$
- A multi-lobe Cook-Torrance BRDF [CT82] for the brightness, where the k-th lobe is defined as:

$$f^{CT}_{s_k,\alpha_k,F_{0,k}}\left(\bar{\mathbf{i}},\bar{\mathbf{o}}\right) = \frac{s_k}{\pi}\frac{D_{\alpha_k}(\bar{\mathbf{h}})F_{F_{0,k}}\left(\bar{\mathbf{h}},\bar{\mathbf{o}}\right)G\left(\bar{\mathbf{i}},\bar{\mathbf{o}}\right)}{\bar{\mathbf{i}}_z\,\bar{\mathbf{o}}_z}, \tag{1}$$

where $\bar{\mathbf{h}}$ is the half vector, s_k is the specular coefficient,

$$D_{\alpha_k}(\bar{\mathbf{h}}) = \frac{1}{\alpha_k^2\bar{\mathbf{h}}_z^4}e^{\frac{\bar{\mathbf{h}}_x^2-1}{\bar{\mathbf{h}}_z^2\alpha_k^2}} \tag{2}$$

is the microfacet distribution,

$$F_{F_{0,k}}(\bar{\mathbf{h}},\bar{\mathbf{o}}) = F_{0,k} + (1-F_{0,k})(1-\bar{\mathbf{h}}\cdot\bar{\mathbf{o}})^5, \tag{3}$$

where

$$F_{0,k} = \left(\frac{n_{1,k}-n_{2,k}}{n_{1,k}+n_{2,k}}\right)^2 \tag{4}$$

is Schlick's approximation [Sch94] of the Fresnel term, $n_{1,k}, n_{2,k}$ are the refractive indices and

$$G(\bar{\mathbf{i}},\bar{\mathbf{o}}) = \min\left(1, \frac{2\bar{\mathbf{h}}_z\bar{\mathbf{o}}_z}{\bar{\mathbf{h}}\cdot\bar{\mathbf{o}}}, \frac{2\bar{\mathbf{h}}_z\bar{\mathbf{i}}_z}{\bar{\mathbf{h}}\cdot\bar{\mathbf{o}}}\right) \tag{5}$$

is the geometry term, where

$$\mathbf{y}_z = \mathbf{y}\cdot\mathbf{n}, \mathbf{y} \in \{\bar{\mathbf{h}}, \bar{\mathbf{i}}, \bar{\mathbf{o}}\} \tag{6}$$

denotes the dot product of $\mathbf{y}$ with the surface normal.
- A 2D color table $\chi(\theta_{\bar{\mathbf{h}}}, \theta_{\bar{\mathbf{i}}})$, that is used to represent large-scale color shifts, observed in pearlescent paints. It is parametrized by the angles $\theta_{\bar{\mathbf{h}}}$ and $\theta_{\bar{\mathbf{i}}}$, where $\theta_{\bar{\mathbf{h}}} = \arccos(\bar{\mathbf{h}}_z)$ is the angle between half vector and normal and $\theta_{\bar{\mathbf{i}}} = \arccos(\bar{\mathbf{h}}\cdot\bar{\mathbf{i}})$ is the angle between half vector and incoming direction.

- A BTF representing the sparkling effects caused by the metallic flakes. It is parameterized by $\theta_{\bar{\mathbf{h}}}$, $\theta_{\bar{\mathbf{i}}}$ and $\mathbf{x} \in \mathbb{R}^2$, the position on the surface, i.e. it is a 4D table, denoted as $\Xi(\mathbf{x}, \theta_{\bar{\mathbf{h}}}, \theta_{\bar{\mathbf{i}}})$. An angular sampling of 24-30 samples along each direction is usually chosen. This is according to research by Rump et al. [RMS*08], who observed that the angular lifetime of a metallic flake is around 6-7 degrees. Each combination of $\theta_{\bar{\mathbf{h}}}$ and $\theta_{\bar{\mathbf{i}}}$ results in a 2D texture. For clarity, we will call this function *flake BTF* in the following.

The complete model is:

$$f(\mathbf{x}, \bar{\mathbf{i}}, \bar{\mathbf{o}}) = \chi(\theta_{\bar{\mathbf{h}}}, \theta_{\bar{\mathbf{i}}}) \left(\frac{a}{\pi} + \sum_{k=1}^{K} f^{CT}_{s_k, \alpha_k, F_{0,k}} \left(\bar{\mathbf{i}}, \bar{\mathbf{o}} \right) \right) + \Xi(\mathbf{x}, \theta_{\bar{\mathbf{h}}}, \theta_{\bar{\mathbf{i}}}) \quad (7)$$

According to the literature [ML15, GCG*05], good results can be achieved with three lobes.

3.2. The Statistical Model

Golla and Klein [GK17] replaced the memory-intensive flake BTF with a statistical representation for compression purposes. We will explain their flake model in the following. They keep the basic discrete parametrization of the flake BTF by angles $\theta_{\bar{\mathbf{h}}}$ and $\theta_{\bar{\mathbf{i}}}$. Each discretized angle combination yields a 2D texture, which we will call *flake slice*. Their approach is based on representing each slice by a number of cuboid-shaped *color clusters*. Within each cluster, they assume a uniform distribution of colors. Each of these clusters has a probability p_i, where $\sum p_i = 1$. In order to compute these color clusters, they employ an algorithm based on reducing an octree in color space to a pre-defined number of leave nodes n. The latter is the only parameter in their method, and thus in ours. They suggest using $n = 50$, which we adopt. In order to render metallic car paints in real-time from this representation, they provide a GPU-friendly representation. Each flake slice is represented by one row of two textures. They represent the color clusters by storing two opposing corners of the cluster cuboid of each cluster in a common texture. For representing the probabilities p_i, they compute the discrete cumulative distribution function F and store its kth value in the kth pixel of a single channel texture row, i.e. the texture row's kth pixel value equals $\sum_{i=1}^{k} p_i$. They do this to obtain the best possible compression ratio. This however forces their algorithm to read the complete texture row for each reconstruction. Since our focus is on speed, we slightly modify their approach by storing F^{-1}. This forces us to use a different discretization. Instead of the original 50 discretization steps for F, we use 500 steps for F^{-1}. The overall texture size remains comparatively small, such that this does not pose an issue memory-wise. For real-time reconstruction, random numbers are required. In order to ensure frame-to-frame coherence they generate a texture of pseudo-random numbers.

Real-time reconstruction is then performed on the GPU in the fragment/pixel shader. First, the discrete angle combination $\theta_{\bar{\mathbf{h}}}$ and $\theta_{\bar{\mathbf{i}}}$ is computed for the current pixel to be shaded. The random value texture is read for the current u, v position, which yields a random value. Different from Golla and Klein's approach, the random value is identical to the u coordinate in the inverse CDF texture. The inverse CDF texture value is the index of the color cluster to be used. We read the appropriate cluster cuboid corner values from the respective texture. Like Golla and Klein, we then read additional random values from the random value texture by using fixed offsets. Using these values, we generate a color value in the color cluster.

4. Interpolation Method

For synthesis, we wish to compute inter- and extrapolations of measured metallic paints. For simplicity, we first consider two metallic paints, where we regard one paint as the source paint and the other as target paint. In order to be able to interpolate the flake BTF responsible for the metallic sparkling effect, we preprocess it, which is explained in the following. The other components of the model require no preprocessing.

4.1. Preprocessing for the Interpolation of the Flake BTFs

First, the statistical model according to Golla and Klein [GK17] is generated for both metallic flake BTFs. Based on these representations, we set up an optimal transport problem. This idea is related to the Wasserstein metric [Was69, Dob70], also known as Earth Mover's Distance [RTG00, RT01]. Note that all computations are done in the Lab color space, where distances and interpolations behave in a perceptually plausible way. The interpolation has to be performed for all flake BTF texture slices, i.e. separate textures of the two metallic paint BTFs. We assume an identical angle discretization for both BTFs. The interpolation is then performed on each matching pair of BTF texture slices of the two paints. In the following we will consider one flake BTF slice pair of a source paint and a target paint. The process is repeated analogously for each slice pair.

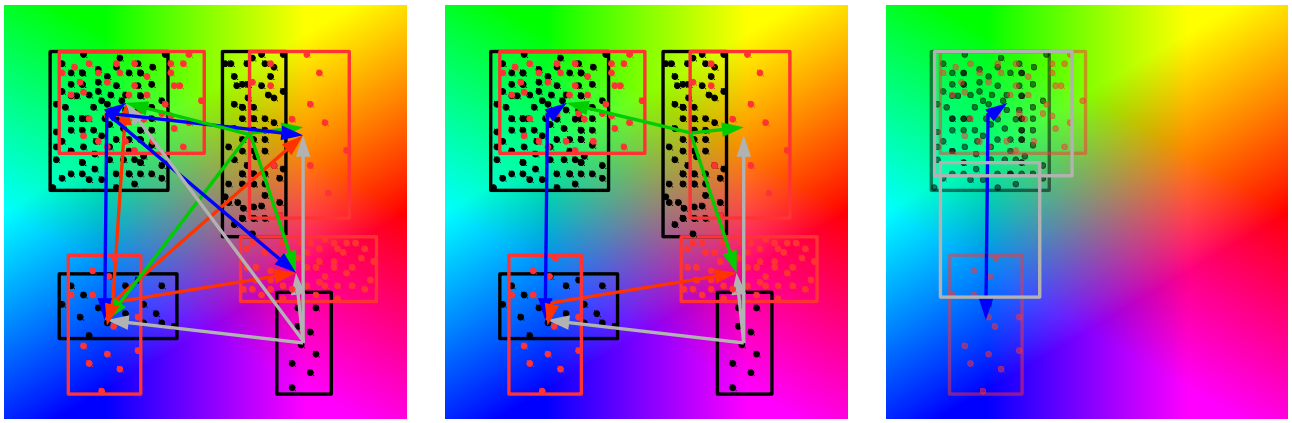
For each BTF slice, we compute the centers of the color cluster cuboids. We assign each cluster's probability to the respective center. The problem is now to transport the probabilities of the source BTF slice to those of the respective target BTF slice. The transportation cost is given by the Euclidean distance of the centers in the Lab color space. It can be visualized as a bipartite graph where each center of the source BTF slice is connected to each center of the target BTF slice – see Figure 2. The edge cost is the Euclidean distance. This can be formulated as a linear programming problem. The number of variables is equal to the number of edges, which is $n_s \cdot n_t$, where n_s, n_t is the number of color clusters of the source, respectively target BTF slice. Note that n_s does not have to be equal to n_t. Let $p_i, i \in \{1, \ldots, n_s\}$ be the probabilities of the source BTF slice color clusters, $q_j, j \in \{1, \ldots, n_t\}$ the probabilities of the target BTF slice color clusters. Let $\|\cdot\|$ be the Euclidean norm and $c_i, d_j, i \in \{1, \ldots, n_s\}, j \in \{1, \ldots, n_t\}$ be the source, respectively target, BTF slice color clusters' centers in the Lab color space. Let

$$w_{ij} = \|c_i - d_j\|, i \in \{1, \ldots, n_s\}, j \in \{1, \ldots, n_t\} \quad (8)$$

be the Euclidean distances of the color clusters' centers.

The optimization problem's objective function, which is to be minimized, is:

$$f(\boldsymbol{x}) = \sum_{i=1}^{n_s} \sum_{j=1}^{n_t} w_{ij} x_{ij}, \quad (9)$$



(a) *Schematic of the setup of the transportation transport problem. Black: Color clusters of the source car paint. Red: Color clusters of the target paint. Each cluster of the source material is connected to each cluster of the target material. The distances of the cluster centers in color space are the transportations costs. Not shown: Each cluster has a probability relative to the number of color data points it encompasses.*

(b) *A solution to the optimal transport problem. The arrows indicate that some of the probability of the cluster is transported to the respective target cluster.*

(c) *For each transport path used, a copy of the source color cluster's bounding box is generated – shown in gray. During real-time interpolation, the boxes are transported and transformed through the color space, until the respective target cluster's bounding boxes positions and shapes are reached at* 100% *interpolation. A* 50% *interpolation for only the top left cluster is shown.*

Figure 2: *Our interpolation method for metallic paints is based on solving an optimal transport problem. After clustering the colors present in the car paint BTF, we set up an optimal transport problem and use its solution for interpolating between the paints. Schematics of the involved steps are given in the subfigures.*

where $\boldsymbol{x} = (x_{ij}) \in \mathbb{R}^{n_s \cdot n_t}$ under the constraints:

$$x_{ij} >= 0 \; \forall i \in \{1,\ldots,n_s\} \; \forall j \in \{1,\ldots,n_t\} \tag{10}$$

$$\sum_{i=1}^{n_s} x_{ij} = q_j \; \forall j \in \{1,\ldots,n_t\} \tag{11}$$

$$\sum_{j=1}^{n_t} x_{ij} = p_i \; \forall i \in \{1,\ldots,n_s\} \tag{12}$$

The constraints also imply that $x_{ij} \in [0,1]$ and

$$\sum_{i=1}^{n_s} \sum_{j=1}^{n_t} x_{ij} = 1. \tag{13}$$

These properties implicate that the x_{ij} can be used as a probability measure for some countable collection $\{E_{ij}\}$. The solution can be computed by one of the existing solvers for linear programming problems – we used the Cbc solver from the COIN-OR project [LH03].

Using this information, we generate a new representation which allows interpolation between the two original BTF slices. We will now explain how to generate the collection $\{E_{ij}\}$. Let $C_i, i \in \{1,\ldots,n_s\}$, be the source and $D_j, j \in \{1,\ldots,n_t\}$, the target BTF slice color clusters. For each $x_{ij} > 0$, we create copies of the source BTF slice's ith color cluster, as well as of the target BTF slice's jth color cluster, i.e. $E_{ij} = (C_i, D_j)$. The tuple (x_{ij}, E_{ij}) then represents an *interpolatable color cluster* of our new *interpolatable* material. In theory, we could also generate the tuples (x_{ij}, E_{ij}) for $x_{ij} = 0$, but they would never be used for rendering and thus be useless. The number n_u of non-zero x_{ij} is at most $n_s + n_t - 1$ [Flo53, BVDPPH11].

4.2. Real-Time Interpolation

The coefficients of the analytical parts of the original paints can be inter- and extrapolated in a straightforward way. For the inter- and extrapolation of the diffuse color lookup table, we convert its entries to the Lab color space and interpolate linearly in this space.

To generate a *realization* of the interpolatable material of two metallic paints, we choose interpolation values $\alpha, \beta \in [0,1], \alpha + \beta = 1$. Besides the interpolation of the analytical parameters, we have to compute the flake BTF for this realization. Let us consider one interpolatable color cluster $(x_{ij}, E_{ij}) = (x_{ij}, (C_i, D_j))$. The probability x_{ij} was already computed appropriately in the preprocessing step to be valid for all realizations. We only need to compute an interpolated color cluster. For this, we consider two opposing corners g_i and g_i' of the cuboid representing C_i and the matching opposing corners h_j and h_j' of D_j. Without loss of generality, g_i and h_j are chosen with the smallest possible coordinates and g_i' and h_j' with the largest possible coordinates. The respective corners l_{ij}, l_{ij}' of the realization of the interpolatable color cluster are then computed as

$$l_{ij} = \alpha g_i + \beta h_j \tag{14}$$

and

$$l'_{ij} = \alpha g'_i + \beta h'_j. \tag{15}$$

This can be computed in real-time, on the CPU or the GPU, e.g. in a compute shader. In both cases, only the texture representing the cluster corners has to be updated, while the other properties of the flake BTF remain constant. The analytical parameters and the color lookup table have to be updated as well, which also require only a small amount of memory. Thus the necessary bandwidth is relatively low. Computing on the GPU is faster and has only the disadvantage of using slightly more memory, because all parameters have to be kept in the video RAM.

By dropping the requirement $\alpha, \beta \in [0,1]$, extrapolations can be generated. Dropping the requirement $\alpha + \beta = 1$ is also possible, however the results are less intuitive.

4.3. Interpolation of Multiple Materials

The approach can be generalized to an arbitrary number of metallic paints. While it would be possible to set up a transportation problem for many paints, this is not recommended, because the number of variables in the linear programming problem strongly increases. Let n_s, n_t, n_o be the number of color clusters of matching flake BTF slices of the three metallic paints and $F_l, l \in \{1 \dots n_o\}$ the color clusters of the third paint's BTF slice. The number of variables in the objective function would be $n_s \cdot n_t \cdot n_o$. We therefore choose a different approach: We iteratively solve transportation problems. We start with two original materials and generate their resulting interpolatable material. From this material's realization with $\alpha = \beta = 0.5$, we compute the interpolatable material with a new paint. Let us denote the first interpolatable material's probabilities $x_{ij} \neq 0$ by $x_k, k \in \{1, \dots, n_u\}$ and $k \leftrightarrow (i, j)$ for the respective i, j. As mentioned, in practice $n_u << n_s \cdot n_t$. Let $r_l, l \in \{1 \dots n_o\}$ be the probabilities of the third paint. The second optimization problem can then be set up analogously to the first with variables $y_{kl}, k \in \{1, \dots, n_u\}, l \in \{1 \dots n_o\}$. The new interpolatable material will then have color clusters that are represented as tuples $(y_{kl}, (C_i, D_j, F_l)), k \leftrightarrow (i, j)$. A realization will be computed as a linear combination of the three clusters and the analytical parameters with interpolation parameters $\alpha, \beta, \gamma \in [0,1], \alpha + \beta + \gamma = 1$. Again, extrapolations are possible. This process can be repeated with every additional paint.

Note that the number of color clusters and thus the memory usage typically increases with each additional paint added to the material. For practical applications however, we consider three to five paints being the maximum number of paints being intuitively usable. Since the statistical model and thus the interpolatable version are very memory efficient, this does not pose a problem in practice.

While solving the transportation problem is computationally somewhat intensive, it has to be done only once for each additional new paint and thus is considered a preprocessing step. In our implementation, the whole process of loading the original paints and computing the interpolatable material typically took around 50 seconds. The actual inter- and extrapolation, i.e. material synthesis can be done in real-time.

Figure 3: *Manipulating the metallic sparkling intensity by interpolation of only the flake BTF's lightness. From left to right: Original blue paint, blue paint with flake BTF lightness interpolated halfway between blue and brown paint, blue paint with flake lightness set to match the brown paint's flake BTF lightness, original brown paint. The basic color impression of the interpolated material remains blue, while the metallic sparkling intensity increases, matching that of the brown paint.*

4.4. Separate Interpolation of the Flake Intensity and the Color

By introducing smaller modifications to the method, we can create further useful applications. The first is to only interpolate the lightness color channel of the flake BTF, but leaving the hue channels as they are and also leaving all other parameters at those of the source paint. This makes the flakes sparkling intensity look more like that of the target paint, while keeping the color hue of the source paint. See Figure 3 for an example. The "dual" operation is also possible: One can keep the flake BTF's lightness information while interpolating its hue channels and the other paint parameters. This results in paints with a new color impression while keeping the flake sparkling intensity. See Figure 4 for an example. Note that some results can be achieved with both techniques. Another interesting experiment is to remove the color, i.e. a and b color channel information from the flake BTF. The result looks slightly less convincing as a metallic paint, because it has a very smooth look. See Figure 5. This also shows that the color (Lab ab) information in the flake BTF contributes to the overall look.

5. Results

All if our experiments were performed on a standard PC with an Intel Core i7-4930K CPU, 64 GB RAM and an Nvidia Geforce Titan first generation graphics card. Including reading and writing to disk, the computation of an interpolatable material from two paints took 50 seconds in our unoptimized Python implementation. Interpolation of three paints took 103 seconds.

Figure 6 shows several different interpolated materials. In each row, the paints in the left and right column are the original measured paints, paints in between are interpolated between them. The green-blue paint in the bottom picture is a flip-flop paint, which is also correctly interpolated. The flip-flop effect gets weaker the more one goes towards the gray paint.

We devised an interactive demo with three capsule models with

Figure 4: *Partial Interpolation. Left to right: Original brown paint, brown paint with all parameters except the flake BTF's lightness interpolated half-way between the brown and the blue paint's, brown paint with all parameters except the flake BTF's lightness interpolated to the blue paint's, original blue paint*

Figure 5: *Left: Original blue car paint material. Right: Blue paint with the flake BTF's color hue information set to white. This yields a blue material with white sparkles. However, it looks less convincing as a metallic paint, because it looks too smooth.*

original measured materials (blue, brown and gray), one capsule with a material that is interpolatable between the other three and a car with the same interpolatable material. The user can move the capsule with the interpolatable material and this material will be set according to the distance of this capsule to the other three capsules. That is, if the capsule with the interpolatable material is e.g. moved close to the blue capsule, it will be mostly blue. This demo is shown in Figure 7. This demo was running with 105 frames per second, when the materials remained constant. It dropped to 28 frames per second when the user started moving the capsules, i.e. the materials had to be interpolated. We are thus able to maintain real-time speeds even on slightly outdated hardware. The interpolation was performed on the CPU. We expect the frame rate to be even higher when the interpolation is performed on the GPU. The total VRAM requirement of this demo was 131 MB.

Figure 3 shows an example where only the flake BTF lightness is interpolated, but the paint's basic color remains that of the source paint. On the left, the original blue metallic paint is shown, which has flakes of only a low sparkling intensity. On the right, the original brown paint is shown, which has more intensely sparkling flakes. The second capsule from the left consists of a 50 % interpolation of the BTF lightness of the two original paints. Visually

Figure 6: *Several interpolations: In each row, the capsules in the left and right column are have been assigned the original measured paints, capsules have been assigned materials interpolated between the respective measured materials. The green-blue paint in the bottom picture is a flip-flop paint, which is also correctly interpolated. The flip-flop effect gets weaker the more one goes towards the gray paint.*

its sparkling intensity is between the original paints. The material of the third capsule from the left has the blue base color and also the bluish tint of the flakes, but the sparkling intensity of the brown paint. It looks like a version of the blue paint with more and brighter flakes.

Figure 4 shows the orthogonal operation: Starting with the brown paint on the right, we interpolate its base color and the flake BTF hue, but keep the flakes BTF lightness. The material of the second capsule from the right is a 50 % interpolation. It has a violet color, which is between blue and brown in the Lab color space. Its sparkling intensity remains that of the brown paint. The second paint from the left is the brown paint, with its hue fully interpolated to blue. Note that the result looks very similar to the third paint from the left in Figure 3, where the orthogonal operation, starting from the blue paint was performed. The leftmost paint is again the original blue metallic paint.

Figure 7: *Example Application: The capsules can be moved. The central capsule's material is interpolated from the other three capsules' materials, depending on the distance to them. The car has the same interpolated material. This way the user can generate a desired material in real-time. See also the accompanying video.*

When only small manipulations are desired, it is also possible to just manipulate one paint. Figure 5 shows the blue paint with the flake representation's color hue set to white. This yields a blue material with white sparkles. However, it looks less convincing as a metallic paint, because it looks too smooth.

Figure 8 shows materials generated by extrapolation. The measured brown and blue metallic paints are shown in Figure 8b. The result from extrapolating in 25% steps are shown in Figures 8a and 8c. The results are yellow-brown (8a) and intensely blue materials (8c).

(a) *Extrapolated* **(b)** *Measured* **(c)** *Extrapolated*

Figure 8: *Our approach can also be used to generate extrapolated versions of the metallic paints. The two paints displayed in the center (b) are the original paints. To the left (a) and right (b) are extrapolations in 25% steps. The respective interpolations are shown in Figure 1b.*

6. Conclusion

We presented a method that allows to generate realistic complex metallic car paint materials in real-time, based on interpolating captured real-world car paints. Our method requires no manual parameter tuning, only a short preprocessing step and is intuitive to use. We showed how to separately interpolate the larger-scale reflective properties, including the basic color hue, the local color hue, and the sparkling intensity of the metallic paints' appearances, allowing for even greater artistic freedom. Our method is simple to implement and efficient in terms of memory and computation requirements and thus only has moderate hardware requirements. Using our approach facilitates intuitively exploring the space of possible metallic paints spanned by given real-world measurements. We believe that our method will be useful for designers in the game, car and movie industry.

Acknowledgments

We would like to thank Volkswagen and X-Rite for providing measurements of the metallic paints. Uffizi Gallery Light Probe Image ©1999 Paul Debevec, http://www.debevec.org/Probes/

References

[AK16] ATANASOV A., KOYLAZOV V.: A practical stochastic algorithm for rendering mirror-like flakes. In *ACM SIGGRAPH 2016 Talks* (2016), ACM, p. 67. 2

[BVDPPH11] BONNEEL N., VAN DE PANNE M., PARIS S., HEIDRICH W.: Displacement interpolation using lagrangian mass transport. In *ACM Transactions on Graphics (TOG)* (2011), vol. 30, ACM, p. 158. 2, 3, 5

[CPSV16] CHIZAT L., PEYRÉ G., SCHMITZER B., VIALARD F.-X.: Scaling algorithms for unbalanced transport problems. *arXiv preprint arXiv:1607.05816* (2016). 3

[CT82] COOK R. L., TORRANCE K. E.: A reflectance model for computer graphics. *ACM Transactions on Graphics (TOG) 1*, 1 (1982), 7–24. 2, 3

[DBFK*01] DUMONT-BÈCLE P., FERLEY E., KEMENY A., MICHELIN S., ARQUÈS D.: Multi-texturing approach for paint appearance simulation on virtual vehicles. In *Proceedings of the driving simulation conference* (2001), pp. 123–133. 2

[ĎM03] ĎURIKOVIČ R., MARTENS W. L.: Simulation of sparkling and depth effect in paints. In *Proceedings of the 19th spring conference on Computer graphics* (2003), ACM, pp. 193–198. 2

[ĎM13] ĎURIKOVIČ R., MIHÁLIK A.: Metallic paint appearance measurement and rendering. *Journal of Applied Mathematics, Statistics and Informatics 9*, 2 (2013), 25–39. 2

[Dob70] DOBRUSHIN R. L.: Prescribing a System of Random Variables by Conditional Distributions. *Theory of Probability & Its Applications 15*, 3 (Jan. 1970), 458–486. URL: http://epubs.siam.org/doi/10.1137/1115049, doi:10.1137/1115049. 4

[DRS10] DORSEY J., RUSHMEIER H., SILLION F.: *Digital modeling of material appearance*. Morgan Kaufmann, 2010. 2

[EĎKM04] ERSHOV S., ĎURIKOVIČ R., KOLCHIN K., MYSZKOWSKI K.: Reverse engineering approach to appearance-based design of metallic and pearlescent paints. *The Visual Computer 20*, 8-9 (2004), 586–600. 2

[EKK99] ERSHOV S., KHODULEV A., KOLCHIN K.: Simulation of sparkles in metallic paints. In *Proceeding of Graphicon* (1999), pp. 121–128. 2

[EKM01] ERSHOV S., KOLCHIN K., MYSZKOWSKI K.: Rendering pearlescent appearance based on paint-composition modelling. In *Computer Graphics Forum* (2001), vol. 20, Wiley Online Library, pp. 227–238. 2

[Flo53] FLOOD M. M.: On the Hitchcock distribution problem. *Pacific Journal of Mathematics 3*, 2 (1953), 369–386. 5

[FPPA14] FERRADANS S., PAPADAKIS N., PEYRÉ G., AUJOL J.-F.: Regularized discrete optimal transport. *SIAM Journal on Imaging Sciences 7*, 3 (2014), 1853–1882. 3

[FSDH14] FRIGO O., SABATER N., DEMOULIN V., HELLIER P.: Optimal transportation for example-guided color transfer. In *Asian Conference on Computer Vision* (2014), Springer, pp. 655–670. 3

[GCG*05] GÜNTHER J., CHEN T., GOESELE M., WALD I., SEIDEL H.-P.: Efficient acquisition and realistic rendering of car paint. In *Vision, Modeling, and Visualization* (2005), vol. 5, pp. 487–494. 2, 4

[GD04] GRAUMAN K., DARRELL T.: Fast contour matching using approximate earth mover's distance. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on* (2004), vol. 1, IEEE, pp. I–I. 3

[GGG*16] GUARNERA D., GUARNERA G. C., GHOSH A., DENK C., GLENCROSS M.: BRDF representation and acquisition. In *Computer Graphics Forum* (2016), vol. 35, Wiley Online Library, pp. 625–650. 2

[GGH*17] GUARNERA G. C., GHOSH A., HALL I., GLENCROSS M., GUARNERA D.: Material capture and representation with applications in virtual reality. In *ACM SIGGRAPH 2017 Courses* (2017), ACM, p. 6. 2

[GK17] GOLLA T., KLEIN R.: An efficient statistical data representation for real-time rendering of metallic effect car paints. In *Virtual Reality and Augmented Reality: 14th EuroVR International Conference, EuroVR 2017* (2017), Springer International Publishing, pp. 51–68. 2, 3, 4

[HF04] HAINDL M., FILIP J.: A fast probabilistic bidirectional texture function model. *Image Analysis and Recognition* (2004), 298–305. 3

[HF07] HAINDL M., FILIP J.: Extreme compression and modeling of bidirectional texture function. *IEEE Transactions on Pattern Analysis and Machine Intelligence 29*, 10 (2007). 3

[HHCD05] HAINDL M., HATKA M., CHANTLER M., DRBOHLAV O.: BTF roller. In *Proceedings of the 4th International Workshop on Texture Analysis and Synthesis* (2005), pp. 89–94. 3

[JHY*14] JAKOB W., HAŠAN M., YAN L.-Q., LAWRENCE J., RAMAMOORTHI R., MARSCHNER S.: Discrete stochastic microfacet models. *ACM Transactions on Graphics (TOG) 33*, 4 (2014), 115. 2

[KBD07] KAUTZ J., BOULOS S., DURAND F.: Interactive editing and modeling of bidirectional texture functions. In *ACM Transactions on Graphics (TOG)* (2007), vol. 26, ACM, p. 53. 3

[Kit08] KITAGUCHI S.: *Modelling texture appearance of gonioapparent objects*. PhD thesis, University of Leeds, 2008. 2

[KSKK10] KURT M., SZIRMAY-KALOS L., KŘIVÁNEK J.: An anisotropic BRDF model for fitting and monte carlo rendering. *ACM SIGGRAPH Computer Graphics 44*, 1 (2010), 3. 2

[KSOF05] KAWASAKI H., SEO K.-D., OHSAWA Y., FURUKAWA R.: Patch-based BTF synthesis for real-time rendering. In *Image Processing, 2005. ICIP 2005. IEEE International Conference on* (2005), vol. 1, IEEE, pp. I–393. 3

[LB01] LEVINA E., BICKEL P.: The earth mover's distance is the mallows distance: Some insights from statistics. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on* (2001), vol. 2, IEEE, pp. 251–256. 3

[LH03] LOUGEE-HEIMER R.: The Common Optimization INterface for Operations Research: Promoting open-source software in the operations research community. *IBM Journal of Research and Development 47*, 1 (Jan. 2003), 57–66. doi:10.1147/rd.471.0057. 5

[LHZ*04] LIU X., HU Y., ZHANG J., TONG X., GUO B., SHUM H.-Y.: Synthesis and rendering of bidirectional texture functions on arbitrary surfaces. *IEEE transactions on visualization and computer graphics 10*, 3 (2004), 278–289. 3

[LKYU12] LÖW J., KRONANDER J., YNNERMAN A., UNGER J.: BRDF models for accurate and efficient rendering of glossy surfaces. *ACM Transactions on Graphics (TOG) 31*, 1 (2012), 9. 2

[LO07] LING H., OKADA K.: An efficient earth mover's distance algorithm for robust histogram comparison. *IEEE transactions on pattern analysis and machine intelligence 29*, 5 (2007), 840–853. 3

[ML15] MÜLLER G., LAMY F.: *AxF - Appearance exchange Format*. Tech. rep., X-Rite, Inc., 4300 44th St. SE, Grand Rapids, MI 49505, 2015. Version 1.0. 2, 3, 4

[MMK03] MESETH J., MÜLLER G., KLEIN R.: Preserving realism in real-time rendering of bidirectional texture functions. In *OpenSG Symposium* (2003), pp. 89–96. 3

[MMS*05] MÜLLER G., MESETH J., SATTLER M., SARLETTE R., KLEIN R.: Acquisition, synthesis, and rendering of bidirectional texture functions. In *Computer Graphics Forum* (2005), vol. 24, Wiley Online Library, pp. 83–109. 3

[MSK07] MÜLLER G., SARLETTE R., KLEIN R.: Procedural editing of bidirectional texture functions. In *Proceedings of the 18th Eurographics conference on Rendering Techniques* (2007), Eurographics Association, pp. 219–230. 3

[NDM05] NGAN A., DURAND F., MATUSIK W.: Experimental analysis of BRDF models. *Rendering Techniques 2005*, 16th (2005), 2. 2

[PKD07] PITIÉ F., KOKARAM A. C., DAHYOT R.: Automated colour grading using colour distribution transfer. *Computer Vision and Image Understanding 107*, 1 (2007), 123–137. 3

[PW09] PELE O., WERMAN M.: Fast and robust earth mover's distances. In *Computer vision, 2009 IEEE 12th international conference on* (2009), IEEE, pp. 460–467. 3

[RFP14] RABIN J., FERRADANS S., PAPADAKIS N.: Adaptive color transfer with relaxed optimal transport. In *Image Processing (ICIP), 2014 IEEE International Conference on* (2014), IEEE, pp. 4852–4856. 3

[RGB16] RAYMOND B., GUENNEBAUD G., BARLA P.: Multi-scale rendering of scratched materials using a structured SV-BRDF model. *ACM Transactions on Graphics (TOG) 35*, 4 (2016), 57. 2

[RMS*08] RUMP M., MÜLLER G., SARLETTE R., KOCH D., KLEIN R.: Photo-realistic rendering of metallic car paint from image-based measurements. In *Computer Graphics Forum* (2008), vol. 27, Wiley Online Library, pp. 527–536. 2, 4

[RP11] RABIN J., PEYRÉ G.: Wasserstein regularization of imaging problem. In *Image Processing (ICIP), 2011 18th IEEE International Conference on* (2011), IEEE, pp. 1541–1544. 3

[RPDB11] RABIN J., PEYRÉ G., DELON J., BERNOT M.: Wasserstein barycenter and its application to texture mixing. In *International Conference on Scale Space and Variational Methods in Computer Vision* (2011), Springer, pp. 435–446. 3

[RSK09] RUMP M., SARLETTE R., KLEIN R.: Efficient resampling, compression and rendering of metallic and pearlescent paint. In *VMV* (2009), pp. 11–18. 2

[RSK13] RUITERS R., SCHWARTZ C., KLEIN R.: Example-based interpolation and synthesis of bidirectional texture functions. In *Computer Graphics Forum* (2013), vol. 32, Wiley Online Library, pp. 361–370. 3

[RT01] RUBNER Y., TOMASI C.: The earth mover's distance. In *Perceptual Metrics for Image Database Navigation*. Springer, 2001, pp. 13–28. 3, 4

[RTG98] RUBNER Y., TOMASI C., GUIBAS L. J.: A metric for distributions with applications to image databases. In *Computer Vision, 1998. Sixth International Conference on* (1998), IEEE, pp. 59–66. 3

[RTG00] RUBNER Y., TOMASI C., GUIBAS L. J.: The earth mover's distance as a metric for image retrieval. *International journal of computer vision 40*, 2 (2000), 99–121. 3, 4

[RYZ11] REN Z., YUAN J., ZHANG Z.: Robust hand gesture recognition based on finger-earth mover's distance with a commodity depth camera. In *Proceedings of the 19th ACM international conference on Multimedia* (2011), ACM, pp. 1093–1096. 3

[Sch94] SCHLICK C.: An inexpensive BRDF model for physically-based rendering. In *Computer graphics forum* (1994), vol. 13, Wiley Online Library, pp. 233–246. 3

[SDGP*15] SOLOMON J., DE GOES F., PEYRÉ G., CUTURI M., BUTSCHER A., NGUYEN A., DU T., GUIBAS L.: Convolutional wasserstein distances: Efficient optimal transportation on geometric domains. *ACM Transactions on Graphics (TOG) 34*, 4 (2015), 66. 2, 3

[TTOO90] TAKAGI A., TAKAOKA H., OSHIMA T., OGATA Y.: Accurate rendering technique based on colorimetric conception. In *ACM SIGGRAPH Computer Graphics* (1990), vol. 24, ACM, pp. 263–272. 2

[TWB05] TAKAGI A., WATANABE A., BABA G.: Prediction of spectral reflectance factor distribution of automotive paint finishes. *Color Research & Application 30*, 4 (2005), 275–282. 2

[TZL*02] TONG X., ZHANG J., LIU L., WANG X., GUO B., SHUM H.-Y.: Synthesis of bidirectional texture functions on arbitrary surfaces. In *ACM Transactions on Graphics (ToG)* (2002), vol. 21, ACM, pp. 665–672. 3

[Was69] WASSERSTEIN L. N.: Markov processes over denumerable products of spaces describing large systems of automata. *Problems of Information Transmission 5*, 3 (1969), 47–52. 4

[YHJ*14] YAN L.-Q., HAŠAN M., JAKOB W., LAWRENCE J., MARSCHNER S., RAMAMOORTHI R.: Rendering glints on high-resolution normal-mapped specular surfaces. *ACM Transactions on Graphics (TOG) 33*, 4 (2014), 116. 2

[YHMR16] YAN L.-Q., HAŠAN M., MARSCHNER S., RAMAMOORTHI R.: Position-normal distributions for efficient rendering of specular microstructure. *ACM Transactions on Graphics (TOG) 35*, 4 (2016), 56. 2

[ZDW*05] ZHOU K., DU P., WANG L., MATSUSHITA Y., SHI J., GUO B., SHUM H.-Y.: Decorating surfaces with bidirectional texture functions. *IEEE Transactions on Visualization and Computer Graphics 11*, 5 (2005), 519–528. 3