AxF - Appearance exchange Format

Gero Müller, Jochen Tautges, Alexander Gress and Francis Lamy
X-Rite, Inc., 4300 44th St. SE, Grand Rapids, MI 49505

March 23, 2018
Version 1.5

Abstract

In this short paper X-Rite presents a file format for exchanging digital material appearance. The format is a vital part of X-Rite's appearance initiative consisting of a new generation of appearance capturing devices. In order to make the usage of measured appearance as simple as possible for users, a broad support by software vendors is essential. Therefore, X-Rite aims to initiate the assembly of a consortium of hard- and software vendors, members of the scientific community and users, which would be responsible for the further development and standardization of the format.

1 Motivation

Driven by the ongoing virtualization of the product design process we see a growing demand for measured material appearance. This trend pushes the need for efficient and standardized exchange, communication and archiving of digital appearance data.

Nonetheless, current digital material workflows are far from being standardized. In practice, most often proprietary data formats are used. If materials need to be transferred between software packages data usually needs to be stripped down to a least common denominator like an image or a simple reflectance model like Blinn-Phong which is used for instance by OpenGL and Direct3D's fixed function pipeline. While workarounds can be found for individual cases, this situation is not acceptable for users of appearance measurement devices because a physical material measurement should be independent of specific 3D rendering software platforms.

We are aware of the fact that due to the highly competitive and innovative nature of the field it is not an easy task to reach for standardization in graphics. On the other hand, starting this endeavor from the measurement side seems to be a promising approach. It should be the common goal of both software and hardware vendors to achieve a consistent reproduction of measured appearance across different systems. This, and nothing less, is expected by the users.

2 Requirements

Besides mandatory requirements like efficiency and platform independence we want to bring attention to some special requirements for appearance data.

Scalability The format should be able to store raw measurement data which easily can range up to several gigabytes. Therefore the format must be scalable, which means access times should not depend on the filesize. This requirement rules out text-based formats like XML which lack indexing capabilities.

Generality The structure of appearance data can be quite diverse, ranging from a single spectrum up to complex combinations of data from hundreds of sensors or even procedural descriptions. Therefore, the format must be extensible and self-describing in order to accommodate all these different kinds of representations. This requirement does not only hold for payload but also for metadata.

Workflow compatibility Supporting a least common denominator of appearance representations in addition to a full blown BTF or even BSSRDF is critical for format adoption. This requires the ability to store different versions of a material in a single file. We propose to define an SVBRDF variant as least common denominator since most state of the art rendering software supports at least a material model based on diffuse+specular color texture, normal and/or height map and some specification of gloss.

In an industrial environment, data security and protection are also important and will become an integral part of the format in future versions.

3 Design Decisions

Based on the above requirements, AxF is designed in layers. Since a binary format is inevitable for efficiency and scalability we first define a binary layer. On top of the binary layer we propose a basic structuring mechanism which allows to define the hierarchical semantic structure of payload and metadata. The final layer consists of the explicit definition of the semantic structures.

We propose to use the HDF5 [hdf15] format as *binary base layer*. HDF5 defines a versatile data model which can represent any kind of complex data objects and metadata. The format imposes no (practical) limits on the number or size of data objects in the file and is available for all major platforms.

The second layer implements a simple but powerful *property node tree* concept on top of HDF5 (cf. Figure 1). This concept is analog to a basic filesystem concept where nodes correspond to folders (structuring) and properties to files (payload). Properties support various typical datatypes like integral types, strings and unbound multi-dimensional data arrays.

The third layer defines a *baseline* for valid AxF files that partly should be supported by all applications integrating AxF. Essentially it defines a set

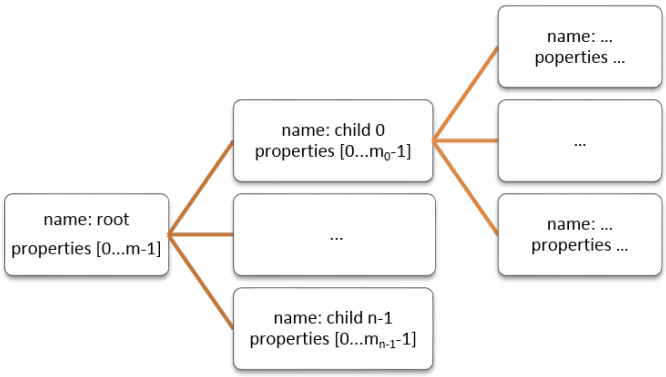

Figure 1: Property node tree concept

of node- and property-types and their semantics as described in the following section.

4 Baseline AxF

The basic semantic structure of an AxF file is related to the Color Exchange Format [CxF] which has been defined by X-Rite as a portable format for system and device independent exchange of color data and which now has been published as an ISO Standard: ISO17972 Graphic technology - Colour data exchange format (CxF/X).

An exhaustive list of all node types and their semantics is beyond the scope of this document but to convey the idea we will briefly sketch the three so-called root-level node collections as shown in Figure 2:

DeviceSpecCollection contains specification and calibration data for measurement devices. It allows the specification of geometric and radiometric device properties.

ColorSpecCollection specifies arbitrary (linear) color spaces and spectral samplings. Since AxF stores floating point color values, non-linear color spaces are not part of the current specification.

MaterialCollection is a general container for metadata, semantics (*RepresentationCollection*) and payload (*ResourceCollection*) of measured materials.

The *RepresentationCollection* as part of a material definition deserves further explanation: it contains the particular material models like a BRDF or BTF model which references basic (simply structured) payload like colors and textures stored in the *ResourceCollection*. Please note, that the general layer concept of AxF does not restrict the range of possible representations in any way.

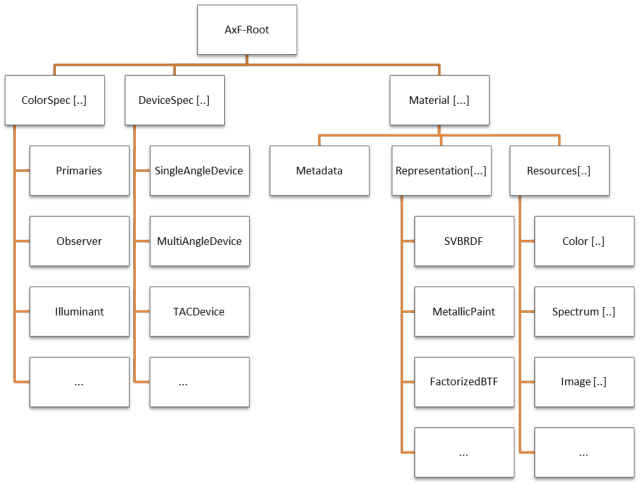

Figure 2: Excerpt from the definition of *Baseline AxF*. Collections are denoted by [..].

It is mainly a question of support by devices and software and upcoming standardization efforts, which representations will become an obligatory part of AxF in the future.

For a start, X-Rite defined a set of basic representations that should be supported by AxF-capable rendering and visualization applications. In the following, we give a brief overview on the different representations.

4.1 Built-in Representations

Currently, AxF supports three different types of so-called *Representation Classes* which refer to specific classes of related reflectance models and have been chosen to support a wide range of material types and target applications. Currently there are the *SVBRDF*, *CarPaint* and *FactorizedBTF* representation classes. For each class AxF defines a baseline of a single or even several appearance models which should be implemented by rendering applications aiming to support AxF. The properties and features of the respective baseline models are defined using *representation compatibility profiles*. E.g., for the SVBRDF class the corresponding baseline profile includes a classic layered shading model based on the Ward BRDF [War92] and texturing[1]. For the *FactorizedBTF* class the profile defines a BTF compression technique as in [MMK03] while the *CarPaint* class includes a model for measured effect paints like in [RMS+08].

[1]For the sake of brevity we omit the formal introduction of standard computer graphics entities and algorithms like BRDFs, BTFs or texture mapping here. Please consult related literature on the topic.

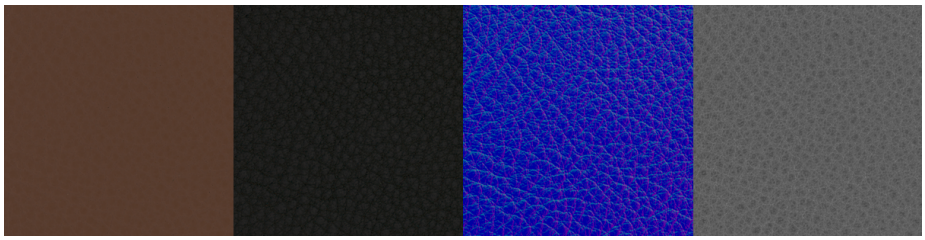

Figure 3: The standard SVBRDF (isotropic Ward) parameter textures of a measured leather material (from left to right): *diffuse, specular, normal, roughness.*

Spatially Varying BRDF AxF defines a general SVBRDF representation based on the following evaluation of a BRDF consisting of a diffuse and a specular component with spatially varying parameters stored in textures (as shown in Figure 3):

$$f(\mathbf{x},\mathbf{i},\mathbf{o}) = f_d(T_{c_d}(\mathbf{x}), T_N(\mathbf{x}); \mathbf{i},\mathbf{o}) + f_s(T_{c_s}(\mathbf{x}), T_{p_s}(\mathbf{x}), T_N(\mathbf{x}); \mathbf{i},\mathbf{o})$$

The texture operator $T(\mathbf{x})$ represents a discrete 2D table of values of various types indexed by the texture coordinate $\mathbf{x}$, e.g. $T_c(\mathbf{x})$ contains color values (typically trichromatic RGB values) which are stored as weights for the diffuse and the specular BRDF part. $T_N(\mathbf{x})$ is the *normal map* which is used to vary the local coordinate system in order simulate a bumpy surface. Last but not least, $T_{p_s}(\mathbf{x})$ is a texture of specular model parameters like roughness.

The diffuse model f_d is simply Lambert (neglecting the normal map parameter which influences the incoming and outgoing directions $\mathbf{i}$, $\mathbf{o}$):

$$f_d^{lambert}(a;\mathbf{i},\mathbf{o}) = \frac{a}{\pi}$$

AxF's default specular model f_s is a variant of the Ward model as proposed by Geisler-Moroder [GMD10] (again neglecting the normal map):

$$\begin{aligned} f^{ward}(s,\alpha,\beta;\mathbf{i},\mathbf{o}) &= s\frac{1}{\pi\alpha\beta}N_{GM2010}e^{-\frac{1}{h_z^2}\left(\frac{h_x^2}{\alpha^2}+\frac{h_y^2}{\beta^2}\right)} \\ N_{GM2010} &= \frac{\langle\mathbf{h},\mathbf{h}\rangle}{h_z^4} \end{aligned}$$

whereas $\mathbf{h}$ denotes the half-angle vector.

Beyond the introduced basic model, AxF starting with version 1.1 also supports an extension of the Ward BRDF based on Schlick's Fresnel approximation [Sch94].

Version 1.5 adds support for the increasingly popular GGX Brdf model [WMLT07].

Displacement maps (heightfields) for modeling height variation and alpha maps for simple transparency are part of the AxF baseline starting with version 1.4 and are described in more detail below.

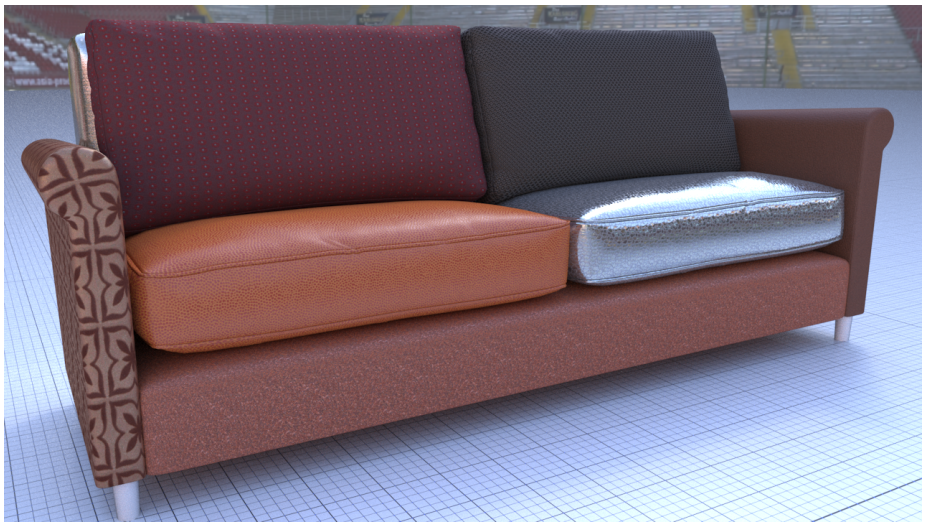

Figure 4: The *"AxF Sofa"* covered with several measured AxF SVBRDF materials like fabrics, plastics, leather and metals rendered using Autodesk VRED. Some of the materials are taken from the MAM-2014 sample set [mam14].

Figure 4 shows a rendering using a few examples of measured SVBRDFs represented using AxF's basic SVBRDF model.

Coated Materials An important material class are materials made of different semi-transparent layers. For the sake of simplicity we restrict Baseline AxF to materials with a single *clear coat* on top of an opaque base material as sketched in Figure 5.

While these materials occur surprisingly often in industry they can still be measured and modeled with acceptable additional effort compared to standard SVBRDFs. In particular it is assumed that the coating is completely transparent (no absorption and scattering despite at the layer boundaries), infinitesimal thin (refracted rays from the coating hit the lower layer at the same point) and solely defined by its (spatially varying) index of refraction and an (optional) normal map which allows to model mesoscopic effects like the *Orange Peel*-effect shown in Figure 6. According to [WW07] we evaluate the two layers f_c and f_b (coating and base) as follows (with $\bar{\mathbf{i}} = \mathbf{refract}(\mathbf{i}, n_c, n_{air})$ denoting the refracted ray):

$$f(\mathbf{i}, \mathbf{o}) = f_c(\mathbf{i}, \mathbf{o}) + T_{air \to c} f_b(\bar{\mathbf{i}}, \bar{\mathbf{o}}) \cdot a_c \cdot t_c$$

We neglect absorption ($a_c = 1$) and for the time being also don't consider the effect of total internal reflection which simplifies t_c to $T_{c \to air} = 1 - F(\mathbf{o})$. As a consequence the energy reflected back to the base is formally lost but please keep in mind that AxF stores models fitted to real data which means that this energy will actually be "baked" into the base SVBRDF f_b during fitting. Hence

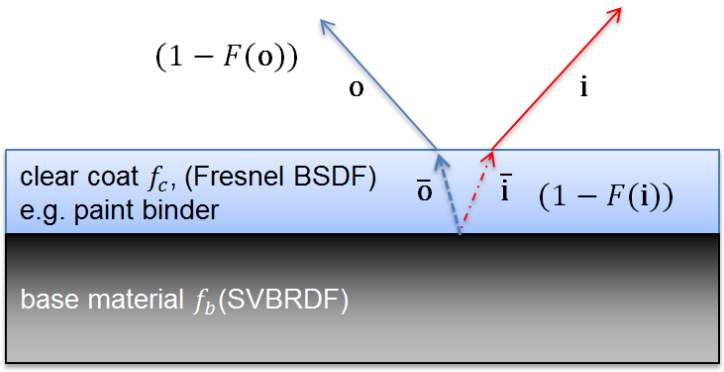

Figure 5: Base SVBRDF material covered by a transparent Fresnel BSDF layer.

f_b will not fully model the true underlying BRDF (which will only become apparent if the base material is rendered without the coating). In fact, this general strategy is well inline with the whole concept of using an SVBRDF for modeling a real-world material: interaction i.e. (sub-surface) scattering between different surface points is not factored out but baked into the per-pixel BRDF. Finally we arrive at the following formula for the coated BRDF:

$$f(\mathbf{i}, \mathbf{o}) = f_c(\mathbf{i}, \mathbf{o}) + (1 - F(\mathbf{i})) \cdot f_b(\bar{\mathbf{i}}, \bar{\mathbf{o}}) \cdot (1 - F(\mathbf{o}))$$

whereas $f_c(\mathbf{i}, \mathbf{o}) = F(\mathbf{i}) \cdot \delta(reflect(\mathbf{i}) - \mathbf{o})$ is the perfect reflection. In practice, Schlick's Fresnel approximation can be used here, which means the additional effort for evaluating such a coated SVBRDF is the perfect reflection, two Fresnel terms and two refractions.

In order to ensure compatibility with many existing rendering systems starting with version 1.3 AxF also supports a non-refractive clearcoat layer, i.e. the directions for evaluating the base layer are not refracted. Similar to the effect of the total internal reflection the refraction will then be "baked" into the SVBRDF model.

Alpha Mapping Starting with version 1.4 AxF officially supports also standard alpha mapping (e.g. [PD84]) for SVBRDFs. By assuming an infinitely thin material with a spatially varying absorption coefficient between 0 (completely transparent) and 1 (fully opaque) this map can be used to model simple transparency. The left image in Figure 7 shows a typical example how alpha mapping can be used to represent a perforated leather material.

Displacement Mapping Another convenient and widely used technique is displacement mapping (first introduced in [Coo84]) which stores spatially varying height in a scalar map which is used by the rendering engine to locally offset and re-tesselate the geometry. Starting with version 1.4 AxF officially specifies

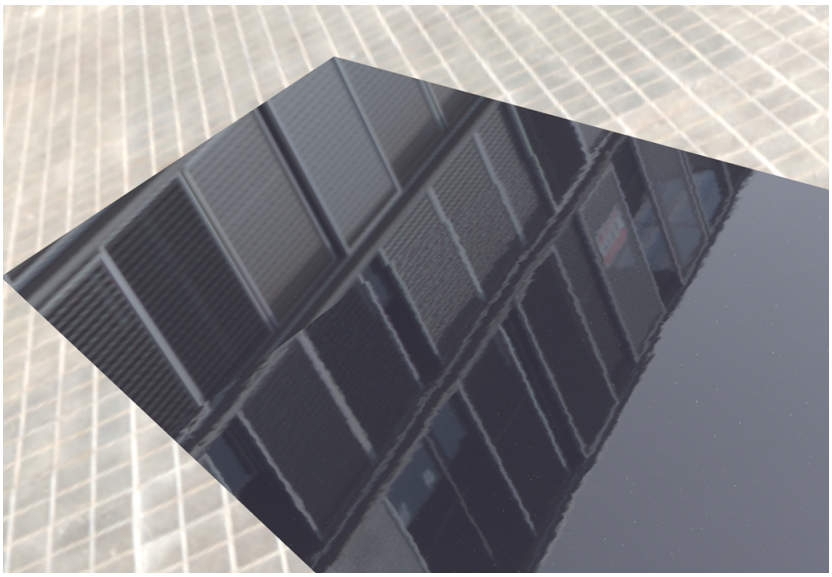

Figure 6: The Orange Peel effect shown for a measured paint material. For comparison the rear patch has a perfectly smooth coating.

a displacement aka height map for SVBRDFs. The right part of Figure 7 shows an example of a displacement mapped textile.

Measured BTF Since BTFs are not in widespread use they are defined in a different compatibility profile than SVBRDFs. BTFs are naturally understood as a multi-dimensional table of reflectance measurements of a material sample.

Let us consider that each such measurement corresponds to sampling the general 8-dimensional reflectance field $R_V(\mathbf{x}_i, \omega_i; \mathbf{x}_o, \omega_o)$ which transfers incident light fields to their corresponding outgoing light fields parameterized on a bounding surface V. See Figure 8 for an illustration.

The discrete version of this operator is called the *light transport matrix* $\mathbf{R}$. It can be obtained by discretizing the domain of the incoming and outgoing light field:

$$\mathbf{L}_o = \mathbf{R}^{(K \times I)} \mathbf{L}_i$$

By assuming distant incoming lighting a typical parameterization of the incoming lighting is by directions of incoming light sources:

$$I = \{(\theta_{i,0}, \phi_{i,0}), (\theta_{i,1}, \phi_{i,1}), \ldots, (\theta_{i,l}, \phi_{i,l})\}$$

K describes the discretization of the outgoing light-field and is usually given by the image or texture size $E = W \times H = \{1, 2, \ldots, w\} \times \{1, 2, \ldots, h\}$ and a set of outgoing directions $O = \{(\theta_{o,0}, \phi_{o,0}), (\theta_{o,1}, \phi_{o,1}), \ldots, (\theta_{o,v}, \phi_{o,v})\}$ similar

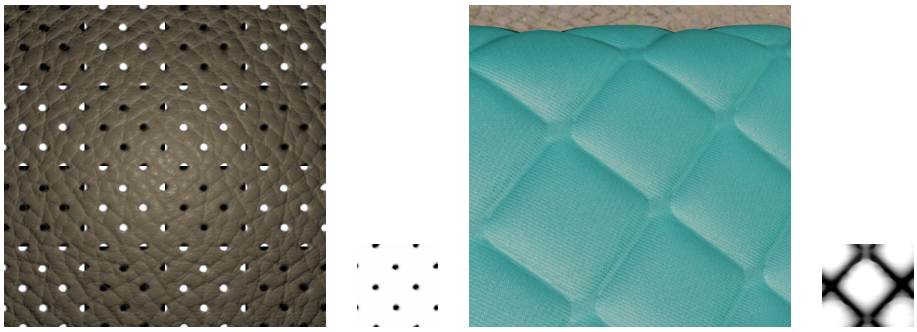

Figure 7: Left: Perforated leather overlaid onto a checkerboard pattern and the corresponding measured alpha map representing the holes. Right: Textile rendering with measured displacement. Note the geometry's silhouette which would've been a perfect line without displacement.

as above and hence

$$K = E \times O = \{(1, 1, \theta_{o,0}, \phi_{o,0}), (1, 1, \theta_{o,1}, \phi_{o,1}), \ldots, (w, h, \theta_{o,v}, \phi_{o,v})\}.$$

In this sense the light transport matrix $\mathbf{R}_{(K \times I)}$ as given above can considered as a 2D table where each of the two dimensions enfolds a higher-dimensional index set: the 2D incoming directions are enfolded along the columns and the four dimensions of spatial position and outgoing direction are enfolded along the rows of the matrix.

The basic idea of matrix factorization-based appearance representations is now to apply matrix factorization techniques to the sampled data arranged into a 2D matrix and use these techniques for data compression.

Since the mathematical field of matrix factorization is huge, we'll leave it here with the most common definition of the Single Value Decomposition which states that each $m \times n$ matrix $\mathbf{A}$ of rank r can be decomposed into the product $\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^T$ where $\mathbf{U}$ and $\mathbf{V}$ are unitary matrices and $\mathbf{S}$ is a diagonal matrix with $s_{ii} \geq s_{i+1,i+1} > 0$ for $1 \leq i < k$ and $s_{ii} = 0$ for $k+1 \leq i \leq \min(m, n)$. The numbers s_{ii} are the nonnegative square roots of the eigenvalues of $\mathbf{A}\mathbf{A}^T$. The columns $\mathbf{u}_j$ of $\mathbf{U}$ are the eigenvectors of $\mathbf{A}\mathbf{A}^T$ and the columns $\mathbf{v}_j$ of $\mathbf{V}$ are the eigenvectors of $\mathbf{A}^T\mathbf{A}$.

The key insight for data compression is now the well-known *Eckhart-Young Theorem* which states that the matrix

$$\mathbf{A}_{r_0} := \sum_{j}^{r_0} \mathbf{u}_j s_{jj} \mathbf{v}_j^T$$

is the best rank-r_0 approximation of $\mathbf{A}$ in the least-squares sense. Since the storage requirements for $\mathbf{A}_{r_0}$ are $O\left((m+n)r_0\right)$ compared to $O(mn)$ we have a compression ratio of $\left(\frac{r_0}{n} + \frac{r_0}{m}\right)$.

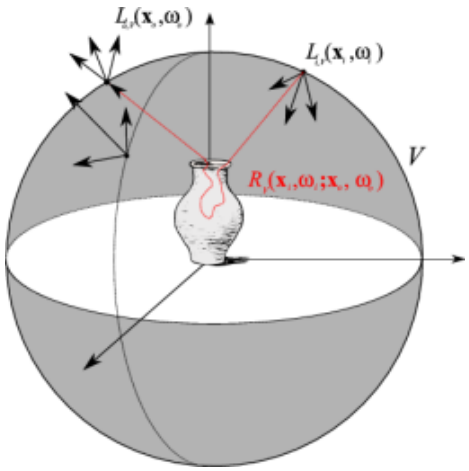

Figure 8: The reflectance field.

The 2D layout of 6D appearance data (which is in fact a BTF) as sketched above is not optimal for data compression since the compression ratio becomes optimal if n and m, i.e. the number of rows and columns are roughly of the same size. We won't go too much into detail here but a more balanced arrangement which puts the spatial dimension E in the rows and the angular dimension $I \times O$ in the columns has proven to be quite convenient i practice:

$$\mathbf{R}_{E\times(O\times I)} = \mathbf{U}_E \mathbf{S} \mathbf{V}^T_{I\times O}$$

Now the matrix $\mathbf{U}_E$ contains in its columns the so-called *EigenTextures* and the columns of $\mathbf{V}_{I\times O}$ are the *EigenBRDFs*. Figure 9 shows an example for these factorized components by depicting the first three rows or columns respectively whereas the 4D EigenBRDFs have been enrolled into a 2D image.

Measured Carpaint Since metallic paints as they are typically used in automotive industry are not well represented by either SVBRDFs or standard BTFs AxF supports a specialized representation for measured metallic paints in the spirit of the work of Rump et al. [RMS+08]. Again, we do not go too much into details here but refer the reader to the related scientific literature.

We'll leave it with the introduction of the main components such that general complexity and expressiveness of the model can be judged quickly by the experienced reader.

The model consists of three main components: first, a BRDF part which models the angular brightness variation of the paint, second, a angular dependent color table which captures low-frequency color shift of modern effect paints and, finally, a BTF part which captures the visible flakes. The three components are sketched in the following paragraphs:

EigenBRDF EigenTexture Rendering (OpenGL)

Figure 9: The first three angular and spatial components of a factorized BTF encoded in a color mapped RGB image. The right image shows a rendering of the material.

Brightness BRDF The BRDF is modeled using a multi-lobe Cook-Torrance:

$$f(\bar{\mathbf{i}}, \bar{\mathbf{o}}) = \frac{a}{\pi} + \sum_{k=1}^{K} \frac{s_k}{\pi} \frac{D_{\alpha_k}(\bar{\mathbf{h}}) F_{F_{0,k}}(\bar{\mathbf{h}}, \bar{\mathbf{o}}) G(\bar{\mathbf{i}}, \bar{\mathbf{o}})}{\bar{i}_z \bar{o}_z}$$

We use the Beckmann distribution, the original geometry term [CT82] and Schlick's Fresnel approximation. Although not finally decided yet $K = 3$ is the default number of lobes and can be assumed as de-facto standard. Please note the difference in notation for the direction vectors compared to the SVBRDF representation introduced previously. This should indicate that all directions used in the model are defined *below* the clear coat. In this case, e.g., the incoming direction is the refracted direction $\bar{\mathbf{i}} = \mathbf{refract}(\mathbf{i}, n_{cc}, n_{air})$ whereas n_{cc} denotes the (measured) clear coat's index of refraction. The other directions are defined accordingly.

Color Table The BRDF brightness term is modulated by a 2D color table $\chi(\cdot, \cdot)$ parameterized by the angle between half vector and normal $\theta_{\bar{\mathbf{h}}} = \arccos(\bar{h}_z)$ and the angle between half vector and incoming direction $\theta_{\bar{\mathbf{i}}} = \arccos(\langle \bar{\mathbf{h}}, \bar{\mathbf{i}} \rangle)$. In classical microfacet modeling $\theta_{\bar{\mathbf{h}}}$ is interpreted as the angle between the overall surface normal and the microfacet normal. In the case of a metallic paint we identify microfacets with metallic flakes, hence $\theta_{\bar{\mathbf{h}}}$ can be interpreted as the angle between the overall surface normal and the normal of a flake (modeled as a perfect mirror). Correspondingly, $\theta_{\bar{\mathbf{i}}}$ is the angle between the refracted illumination direction and the flake normal.

Flake BTF Textures The spatially varying part, i.e. the visible flakes, is also parameterized based on $\theta_{\bar{\mathbf{h}}}$ and $\theta_{\bar{\mathbf{i}}}$ but now each entry does not contain a single color value but a complete flake texture slice as shown on the left of

Figure 10: The left image shows a single slice from flake texture table. The other images show renderings with different illumination conditions.

Figure 10 which actually leads to a 4-D table $\Xi(\mathbf{x}, \theta_{\bar{\mathbf{h}}}, \theta_{\bar{\mathbf{i}}})$. The sampling of the angular space can actually be set quite sparse which results in overall texture sizes of 20-50 Megabytes.

Combined Model To summarize, the complete spatially varying paint model reads as follows:

$$f(\mathbf{x}, \bar{\mathbf{i}}, \bar{\mathbf{o}}) = \chi(\theta_{\bar{\mathbf{h}}}, \theta_{\bar{\mathbf{i}}}) \left(\frac{a}{\pi} + \sum_{k=1}^{3} f^{CT}_{s_k, \alpha_k, F_{0,k}}(\bar{\mathbf{i}}, \bar{\mathbf{o}}) \right) + \Xi(\mathbf{x}, \theta_{\bar{\mathbf{h}}}, \theta_{\bar{\mathbf{i}}})$$

5 Integration and Performance

For the integration of AxF into third-party applications, X-Rite provides a C/C++ based SDK, which is made available on request under a royalty-free license (contact one of the authors or LicensingTeam@XRITE.com directly).

The SDK is fully documented and comes with example code. As demonstrated there, the usage of AxF usually begins with opening the file and analyzing its contents, which typically consists of the following steps:

1. Retrieve a valid AxF file handle using `axfOpenFile()`[2].
2. Select a material (by index) using `axfGetMaterial()`.
3. Retrieve a valid representation handle H for the selected material, e.g. using `axfGetPreferredRepresentation()`.
4. Determine the type of the representation as follows:
   (a) Retrieve the main appearance class (*SVBRDF*, *CarPaint*, or *FactorizedBTF*) using `axfGetRepresentationClass(`H`)`.

[2] All functions mentioned here are defined in namespace `axf::decoding`.

(b) In case of an SVBRDF, retrieve the handles H_d and H_s for the diffuse and specular sub-components of the representation, using `axfGetSvbrdfDiffuseModelRepresentation(`H`)` and `axfGetSvbrdfSpecularModelRepresentation(`H`)`, respectively, from which the following additional information can be queried:

i. the particular diffuse BRDF model (so far always *Lambert*) using `axfGetRepresentationTypeKey(`H_d`)`;

ii. the particular specular BRDF model (*Ward*, *Cook-Torrance*, ...) using `axfGetRepresentationTypeKey(`H_s`)`;

iii. the precise variant of the specular BRDF model using `axfGetSvbrdfSpecularModelVariant(`H_s`)` and `axfGetSvbrdfSpecularFresnelVariant(`H_s`)`.

5. Decide if the found representation is supported by the client application, either by analyzing the previously determined representation type or via the helper function `axfCheckRepresentationCompatibilityProfile()`, which can be exploited if the client application supports one of the predefined compatibility profiles.

5.1 Decoding AxF Representations

After selecting a supported representation it comes to the decoding (i.e. rendering), which can be done in two ways: either by using the SDK's built-in CPU-based decoding functionality or by retrieving the appearance model parameters from the AxF file and implementing and evaluating the model in the client application.

The SDK implementations for CPU-based decoding are not aggressively optimized, but rather should be considered as reference implementations of the particular representations and should be used primarily for reference and testing purposes. In particular, we do not recommend to use this way of decoding in performance critical production environments.

Instead, we encourage client applications to implement the representations internally and use the `axf::decoding::TextureDecoder` interface for retrieval of the appearance model parameters (i.e. textures). The technical details required for implementing a particular representation can be found in section 4.1 and the SDK documentation. Furthermore, the documentation contains pseudo-code for each representation which can be used as starting point.

5.2 Handling Colors and Spectra

AxF supports storing color resources in monochromatic, trichromatic, pseudo-spectral and full spectral color spaces. To facilitate that, each color resource is associated with a certain *ColorSpec* node (see the *ColorSpecCollection* sketched in section 4), which specifies the corresponding color space precisely.

In any case, AxF uses a *linear* color space (i.e. a linear subspace of the spectral space) to store a color resource. Please note that we consider a color (from a

color resource) to correspond to a reflectance spectrum (i.e. *spectral reflectance* in physical terms) or to a projection thereof into the respective linear subspace that the ColorSpec represents. A unity reflectance spectrum represents a neutral material, i.e. one for which the spectrum of the reflected light corresponds exactly to the spectrum of the incoming light.

Currently, the information from the ColorSpec node cannot be retrieved explicitly via the SDK. Instead the client application defines a target color space into which all retrieved color resources will be converted (based on the AxF-internal ColorSpec node) before returning them to the client. Details can be found in the SDK documentation.

Spectral Rendering It is planned to add functions to the SDK to query color resources in spectral space as well, but right now this is not yet supported. Nonetheless, besides selecting a trichromatic target color space and querying color resources in this target color space, the SDK also allows to query a "spectralization transformation", which is a matrix that can be used to transform a trichromatic color from the selected target color space to a corresponding spectrum (for any given spectral sampling). Obviously, this cannot always guarantee a precise reconstruction of the original spectra. Nonetheless, the SDK tries to determine a transformation that is optimized for reconstructing the primary material spectra as good as possible, provided that respective spectral information is available in the AxF file. So this transformation might improve significantly upon a standard RGB-to-spectral conversion that doesn't take such additional spectral information into account.

5.3 Performance and Memory Consumption

Measured appearance has often been considered as being more resource demanding than *tweaked* materials based on manually edited texture images. This has been true for BTF-based material representations which rely on dense sampling of both the spatial and the angular dimension of appearance, but it does not apply to measured materials based on the SVBRDF representation. The reason is simply that even hand-tweaked materials are essentially SVBRDFs: every texel stores parameters of a certain BRDF model. As a result most of the optimization techniques which are applied to hand-tweaked materials can also be applied to measured materials with a similar effect.

While the numbers given in the following for illustrating AxF storage requirements are based on SVBRDFs, most of the techniques could also be applied to BTFs and Carpaint representations.

A typical AxF 1.1 anisotropic SVBRDF resulting from a TAC7 measurement consists of the following parameters:

- Diffuse Color, $|T_{c_d}(\mathbf{x})| = 3$ channels
- Normal Map, $|T_N(\mathbf{x})| = 3$ channels
- Specular Color, $|T_{c_s}(\mathbf{x})| = 3$ channels

- Anisotropic Roughness, $|T_{p_s}(\mathbf{x})| = 3$ channels[3]
- Fresnel, $|T_{F_0}(\mathbf{x})| = 1$ channel
- Material dependent:
  - Clearcoat Layer IOR, 1 channel
  - Clearcoat Layer Normal Map, 3 channels

Without clearcoat, such an SVBRDF has 13 channels and given a spatial resolution of 1024×1024 pixels this leads to $13 \times 1024 \times 1024 \times 4 = 52\,\mathbf{MB}$ (additional 16 MB for clearcoat) of uncompressed data in 32-bit precision.

Note that the storage requirement for the AxF file itself is in general considerably lower, since AxF applies common lossless compression techniques for storing the data. However, this kind of compression is not suitable for random access to the data, thus the data is decompressed transparently when reading it from the AxF file. Therefore, we only consider the uncompressed in-memory storage requirements of AxF materials in the following.

Using roughly 50 MB of storage per material can impose a restriction for the use of AxF materials in real-time applications and on low-performance systems, for instance at the point-of-sale. Fortunately, the in-memory storage requirements can be reduced in numerous ways for most practical applications:

1. Before measurement: By selecting the optimal representation, i.e. SVBRDF model, for the material and use case. This may correspond simply to using less BRDF parameters (channels) like in the isotropic vs. anisotropic case. Or to the reduction of spatial variation by assuming that some parameters are constant over the sample, for instance by using the *global F0* option which assumes a constant Fresnel parameter or the *dielectric material* option which assumes colorless specular reflections and thereby eliminates chromaticity variation in the specular color texture.

2. During data preparation: By using the AxF Editor of X-Rite's **Digital Material Hub** (cf. Figure 11) to reduce the spatial variation in certain BRDF parameter maps and by sharing AxF resources between materials where appropriate (shown in Figure 12).

   This approach is recommended in particular if the material has a dominant base color or BRDF. Think of a material like single-colored molded plastic, which has an almost constant BRDF but high variation in the normal and/or displacement map.

3. While requesting data from the SDK: By choosing an appropriate data type (for instance half-precision floating point compared to 32-bit precision) and the required (maximum) spatial resolution, in which to store textures in memory.

[3] In AxF, this is in general represented by two resources, a 2-channel *Roughness* map (representing roughness values along the two principal axes) and a single-channel *Anisotropic Rotation* map (containing rotation angles).

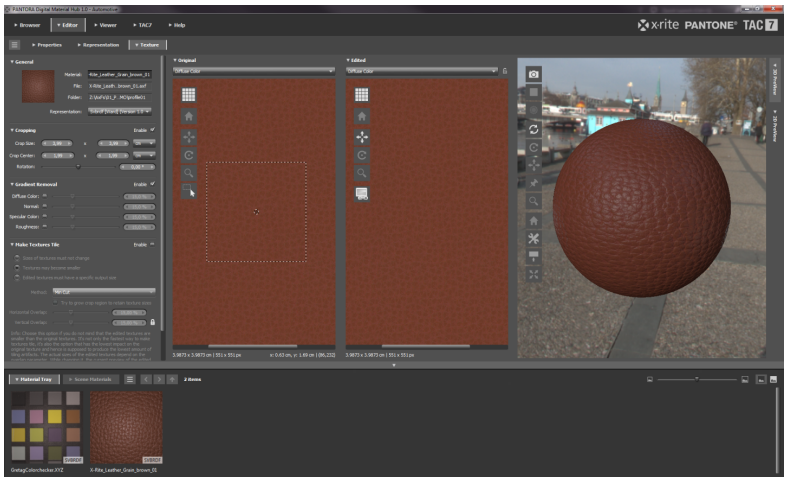

Figure 11: AxF Editor integrated into X-Rite's Digital Material Hub.

4. By further reducing the memory footprint of texture resources in the client application, for instance by applying certain texture compression techniques or by supporting the sharing of resources as represented in the AxF file.

By applying and combining theses techniques, storage reductions from 50%-80% compared to the original 32-bit precision can easily be achieved without losing significant visual fidelity. In our example this might result in in-memory storage requirements of only 10 MB.

Again, further details about how these techniques are applied in practice can be found in the SDK documentation.

6 Wrap-up

In this short paper we introduced X-Rite's proposal for an appearance exchange format. In the presented form it is mainly a general framework which handles the low-level details like efficient storage and access of payload and metadata, how colors are specified etc. Although a first set of representations has been defined by X-Rite and first implementations into commercial software like NVidia's MDL [MDL15] or Autodesk VRED are available (cf. Figure 4 for an example), the definition of the top-layer semantic (Baseline AxF) and especially its material representations are by far not complete.

Further definition of the format needs to focus on additional representations which extent the gamut of materials that can be well represented in a portable, i.e. exchangeable way. Obviously, this cannot be achieved by X-Rite or any measurement devices vendor alone but has to be done in close collaboration with both the software vendors, who decide which kind of material models they

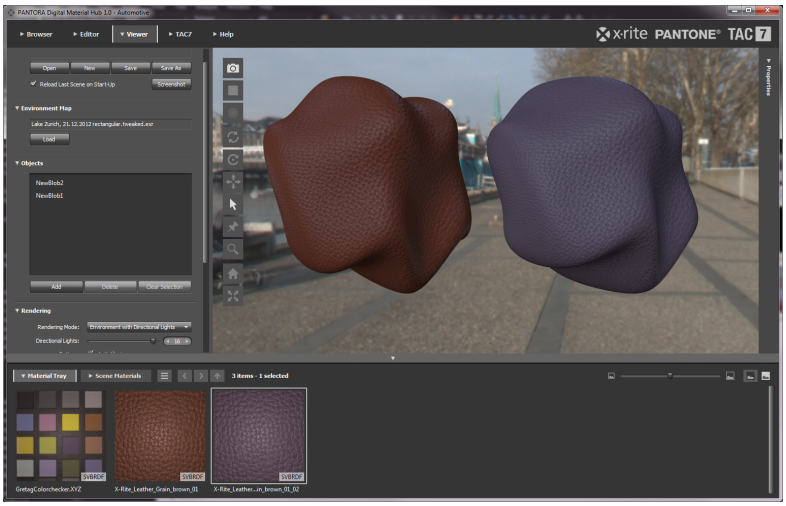

Figure 12: Two leather materials with different diffuse base color, but sharing the same diffuse and specular intensity map and normal map resources.

will support and integrate in their software, and the users, who define the types of materials that need to be digitized as accurate as possible in order to increase the quality and credibility of the digital design process.

As a next step, X-Rite plans to make the detailed definition of the AxF node types and structure and also the SDK mentioned in the previous section available to the community short-term. Then the goal would be to build a consortium of hardware and software vendors, members of the research community and users of measured material appearance to define future requirements and demands for additional AxF features and especially material representations. If a broad support for an agreed baseline of representations can be achieved, this can be the first step towards a standardization of digital material appearance making efficient exchange and archival of appearance possible for industry and research.

References

[Coo84] Robert L. Cook. Shade trees. *SIGGRAPH Comput. Graph.*, 18(3):223–231, January 1984.

[CT82] R. L. Cook and K. E. Torrance. A reflectance model for computer graphics. *ACM Trans. Graph.*, 1(1):7–24, January 1982.

[CxF] `http://www.colorexchangeformat.com/documents/literature/CxF/CxF_Standard.pdf`.

[GMD10] David Geisler-Moroder and Arne Dr. A new ward brdf model with bounded albedo. *Computer Graphics Forum*, 29(4):1391–1398, 2010.

[hdf15] Hierarchical data format, version 5, 1997-2015. http://www.hdfgroup.org/HDF5/.

[mam14] MAM-2014 Sample Set, 2014. https://sites.google.com/site/mam2014samples/.

[MDL15] NVidia MDL - Material Definition Language, 2015. http://www.nvidia.com/object/material-definition-language.html.

[MMK03] Gero Müller, Jan Meseth, and Reinhard Klein. Compression and real-time rendering of measured btfs using local pca. In T. Ertl, B. Girod, G. Greiner, H. Niemann, H.-P. Seidel, E. Steinbach, and R. Westermann, editors, *Vision, Modeling and Visualisation 2003*, pages 271–280. Akademische Verlagsgesellschaft Aka GmbH, Berlin, November 2003.

[PD84] Thomas Porter and Tom Duff. Compositing digital images. *SIGGRAPH Comput. Graph.*, 18(3):253–259, January 1984.

[RMS+08] Martin Rump, Gero Müller, Ralf Sarlette, Dirk Koch, and Reinhard Klein. Photo-realistic rendering of metallic car paint from image-based measurements. *Computer Graphics Forum*, 27(2):527–536, apr 2008.

[Sch94] Christophe Schlick. An inexpensive brdf model for physically-based rendering. *Computer Graphics Forum*, 13:233–246, 1994.

[War92] Gregory J. Ward. Measuring and modeling anisotropic reflection. In *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '92, pages 265–272, New York, NY, USA, 1992. ACM.

[WMLT07] Bruce Walter, Stephen R. Marschner, Hongsong Li, and Kenneth E. Torrance. Microfacet models for refraction through rough surfaces. In *Proceedings of the 18th Eurographics Conference on Rendering Techniques*, EGSR'07, pages 195–206, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

[WW07] Andrea Weidlich and Alexander Wilkie. Arbitrarily layered micro-facet surfaces. *Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia GRAPHITE 07*, (1):171, 2007.

