STAG: Smart Tools and Apps in computer Graphics (2016)
Giovanni Pintore and Filippo Stanco (Editors)

Anisotropic MatCap: Easy Capture and Reproduction of Anisotropic Materials

Dario Magri[1], Paolo Cignoni[2] and Marco Tarini[1]

[1]University of Insubria, Italy
[2]Visual Computing Lab, ISTI - CNR, Italy

Abstract

We propose Anisotropic MatCap, a simple data structure based on a small volumetric texture that is able to represent, under a fixed lighting, the behavior of anisotropic materials. The data structure is designed to allow fast and practical capture of real-world anisotropic materials (like for example fabrics) and to be used in real-time renderings, requiring only negligible time and texture memory overheads.
The resulting technique is suited for application scenarios where digital objects must be inspected by an end user, recreating the look of an object made of a captured anisotropic material and seen under the predetermined lighting conditions. The technique proved particularly useful for garments and cloth visualization and design.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Rendering—Image-based Rendering I.3.3 [Computer Graphics]: Rendering—Real-Time Rendering I.3.3 [Computer Graphics]: Rendering—Illumination [Anisotropic BRDF]: —

1. Introduction

Achieving real-time yet photo-realistic rendering is one of the paramount tasks in Computer Graphics. The task requires digital models for: 3D objects, materials, and lighting environments, each element faithfully reproducing the respective real-world counterparts. Fast, GPU-oriented algorithms must are used to put together these three elements, and the data structures used to model them must be designed accordingly. Each element (3D shapes, materials descriptions, environments) can be acquired from reality or designed, and a very wide literature provides a large array of solutions. Invariably, most realistic results can be obtained only at the price of time consuming capture sections, using carefully calibrated equipments, laboratory acquisition settings, and complex, time consuming renderings algorithms.

We propose a material acquisition and rendering technique that by combining the two steps and tying the material to the current lighting environment it faces the whole problem in simple way and allows even the more complex and general case of materials with anisotropic microstructure (like for example most textiles) can be tackled. Acquiring material and lighting environment together implies a loss in flexibility that is balanced by the advantages, making this the ideal technique for a certain number of specific applicative scenarios; specifically, all these where a user investigates an manipulates a virtual 3D object, and the GUI metaphor dictates that the object is being rotated (around its center), while the virtual camera stays fixed (see Sec. 8.1 for a few examples).

Contributions: The main contribution is the introduction of *Anisotropic MatCap* (AMC), a simple data structure designed to capture, store and recreate in HW-based renderings the look of anisotropic real-world materials lit by given real-world illumination environments. The focus is on ease of capture from reality, and rendering efficiency, leveraging common HW-supported mechanisms. Memory and rendering overhead imposed by using this structure in real time environments are small and manageable.

As a minor collateral contributions, in Sec. 5.3 we show how this structure can be used to compute, from a real-world anisotropic material, a fictitious but plausible material that, while being isotropic (or at least while behaving isotropically in the chosen illumination environment), differs the least from the given real material.

2. Related Work

For a large class of materials, the optical behavior is fully captured by the BRDF, a function describing how a sample of that material reflects the lights that hits it. The process of capturing BRDFs from samples is covered by an extremely wide literature, spacing from technique more specific cases (like isotropic BRDF [MPBM03]) to very general ones. Materials with isotropic BRDF are those where

the behavior of a planar sample never changes in response to a rotation of that sample around its normal (this stems from the fact that the microstructure of the surface does not have recognizable directions in tangent space). An analysis of the several (even drastically different) techniques is beyond the scope of this work, and the reader is redirected to the cohomprehsive and well illustrated State of the Art analysis in [LGM07].

Suffices to say here that acquiring BRDF from reality (e.g. by means of a photogoniometer, or with several images shot in controlled lighting conditions) is recognized as a very resource consuming task.

Capturing Lighting Environments

A lighting environment (also called Environment Maps) consists in a measurement of the distribution of light in a static scene with fixed illumination. They usually are captured by means of pictures shot at mirroring spheres reflecting the environment, often in conjunction with HDR techniques. When capturing a light field, light is often assumed to be at infinite distance come from distant light sources, even tough this assumption is sometimes dropped, then more spheres must be used [CCC08]. Even more general, sometimes the light incident to a scene is captured for each direction and each incoming position, resulting in a data structure known as *incident light field* [UWH*03]. Once captured, the lighting environment provides a parameter, the incident light, that must be plugged into a rendering equation in order to be useful for rendering (e.g. [RH02]).

Mixed Techniques

Ramamoorthi et al. [RH01] noted that decoupling the effect of lighting environment and BRDF from exemplar images is sometimes ill-posed and intrinsically ambiguous (for example, it is to tell apart the effects of area light sources and large specular lobes). In our approach, which captures the combined effect of BRDF and the environment, that ambiguity needs not be solved in either capture or rendering process.

The existing technique most similar to the one presented here is generally known in the field as MatCap, a shorthand for "Material Capture" widely used in sculpting software and based on the concept of LitSpheres. The main idea is that they convey material appearance into a single image of a sphere, which can be easily transferred to an individual 3D object [SMGG01] and more recently [BG07]). This approach has been furtherly extended to allow some more flexibility in use by [ZMB*15]. We will shortly describe them in the next section, before going on describing the presented data structure in Sec. 4, the capture process in Sec. 5, and the rendering process in Sec. 6.

3. Single Image Acquisition of Material-Environment

For sake of clarity, this section describes a straightforward and well understood technique which can be used, in some contexts, to quickly acquire the optical behavior of a real-world isotropic BRDF material inside a pre-determined lighting environment, and then to recreate these elements over arbitrary virtual objects. This technique goes under many names, as “lit sphere approach”, and

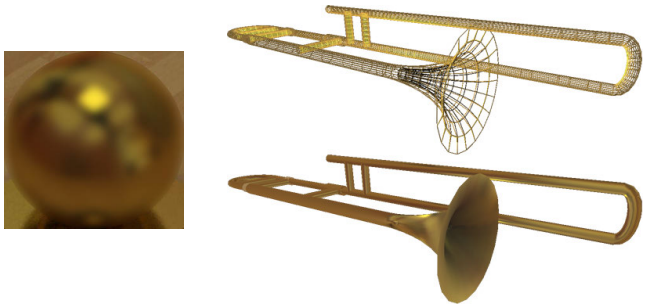


Figure 1: *Single Image Acquisition: a picture of a brass ball is used to light a model of a trombone, using the Single Image Acquisition technique.*

exploit a commonly available hardware feature called “spherical environment mapping”; in OpenGL 2.1 [SA06], it was included in the Fixed Pipeline as a texture generation mode[†]. This idea of storing and using the result of the interaction between lighting environment and material has been explicitly exploited in pre-filtered environment maps [KVHS00] that use a lighting environment preconvolved by a material and stored in a spherical map. In this paper we will refer to this techniques as “Single Image Acquisition” (of material property and lighting environments). The main innovative aspect of AMC here proposed, w.r.t. classical Single Image Acquisition is that it extends its applicability to anisotropic materials.

In Single Image Acquisition, a physical sample of the targeted material must be used, shaped as a sphere (or a semi-sphere). Spherical shaped samples are used because a single image provides an exemplar for each possible front-facing normal orientation in view space. The sample is placed under a targeted lighting environment, and a picture is shot at it. The resulting image can then be used directly as a 2D table recording, for any front-facing normal direction (in screen space), a pre-lighted RGB color value. For the purpose of lighting, the view direction $\overrightarrow{d_e}$ is approximated as constant over the scene (this makes very little practical difference).

Specifically, values in the table T represent the convolution of the lighting environment E with the material Bidirectional Radiance Distribution Function (BRDF) $\mathcal{F}$:

$$T_{i,j} = \int_{d\in\Omega} \mathcal{F}_{n_{i,j}}(d,d_e)\mathcal{E}(d)(d\dot{n}_{i,j}) \qquad (1)$$

where $n_{i,j}$ represents the normal corresponding to table entry (i,j), the lighting environment $\mathcal{E}(v)$ is a function returns the light coming from view-space direction v, and $\mathcal{F}_n$ represents the material isotropic BRDF, reoriented according to normal direction $\mathcal{F}_n$. Function $\mathcal{F}_n(v_0,v_1)$ returns the distribution of light coming from direction v_0 which bounces back at direction v_1, by the surface oriented in direction n). Neither $\mathcal{E}$ nor $\mathcal{F}$ are separately captured by T, but rather their convolution is.

Rendering is as trivial as acquisition: table T is stored as a standard RGB texture, and accessed per fragment using the n_x

[†] but remapping from $[-1+1]$ to $[0,1]$ must be added with a separate texture coordinate transforamtion matrix

and n_y components of the current screen-space normal as coordinates (after a linear transformation to remap interval $[-1..+1]$ into $[0..+1]$). The fetched value is copied, as is, on the screen buffer, thus replacing any lighting computation with a simple texture access.

This simple technique should not be confused with Spherical Environment Mapping, which is targeted at capturing solely an environment E, not material properties. Pure mirroring spheres are used, so that $\mathcal{F}$ in (1) can be assumed to be a pure mirroring function and the values in the table will represent the values of E, which are to be plugged in the chosen lighting equations.

Single Image Technique presents strong limitations but offer several advantages (see Sec. 8 for a discussion). This makes it a *potentially* a very attractive choice in these applicative scenarios where the limitations are acceptable or irrelevant (see Sec. 8.1 for a discussion of such scenarios).

However, even in these applicative scenarios where this technique *would* be an ideal solution, its usability is hindered because it can only be used with materials with isotropic BRDF. The proposed technique, termed "Anisotropic MatCap", removes that limitation, unlocking the use it for important classes of materials which include brushed metals, and, more importantly, textiles (for example allowing cloth and garment related applications).

4. Anisotropic MatCap

We propose Anisotropic MatCap (AMC) a compact data structure describing a given anisotropic material in a given lighting environment, the latter being expressed in view space.

An AMC consists of a 3D table, sized $S_i \times S_j \times S_k$, storing, for each surface tangent orientation $\overrightarrow{t}$ and each front-facing normal orientation $\overrightarrow{n}$, a pre-lit RGB color. Both $\overrightarrow{t}$ and $\overrightarrow{n}$ are expressed in view space, so $n_z >= 0$. Specifically, the RGB values for a normal direction $\overrightarrow{n}$ and a tangent direction $\overrightarrow{t}$ is stored at coordinates $(i',j',k') \in \{0..S_i-1\} \times \{0..S_j-1\} \times \{0..S_k-1\}$:

$$\begin{pmatrix} i' \\ j' \\ k' \end{pmatrix} = \begin{pmatrix} \lfloor i \cdot (S_i-1) \rceil \\ \lfloor j \cdot (S_j-1) \rceil \\ \lfloor k \cdot (S_k-1) \rceil \end{pmatrix} \tag{2}$$

with

$$\begin{pmatrix} i \\ j \\ k \end{pmatrix} = \begin{pmatrix} (n_x+1)/2 \\ (n_y+1)/2 \\ atan2(t_y,t_x)/2\pi \end{pmatrix} \tag{3}$$

Since $\overrightarrow{n}$ is a unit vector values of i and j such that $(n_x^2+n_y^2>1)$ are never used. The part of the table that is actually used is a cylinder with the axis oriented in the k direction. This means that $4-\pi/4 \simeq 0.21$ of the space occupied by the table is wasted, which we consider acceptable (like what happens for classical environment mapping where the same exact portion of texture space is actually wasted).

Equations (2) and (3) signifies that AMC is a regular sampling of the (i,j,k) values defined in (3), $(i,j,k) \in [0,1] \times [0,1] \times [0,1)$. Normals are regularly sampled after projecting them onto the screen plane. This sampling is clearly not well distributed in 3D normal space, but this is nevertheless ideal as the sampling density is proportional the the screen area covered by a surface element having that normal. Conversely tangent directions are sampled at regular angles.

4.1. Choosing AMC Sizes

To maximize compatibility with texture formats (Sec. 6), the sizes of the table in each direction, S_i, S_j, S_k are chosen as powers of 2. Clearly, the normal should be sampled with the same density in the x and y directions, so $S_i = S_j$. To sample tangent directions with roughly the same density, S_k should be at least twice than S_i, and at most four times as much. Ideally $S_k = \pi \cdot S_j$, the sampling would be the same for normals and tangents in the i,j around $1/2$, where the normal is sampled more densely (which is also where denser normal sampling is needed).

While a 3D table can look space consuming at first, we found that even little sized AMC tables produce very good results. In our examples, we used $S_i = S_j = S_k/2 = 128$, or 256, resulting in a reasonable total of $12MB$, or $48MB$, uncompressed data (assuming to store colors in 24 bits).

4.2. Exploiting Rotational Invariances

The BRDF of many real-world anisotropic materials is invariant to a rotation of π, or even of $\pi/2$, around the normal. If so, the table can sample the k axis only in the $[0,1/2)$, or in $[0,1/4)$ interval, respectively, saving space (in general, if the BRDF is invariant to rotations of $2\pi/R$, the k axis can be sampled in $[0,1/R)$).

These rotational invariances are determined by the stochastic invariances of the microstructures which are physically responsible for the perceived BRDF. For example, most woven fabrics exhibit at least a π rotational invariance: the anisotropy of the BRDF is due to the microstructure of the warp and weft strings, but inverting the direction of both warp and weft does not change the visual result. If the waving pattern is such that *swapping* warp with weft does not change the overall behavior, a $\pi/2$ invariance emerges, although that is more seldom the case. Most brushed metals (and all vinyl disks type surface), and many other anisotropic materials, also exhibit a π invariance.

When π invariances are exploited, we use cube-shaped AMC ($S_i = S_j = S_k$), halving the above space occupancies to 6 and $24MB$ respectively.

5. Capturing Anisotropic MatCap

An AMC can be captured by a sequence of pictures shot at an uniform sample of the material of interest covering a (semi-)spherical support, embedded in the chosen lighting environment. At each picture, the support is rotated around the axis connecting the camera and the center of the sphere, leaving the camera position and orientation fixed w.r.t. the lighting environment.

The acquisition process is straightforward, and can be easily automatized. In fact we constructed a fully working simple prototype be constructed by means of a servo assisted rotating support and an off-the-self web cam (see Fig. 2).

To minimize the effect of perspective distortion, the images should be captured from a distance, using long focal lengths. Alternatively, shorter but known focal lengths can be easily corrected for in the resulting images. The part of the image featuring the sphere must be cropped and downsampled to the required resolution $S_i \times S_i$. Very low resolution images are needed (e.g. $S_i = 128$), so even low resolution camera (e.g. webcams) can be employed.

Each of the S_k resulting image is directly used as a slice of the AMC. Care must be taken to rotate the sample by exactly $2\pi/(S_k)$ radiant after each picture, so that at the end of the process it is in the same position again (luckily $1/2^i$ fractions of a circle are easy to achieve by combining only a few gears).

5.1. Capturing the Spherical Sample

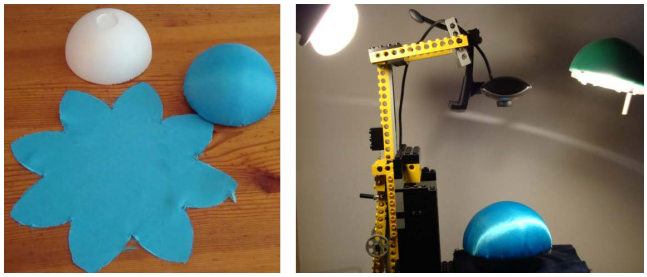


Figure 2: *Left: covering a sperical sample with a textile. Right: a simple prototype capable of acquiring AMC. The sample is exposed to a lighting environment of choice (the lamps) and rotated while a webcam captures the images. Lighting environment is fixed w.r.t the camera.*

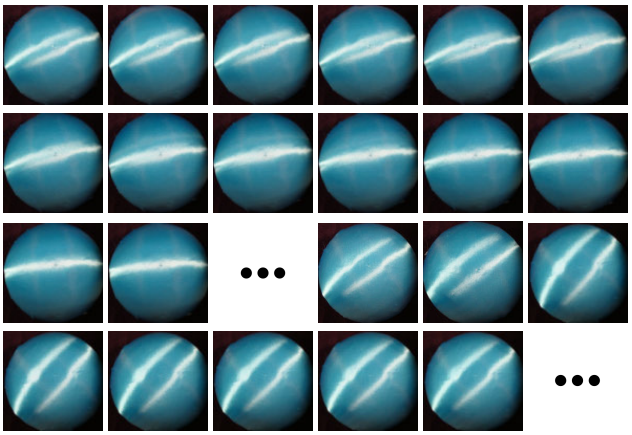


Figure 3: *A subset of the 128 128x128 images obtained during a capture section for a satin textile (exploiting rotational invariance with $R = 2$, see Sec. 4.2).*

The sample must be prepared so that the tangent directions in image-space, for image k, are constant, and consistent with (3). In case of a brushed metal sphere, this means that the brushing must happen along parallel directions around the sphere (like parallels line in a globe), and then the sphere bust be rotated during acquisition around the line connecting two opposite points at the equator). In case of textiles, the problem arises about how to envelope the sphere with a initially flat textile. Folds and cuts are unavoidable. They can be detected in the first image, by marking a few pixels as invalid, and than trivially tracked during the rotation. Marked pixels are then covered expanding neighbors over them.

A good balance can be easily found between having ideal tangent directions, minimizing stretch (most garment are partially elastic, but stretching them affects the BRDF), and minimizing cuts (see Fig. 2).

5.2. Detecting Rotational Invariances

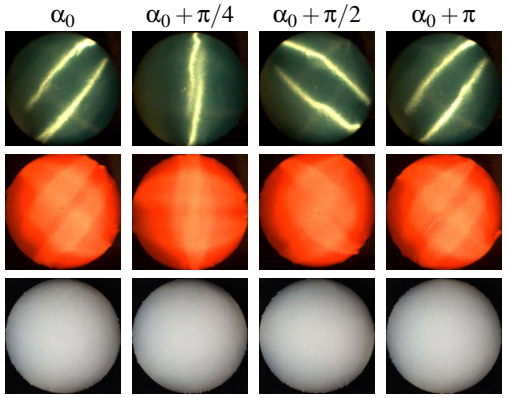


Figure 4: *Searching for rotational invariance (of multiple of $2\pi/R$) over three different captured AMC. Top: a satin exhibiting perfect $R = 2$ invariance (first image similar to last). Middle: a silk exhibiting a usable $R = 4$ invariance (first image similar to last two). Bottom: a cotton cloth is (almost) invariant for any rotation, therefore revealing as isotropic.*

By analyzing the resulting dataset, $2\pi/R$ rotational invariances (Sec. 4.2) can be easily detected, e.g. testing per-pixel image similarities, for $R \in 1..4$. If the similarity is above a user selected threshold, then a R rotational invariance is detected, and only the first $1/R$ images are used. The rest of the images can then be averaged per-pixel with the ones they replace, so to reduce noise and effectively strictly enforce the detected invariance in the resulting dataset. Note that the presence of rotational invariance dependents, in principle, both from the material and from the lighting environment.

According to expectations, of our examples made from all woven fabrics, all showed an almost exact invariance for $R = 2$; a few, showed also it for $R = 4$, but to a lesser degree (see Fig. 4).

5.3. Enforcing Isotropy

The intended use of a captured AMC is to reproduce the anisotropy of the target material in renderings (Sec. 6). As a secondary application an AMC can be used to construct a standard MatCap which is, by construction, isotropic (or at least behaves isotropically under the chosen light conditions), and yet it is similar in appearance to the captured one, under any other aspects (e.g. spotlight brightness,

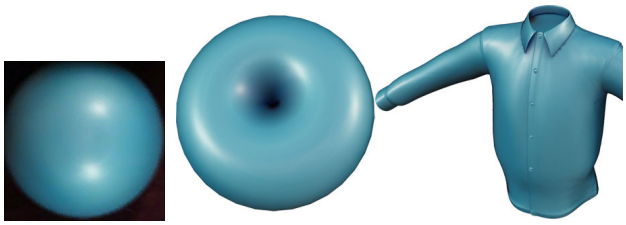


Figure 5: *The averaged image for the AMC composed of the images shown in Fig. 3, and a few renderings performed with it using it as pre-lit single-image standard MatCap (compare with Fig. 10).*

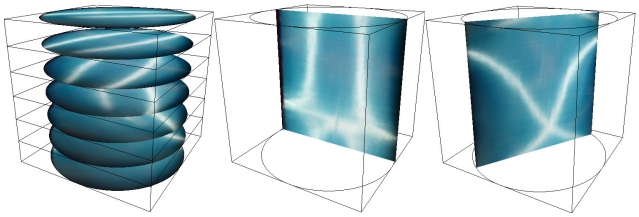


Figure 6: *Left: each captured image (see Fig. 3) is a layer of the AMC , which is stored as a 3D texture (only a few of the 128 slices are shown). Middle, right: orthogonal slices of the dataset.*

colors, reflection colors, etc). To do so, we simply average, in color space, all captured slices and obtain a single image (see Fig. 5). This makes for a good isotropic substitute for the captured material in any context where an AMC cannot be afforded (for example, due to lack of HW support for 3D textures in embedded devices, or in existing rendering engines).

6. Rendering with Anisotropic MatCap

Before the rendering process, the AMC is loaded in texture memory as a 3D texture (see Fig. 6).

Object-space normals $\overrightarrow{n_o}$ and tangent directions $\overrightarrow{t_o}$ must be fed for each vertex into the rendering pipeline. Tangent directions for a given model can be precomputed or designed in a variety of ways [VCD*16], or just procedurally. In our examples, they have been produced simply from the the UV-map associated to the meshes (see Fig 10, bottom left). Specifically, ∇U and ∇V are computed per triangle and averaged at vertices. In our shirt examples, the UV-map approximately reflects the physical production process of the garment, therefore the tangent directions reflect the actual warp-and-weft orientation of the physical object.

In the vertex processor, the corresponding view-space vectors $\overrightarrow{n}$ and $\overrightarrow{t}$ are found by projection, as usual (tangents are multiplied with model-view matrix, and normals with the "normal-matrix", i.e. the transposed inverse of the model-view matrix). Unless an orthogonal view is used, perspective correction for both vectors can be taken in account, after re-normalization, using the projected position p in normalized clip coordinates ($p \in [-1..+1]^3$ for a point inside the view frustum), by:

$$n_x \leftarrow \frac{n_x - \frac{p_x}{p_z} n_z}{\sqrt{1+(\frac{p_x}{p_z})^2}} \qquad n_y \leftarrow \frac{n_y - \frac{p_y}{p_z} n_z}{\sqrt{1+(\frac{p_y}{p_z})^2}} \tag{4}$$

(and likewise for $\overrightarrow{t}$). Then, Equation (3) is computed from $\overrightarrow{n}$ and $\overrightarrow{t}$ and the resulting values (i, j, k) are sent to the rasterizer.

Then the fragment processor fetches with trilinear interpolation the corresponding values from the texture, using the interpolated (i, j, k) as texture coordinates. Since k coordinate is intended as invariant to translation of integer numbers, the texture wrapping mechanism ($k \leftarrow k - \lfloor k \rfloor$) is enabled for that direction (however, see Sec. 6.2 later for an important caveat). If a $2\pi/R$ rotational invariance has been exploited during the construction of the AMC (Sec. 4.2), the k value is multiplied by R before being used to access the 3D texture.

For more accurate results, we can make the the vertex processor send per-vertex $\overrightarrow{n}$ to the rasterizer, and push the computation of values i, j form $\overrightarrow{n}$ down to the fragment processor, so that the per-fragment $\overrightarrow{n}$ can be re-normalized after interpolation and before being plugged into (3). Conversely, the relatively more time consuming computation of k from $\overrightarrow{t}$ is independent from the magnitude of $\overrightarrow{t}$, so $\overrightarrow{t}$ it needs no normalization. Actually, letting the rasterizer interpolate k instead of $\overrightarrow{t}$ is not only more efficient but also more accurate, as it results in a even sampling in angular distance.

6.1. Approximating Shadowing Effects

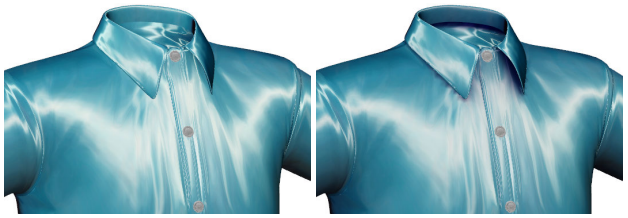


Figure 7: *The effect of the per-fragment darkening of the pre-lit AMC values with a precomputed per vertex AO factors. Left: with. Right: without. Even though this is just an approximation, plausible results are obtained.*

While the method is fully sound only as a local illumination model, basic shadowing effects can be added, or approximated, at an very cheap computational cost and with acceptable results.

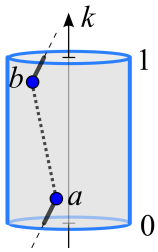
If the lighting environment is dominated by a single point light source, then two separate AMCs can be used, captured with and without that light source. At rendering time, we can determine whether the current fragments are directly lit by that light source, using a standard shadow computation technique (as Shadow Mapping or Shadow Volumes), and access either one of the AMCs accordingly.

Ambient Occlusion shadowing can be also approximated, to some extent: pre-computed per-vertex Ambient Occlusion factors $a_v \in [0..1]$ can be sent for each vertex, interpolated, and used to multiply the final color. This is, strictly speaking, uncorrect, as AO applies to Lambertian surfaces under uniform lighting environments, but the results are visually convincing see Fig.7.

6.2. Correct Interpolation of Cyclic Coordinates

As mentioned, AMC volumetric texture cylinders are by construction "tileable" on the k direction, i.e. the top base of the cylinder is considered as matching the bottom base, as the AMC represents the period of a function that is cyclic in k. The hard-wired texture fetch mechanism can be instructed accordingly, so that the access at a given k is performed using, as coordinate: $(k - \lfloor k \rfloor)$. In what follows the notation $k \simeq (k+i)$ expresses that using coordinate k is equivalent to use $(k+i)$, $\forall i \in \mathbb{Z}$.

The described rendering algorithm suffers from an interpolation problem: values k produced by the vertex program, i.e. the vertical positions on the cylinders, will be occasionally interpolated by the rasterizer in the "wrong direction" (see Fig. 9).



The problem arises every time a triangle is rasterizer which connects two vertices a and b with two values k_a and k_b which are close respectively to the bottom and the top face of the volumetric texture cylinder, or vice versa. Specifically, let $k_a = 0 + \delta_a$ and $k_b = 1 - \delta_b$, with small positive values of $\delta_{a,b}$ such that $\delta_a + \delta_b < 1/2$ (see figure). The interpolation should take advantage of the cyclic nature of the k axis, and correct interpolation has to be performed along the "short route" on the k axis (thick line in the figure), which is as long as $\delta_a + \delta_b$ and passes through value $k_i = 0 \equiv 1$. On the contrary, the linear interpolation wrongly takes the "long route" on the k axis (dotted in the figure), which is long $1 - (\delta_a + \delta_b)$ are passes through the midpoint $k_i = 0.5$.

This problem is closely analogous to the commonly encountered one occurring when a standard tileable 2D texture is used over a closed mesh. This problem has been faced in a more complete way in [Tar12] and here we offer, for sake of comprehension, a short description of that technique for this specific case.

Consider a statically uv-mapped triangular mesh modeling the side area of a cylinder, which is textured with an image, tileable in the horizontal u direction and wrapped around the cylinder, so that the two matching vertical sides of the image meet over a "cutting line" on the cylinder surface. Any mesh triangle spanning across the cutting line will interpolate small and large values of the u coordinate, thus wrongly passing through middle texture values, which should be found only on the other side of the cylinder. Note that defining u values outside $[0,1)$ for vertices of that triangle solves the problem for that triangle, but creates a similar one for the other triangles using that vertex; also, the problem persists even when no triangle is allowed to cross the cutting line, i.e. if that line is carefully tessellated with mesh edges: in that configuration, a vertex on the line must still have, at the same time, a 0 u value, for a triangle connecting it to vertices on a side of the line, but a u value of 1 for another triangle connected on the other side.

In the statically uv-mapped mesh case, the commonplace solution consists in replicating a few vertices and assigning different u coordinate to each copy (i.e. introducing seams); this solution, however common, is inelegant as the redundancy can be harmful not so much as for space occupancy but also for data structure coherence (e.g. imagine a scenario when the mesh must be animated with a mass and spring system). In the AMC case, that solution is even more problematic, as texture coords are not known a priori but generated dynamically with (3), and thus the vertex should be dynamically replicated in the pipeline.

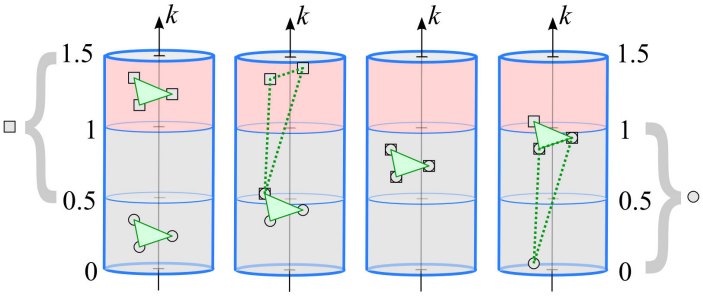


Figure 8: *The solution to the interpolation cyclic texture coordinates. The vertex program always produce both values two values k and k' for the k coordinate, with $k \in [0,1)$, $k' \in [0.5, 1.5)$ and $k \equiv k'$. Here, texture points with k are represented by a circle, and with k' by a square. When any triangle is rasterized which can span less than 0.5 in the k direction, the interpolations of at last one of k and k' yields to the correct result (the full green triangle), specifically the one which takes the "shortest route" taking advantage of the cyclicity of k. The right alternative is identified per fragment as the one among k and k' which presents the smallest absolute derivative. Here, all possible cases are shown. In case 1 and 3, the derivatives are equals, and either one can be safely chosen (as $k = k'$, in case 3, or at least $k \equiv k'$, in case 1). In cases 2 and 4, the discarded interpolated values (of k' and k respectively) are shown as a dotted triangle. In the fragment program, thanks to texture wrapping mechanism, accesses at the reddish coloured part of the texture space, on top, will be actually result in an access in the lower half of the grayed area. Case 4 is the one that causes the artefact (see Fig. 9) if this technique is not employed.*

First, consider what happens if the vertex processor sends, for each vertex, value $k' = k - \lfloor k - 0.5 \rfloor$, instead of k, so that $k' \equiv k$, but while $k \in [0,1)$, $k' \in [0.5, 1.5)$, (see Fig. 8). Using k' the triangles that formerly arose the problem are interpolated correctly. However, the problem now appears on any triangle connecting two vertices a and b with $k_a = 0.5 + \delta_a$ and $k_b = 0.5 - \delta_b$. However, assuming any triangle always spans less that 0.5 in k texture direction (a very reasonable assumption in both scenarios), either k or k', will produce the correct interpolated values for any triangle. In our solution, the vertex processor fist computes k (always $\in [0,1)$) as in (3), then computes k' from k and sends them both down the pipeline to the rasterizer. In the fragment program, the correct value is picked among interpolated values k'_i and k_i as the one presenting

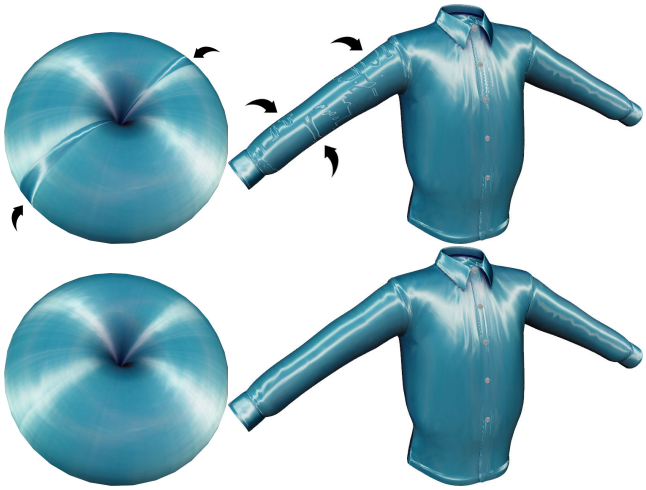


Figure 9: *Top: interpolation artifacts (pointed by arrows) arising from incorrectly interpolation of the cyclic variable k of the AMC, in renderings of two models. Bottom: artefacts corrected (Sec. 6.2).*

the smallest derivative in absolute value (most of the times, it will not make any difference as either $k'_i = k_i$ or $k'_i = k_i + 1$, see Fig. 8). Note that the derivative of a pure *varying* value, as k and k', is constant inside the triangle and is usually computed in closed form by the rasterizer (rather than by finite differences) and automatically passed to the fragment processor.

In the statically uv-mapped scenario stated above, this technique, applied at the u coordinate, does not cost an extra attribute to be sent either, as u' is computed in the vertex program from value u (normally passed as a vertex attribute). Note that this system works even if the tileable texture must be replicated $R > 1$ times over the mesh, before connecting to itself of the cylinder surface. In this (more general) case, value u' is found as $u - \lfloor uR - 0.5 \rfloor / R$ instead.

7. Results

We tested the AMC technique by acquiring a number of textiles and rendering a number of syntectic or real-world objects, including garments (Fig. 10, 9, 11). While the technique can be applied to any anisotropic BRDF, garments offer a good test-bed. In fact, the use of anisotropic material is most evident in presence of discontinuity of the tangent directions, which can result in discontinuities in the lighting. Garments often exhibit this effect where pieces of fabrics are sewed together with different direction of warp and weft. Replicating this effect is quite important in garment design(see Fig. 11). Interestingly enough in meshes representing stitched garments it is quite convenient that the parametrization mimics the pattern used to cut from the fabric roll the actual pieces of the garment; in this case the tangent directions ocomputed from UV-mappings directly corresponds to the anisotropy directions needed to replicate the correct behaviour of the garment material.

8. Discussion

The proposed AMC technique offers several advantages:

- independence from material BRDF complexity;
- independence from lighting environment complexity;
- faithful, direct, and easy capture from reality;
- negligible rendering time overhead;
- straightforward implementation;
- ideal sampling density of normals in the table (more samples devoted to normals more orthogonal to the view direction, and less at normal seen from grazing angles);
- the need for HDR measurement in BRDF and the lighting environments is strongly diminished, because images capturing their combined effect already includes a sort of tone-mapping;
- capture and storage of combined effect of material characteristics and lighting environment in one go.

The last point also represents a drawback: it will not be possible to relight the same material under novel lighting environments. Clearly there are several other limitations:

- local lighting only is recreated; global lighting effects (for example, Ambient Occlusion), can only be approximated (Sec. 6.1);
- lighting environment E must be assumed to stem from distant illuminators;
- likewise, for illumination purposes, the view direction is approximated as constant over the scene;
- since E is expressed in view space, the point of view cannot be changed with respect to the environment - in other words, it is possible to dynamically rotate the object in front of the camera, but not to rotate the camera around the object (keeping it fixed with respect of the light sources).

All the advantages and disadvantages in the above lists are inherited from the standard Single Image MatCap (Sec. 3), except the first item, as the latter is limited to isotropic BRDFs. Removing this limitations, Anisotropic MatCap become ideal candidates in all these applications scenarios where the limitations described above are acceptable (see below), and the need arises to reproduce, over arbitrary virtual objects, the appearance of given real-world materials, under given real-world lighting environments.

The size of a AMC is very manageable. For example, a single $2K^2$ texture, commonplace for example in current games, has twice the number of texels, and the occupancy, of the 128^3 AMC we use in most of our examples. In our tests on commodity computers, the performance impact of activating rendering with AMC proven to be negligible, the application running anyway at full FpS even with the most complex meshes we tested (the shirt of Fig. 11).

8.1. Application Scenarios

Some scenarios, like most games or virtual reality, are not suitable for it as it would be impossible to dynamically change the light environment or to allow navigation of the camera around the object while keeping the lighting environment fixed in world-space (instead of screen-space).

Conversely, AMC represent a very attractive way to compute lighting (or, rather, to bypass that computation) when the lighting environment is constant and fixed with respect to the viewer: for example, that is the case when the scene consists of a virtual 3D object which is to be visualized by the end user, who, in order to

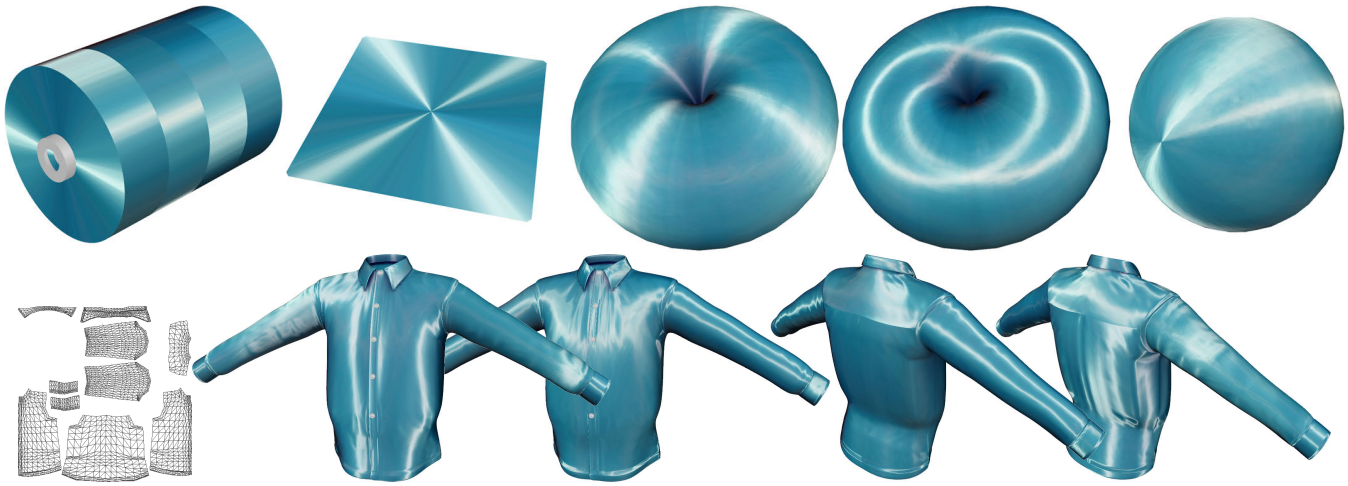


Figure 10: *Examples rendering obtained with a AMC modelling a satin under spotlights (see Fig. 3). Above: AMC applied over simple procedural models (the two torus models differs only for the tangent dirs). Below: application to a 3D satin shirt model (tangent directions defined by the UV map, on the left). Tangent direction are rotated by $\pi/2$, thus inverting in the fabric, the directions of warp and waft.*

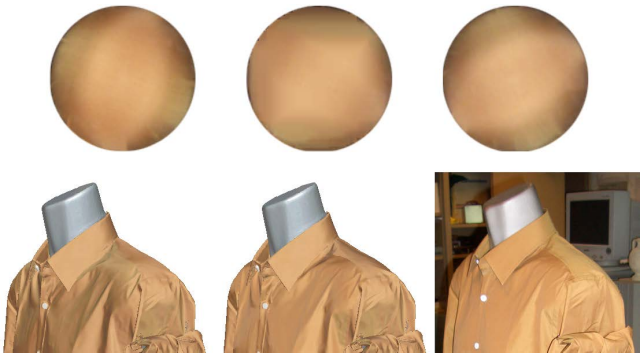


Figure 11: *A range-scanned 3D model of a shirt, rendered with with Single Image Standard MatCap (left) and with an 128^3 AMC (middle), both captured at a sample of the same textile. Right: real picture of the shirt, shot, for comparison, from roughly the same point of view, but under a different lighting environment than the one that was used to capture the materials. Tangent directions on the garment 3D model are only approximated. Notwithstanding these discrepancies, the image rendered with the AMC replicates faithfully the lighting discontinuity which appears at the sewing line on the shoulder in the real image, while this lightning feature is missing in the Single Image MatCap, which is unable to capture material anisotropicity. Top: three slices of the AMC.*

inspect it, must be able to freely rotate it, e.g. by means of a trackball (like in e-commerce, cultural heritage applications, in medical applications, etc). In all these contexts AMC are very effective at producing very realistic results faithfully embedding in virtual 3D scenes both real world materials and real world lighting environment. In such cases it should be preferred to more complex alternatives, like measuring the BRDF and the lighting environment and then recompute (1) dynamically at rendering times.

References

[BG07] BRUCKNER S., GRÖLLER M. E.: Style transfer functions for illustrative volume rendering. *Computer Graphics Forum 26*, 3 (Sept. 2007), 715–724.

[CCC08] CORSINI M., CALLIERI M., CIGNONI P.: Stereo light probe. *Computer Graphics Forum 27*, 2 (2008), 291–300.

[KVHS00] KAUTZ J., VÁZQUEZ P.-P., HEIDRICH W., SEIDEL H.-P.: A unified approach to prefiltered environment maps. In *Rendering Techniques 2000*. Springer, 2000, pp. 185–196.

[LGM07] LENSCH H., GÖSELE M., MÜLLER G.: Capturing reflectance - from theory to practice. Tutorial Notes of Eurographics, Sept. 2007.

[MPBM03] MATUSIK W., PFISTER H., BRAND M., MCMILLAN L.: Efficient isotropic brdf measurement. In *EGRW '03: Proceedings of the 14th Eurographics workshop on Rendering* (2003), pp. 241–247.

[RH01] RAMAMOORTHI R., HANRAHAN P.: A signal-processing framework for inverse rendering. In *SIGGRAPH '01* (New York, NY, USA, 2001), ACM, pp. 117–128.

[RH02] RAMAMOORTHI R., HANRAHAN P.: Frequency space environment map rendering. *ACM Trans. Graph. 21*, 3 (2002), 517–526.

[SA06] SEGAL M., AKELEY K.: *The OpenGL Graphics System: A Specification (2.1)*. Tech. rep., Silicon Graphics Inc., December 2006.

[SMGG01] SLOAN P.-P. J., MARTIN W., GOOCH A., GOOCH B.: The lit sphere: a model for capturing npr shading from art. In *Graphics interface 2001* (2001), pp. 143–150.

[Tar12] TARINI M.: Cylindrical and toroidal parameterizations without vertex seams. *Journal of Graphics Tools 16*, 3 (jul 2012), 144–150.

[UWH*03] UNGER J., WENGER A., HAWKINS T., GARDNER A., DEBEVEC P.: Capturing and rendering with incident light fields. In *EGRW '03: Proceedings of the 14th Eurographics workshop on Rendering* (2003), pp. 141–149.

[VCD*16] VAXMAN A., CAMPEN M., DIAMANTI O., PANOZZO D., BOMMES D., HILDEBRANDT K., BEN-CHEN M.: Directional field synthesis, design, and processing. In *Computer Graphics Forum* (2016), vol. 35, Wiley Online Library, pp. 545–572.

[ZMB*15] ZUBIAGA C. J., MUÑOZ A., BELCOUR L., BOSCH C., BARLA P.: MatCap Decomposition for Dynamic Appearance Manipulation. In *Eurographics Symp. on Rendering 2015* (Darmstadt, Germany, June 2015).