



# Color Rendering in Medical Extended-Reality Applications

Andrea Seung Kim<sup>1,2,3</sup> · Wei-Chung Cheng<sup>1,2,3</sup> · Ryan Beams<sup>1,2,3</sup> · Aldo Badano<sup>1,2,3</sup>

Received: 19 February 2020 / Revised: 15 September 2020 / Accepted: 30 September 2020 / Published online: 17 November 2020  
© Society for Imaging Informatics in Medicine 2020

## Abstract

Cross-platform development of medical applications in extended-reality (XR) head-mounted displays (HMDs) often relies on game engines with rendering capabilities currently not standardized in the context of medical visualizations. Many aspects of the visualization pipeline including the characterization of color have yet to be consistently defined across rendering models and platforms. We examined the transfer of color properties from digital objects, through the rendering and image processing steps, to the RGB values sent to the display device. Five rendering pipeline configurations within the Unity engine were evaluated using 24 digital color patches. In the second experiment, the same configurations were evaluated with a tissue slide sample image. Measurements of the change in color associated with each configuration were characterized using the CIE 1976 color difference ( $\Delta E$ ). We found that the distribution of  $\Delta E$  for the first experiment ranges from zero, as in the case using an Unlit Shader, to 25.97, as in the case using default configurations. The default Unity configuration consistently returned the highest  $\Delta E$  across all 24 colors and also the largest range of color differences. In the second experiment,  $\Delta E$  ranged from 7.49 to 34.18. The Unlit configuration resulted in the highest  $\Delta E$  in three of four selected pixels in the tissue sample image. Changes in color image properties associated with texture import settings were then evaluated in a third experiment using the TG18-QC test pattern. Differences in pixel values were found in all nine of the investigated texture import settings. The findings provide an initial characterization of color transfer and a basis for future work on standardization, consistency, and optimization of color in medical XR applications.

**Keywords** Mixed reality · Virtual reality · Medical imaging · Color transfer · Graphics rendering · Image quality

## Background

Enthusiasm surrounding extended-reality (XR) technology, including augmented reality (AR) and virtual reality (VR), has prompted many opportunities and use cases in medicine. Increasing interest in medical AR and VR applications has led to regular utilization of game engines with libraries accessing input functions across multiple platforms for XR development[1, 2]. Game engines typically include rendering pipelines with numerous shading models

and lighting implementations to be explored for optimization across various hardware. In addition, common material parameters are designed to allow for efficient interpolation (i.e., varying levels of glossiness, metallic levels, specular color, anisotropy). While convenient for users wanting immediate XR visualizations of virtual scenes, these various visualization pipelines have yet to be standardized for medical use (Fig. 1).

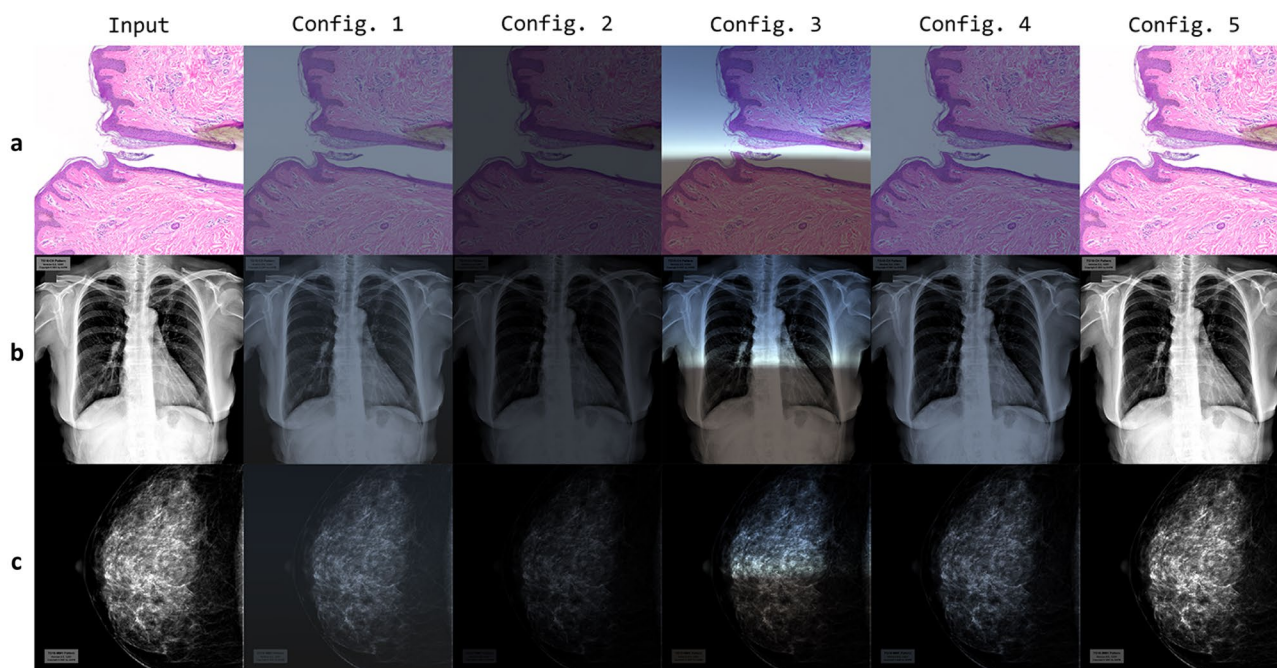
Graphics rendering algorithms are actively being explored and various versions today implement bidirectional reflectance distribution function (BRDF) models in physically based rendering (PBR) engines. Some PBR shading models used in industry include Disney's BRDF[3], Cook-Torrance BRDF[4], GGX[5], Blinn Phong[6], and Schlick models[7]. Discerning unique characteristics of the models from their names may be challenging, and various jargon terms such as main color, diffuse, and albedo hold distinct meanings in one graphics rendering engine while appearing to overlap in another. On a similar note, a setting for enabling high-dynamic

✉ Aldo Badano  
aldo.badano@fda.hhs.gov

<sup>1</sup> Division of Imaging, Diagnostics, and Software Reliability, FDA, 10903 New Hampshire Ave., Silver Spring, Maryland, USA

<sup>2</sup> Office of Science and Engineering Laboratories, FDA, 10903 New Hampshire Ave., Silver Spring, Maryland, USA

<sup>3</sup> Center for Devices and Radiological Health, FDA, 10903 New Hampshire Ave., Silver Spring, Maryland, USA



**Fig. 1** RGB input and output for five digital material, digital lighting, and digital camera configurations within the Unity engine in the rendering of cross-platform applications for selected scenes: (a) a digital

pathology image[8], (b) a digital chest radiograph, and (c) a full-field digital mammogram[9]

range (HDR) may not be consistently defined between a computer graphics engineer and a biomedical scientist. Considering 3D visualization approaches for diagnostic imaging using radiographic images including X-rays that require predictable color representation and compliance to a target luminance model (typically, the DICOM Grayscale Standard Display Function), true- and pseudo-color representations should be utilized in the rendering of such a virtual scene for head-mounted displays (HMDs).

The game engine Unity[10] is commonly used in AR and VR medical visualization applications, including simulations for pre-operative surgical and therapy planning, educational tools, diagnostic imaging, and training[11–15]. In Unity, a fragment program interpolates data from multiple vertex programs to calculate information about the surface of a model. The fragment and vertex programs are specified within Unity shaders, which run on a GPU to provide instructions on how to render a specific pixel. The pipeline in Unity was initially developed using pre-computed (baked) ad hoc rendering calculations to mimic indirect lighting. Since the 2014 release of Unity 5.0, however, Unity's Standard Shader[16] math implementation uses Disney's BRDF model proposed by Burley[3] for the diffuse component, GGX model[5] for specular, with Schlick Fresnel approximation[7] to render interactions between light sources, the view, and the surface normal mapping of objects[17]. Furthermore, Unity 5.0 incorporates global

illumination (GI), or a range of computational techniques that attempts to mimic light interacting with surface meshes present in the scene[18]. While PBR and GI are designed for robust visualizations convenient to most game developers, the anticipated effects may add uncertainty to the visualization of medical data depending on the conditions necessary for an ideal viewing environment. For example, GI and PBR calculations mimicking ambient light under a blue sky in Unity's default features using a sky material called "Skybox" may be extraneous in viewing an X-ray image. In this setting, such calculations may not only undermine the efficiency of the real-time rendering for HMDs, but also add unintended distortions in the transfer of pixel color before reaching the display. While individuals with a background in computer graphics may be well aware that shaders will produce modifications to an underlying base color or texture in order to simulate lighting and shading effects, the cause and degree of these distortions may not be obvious to medical professionals or novice XR developers.

As HMDs are commercial products already presenting with limitations in technology[19–21], medical applications for HMDs should be developed with rendering settings intentionally optimized for intended environments. In this paper, changes to color image properties associated with different rendering settings for a digital test object are characterized within a virtual Unity scene. While material properties have been referred to for a few applications,

detailed reports are not available on the characterization of color transfer considering lighting, materials, and camera configurations for a virtual scene containing medical data. The work presented here provides a framework for objective characterization of the transfer of pixel color and an initial step for color standardization and consistency in medical XR applications.

## Methods

The assessment of color transfer throughout an XR system's visualization pipeline should distinguish the software and hardware component (in this case, Unity and a computer) from the display component (display performance of an XR HMD). For example, evaluation of color distortions across different XR HMDs requires distinction from color distortions as a result of either the rendering software or the hardware specifications that are driving the final composition of the image data sent to the HMD. In this study, the software components are investigated in three experiments.

### Scene Definitions

Unity version 2019.3.0f1 was tested. Opening a new project in Unity using a 3D template creates a default scene containing a Directional Light (Position: 0, 3, 0; Rotation: 50, -30, 0) and a Camera. The test environment designed for all three experiments utilized a primitive cube with scale (1, 1, 1) placed at location (0, 0, 0) and rotation (0, 0, 0) as a digital test object, with a digital rendering camera positioned at (0, 0, 1.366) with a field of view of 60° facing the digital test object as shown in Figure 2.

Color transformation was investigated based on the initial RGB input values for the material color of the digital test object, as compared with the RGB rendered output of the side of the digital test object facing the rendering camera. The output of the display component was assumed to be within the sRGB color space. The configurations vary in material, light, and camera rendering properties as described in Table 2. Default settings for the main camera, light, and Standard Shader of the material were preserved for the initial set of measurements (configuration 1). The graphics

rendering path of the main camera was changed from default in configuration 2. In configuration 3, the "Metallic" and "Smoothness" values of the Standard Shader are adjusted to mimic a more reflective material. Configuration 4 is like configuration 3, except that light influence from skybox reflections are excluded from the rendering process. Finally, configuration 5 tests the Unlit Shader, which is not affected by lighting conditions.

In the first experiment, each of the 24 color patches from ColorChecker was rendered through the five configurations (Table 2) and then sampled as a single-frame, 1024x1024 pixel PNG image with four channels for each R, G, B color and alpha. Values in sRGB color space were input via the Albedo property found in the material's shader script for each of the 24 color patches. In the second experiment, a single-slide tissue sample obtained from multispectral imaging[8] was imported into Unity as a 2D texture file. Four pixels of varying RGB values were selected from the ground truth image (Figure 3) and then compared with the render result through the same five configurations.

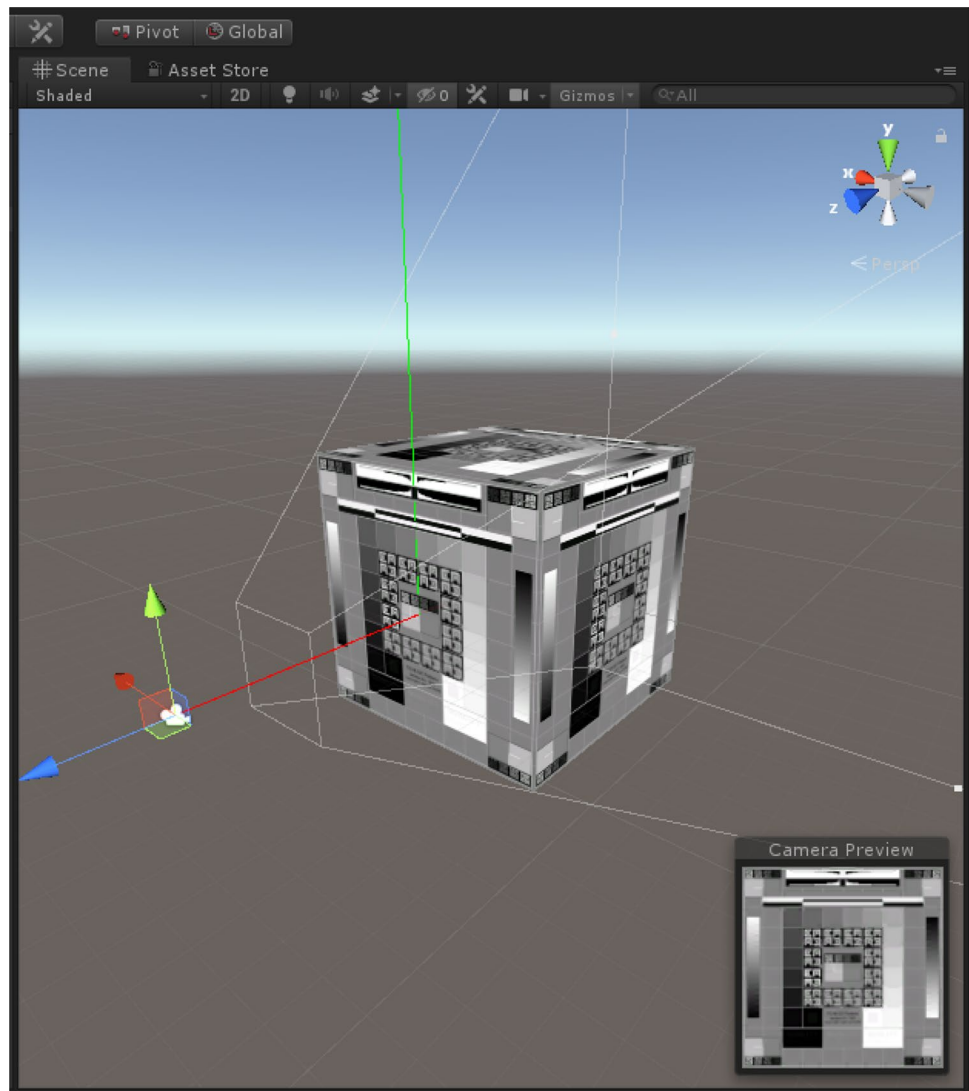
The output was analyzed for RGB values detected at each pixel of the resulting image across the five configurations. Each single-frame render output was encoded at 32-bit floats per channel. A reference white (255, 255, 255) was captured for each configuration and then applied for conversion between the sRGB and CIELAB color spaces. Project settings were kept at gamma color space in Unity, as linear color space is not yet supported by some mobile devices. Matlab and R were used for data analysis of the first two experiments.

In the third experiment, the TG18-QC test pattern[9] was imported as a 2D Texture file[22] with the nine import settings detailed in Table 2. Images were rendered using rendering configuration 5, using an Unlit Shader. Default import settings for a 2D texture file includes a Resize Algorithm, Filter Mode set to Bilinear with Aniso Level of 1, and a compression method set to "Normal Quality." These settings are found in the inspector window of the Unity editor when a texture file is selected from the Project window. Texture import settings include general as well as platform-specific parameters. Filter mode can be found under general import settings. Max size, resize algorithm, format, and compression are designated specifically for individual platforms.

**Table 1** Rendering configurations for experiment 1: ColorChecker 24 and experiment 2: Tissue WSI

Config.	Material Shader	Light Properties	Camera Rendering Path
1	Standard Shader, Metallic 0.0, Smoothness 0.5 (Default)	Default	Default
2	Standard Shader, Metallic 0.0, Smoothness 0.5 (Default)	Default	Legacy Vertex Lit
3	Standard Shader, Metallic 1.0, Smoothness 1.0	Default	Default
4	Standard Shader, Metallic 1.0, Smoothness 1.0	No Skybox Reflections	Default
5	Unlit Shader	Default	Default

**Fig. 2** Setup of the Unity scene for all three experiments



The nine images rendered from Unity and the original TG18-QC image were then imported into ImageJ for region analysis. The original image is a single channel grayscale image with 16 unsigned bits per pixel, whereas the output images are in RGB. In order to operate a difference between the two, the original image type was converted to RGB Color in ImageJ. Every pixel difference of the nine rendered images was evaluated.

### Colorimetry

For analysis of the first two experiments, the 8-bit pixel values,  $R$ ,  $G$ , and  $B$ , were converted to the CIEXYZ color space,  $X$ ,  $Y$ , and  $Z$ , using the sRGB model with the CIE D50 white point[23]:

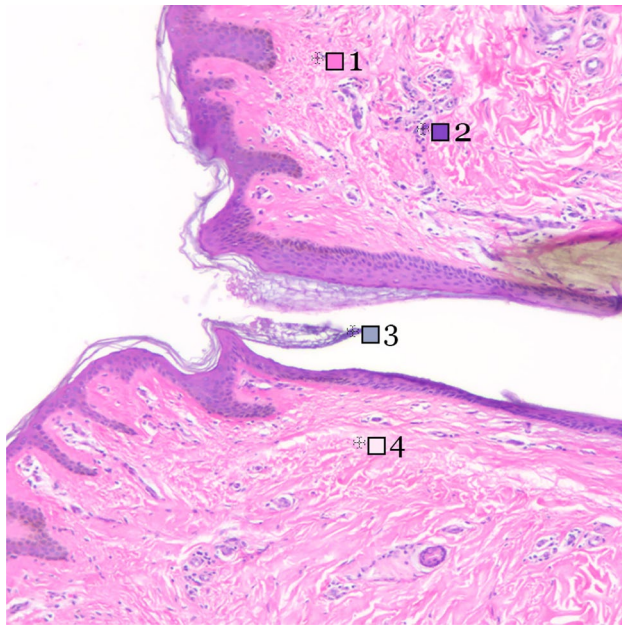
$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.4124 & 0.3576 & 0.1805 \\ 0.2126 & 0.7152 & 0.0722 \\ 0.0193 & 0.1192 & 0.9504 \end{bmatrix} \begin{bmatrix} \gamma(R/255) \\ \gamma(G/255) \\ \gamma(B/255) \end{bmatrix} \quad (1)$$

where

$$\gamma(u) = \begin{cases} 12.92u, & \text{if } u \leq 0.0031308 \\ 1.055u^{2.4} - 0.055, & \text{otherwise} \end{cases} \quad (2)$$

The CIEXYZ values of the reference white in each of the five configurations were individually determined to be  $X_n$ ,  $Y_n$ , and  $Z_n$ . For this work, the reference white was defined using the average pixel output of the white patch in each scene. The CIEXYZ values were then converted to the CIELAB color space as  $L^*$ ,  $a^*$ , and  $b^*$ :





**Fig. 3** Four pixels varying in RGB values are selected from the ground truth of a tissue sample slide obtained from multispectral imaging [8] as input RGB data

$$L^* = 116f\left(\frac{Y}{Y_n}\right) - 16 \tag{3}$$

$$a^* = 500\left(f\left(\frac{X}{X_n}\right) - f\left(\frac{Y}{Y_n}\right)\right) \tag{4}$$

$$b^* = 200\left(f\left(\frac{Y}{Y_n}\right) - f\left(\frac{Z}{Z_n}\right)\right) \tag{5}$$

where

$$f(t) = \begin{cases} \sqrt[3]{t}, & \text{if } t > \delta^3 \\ \frac{t}{3\delta^2} + \frac{4}{29}, & \text{otherwise} \end{cases} \tag{6}$$

$$\delta = \frac{6}{29}. \tag{7}$$

Finally, the difference between two colors was calculated based on the CIE 1976 color difference formula:

$$\Delta E = \sqrt{(L_2^* - L_1^*)^2 + (a_2^* - a_1^*)^2 + (b_2^* - b_1^*)^2}. \tag{8}$$

Details of the colorimetric conversion are described in Ref. [24].

The CIE 1976  $\Delta E$  uses distances in the CIELAB color space which is not necessarily uniform to human perception. However, the 1976 formula contains less assumptions on the visual models that are used and is useful for obtaining objective measurements of rendering settings.

## Results

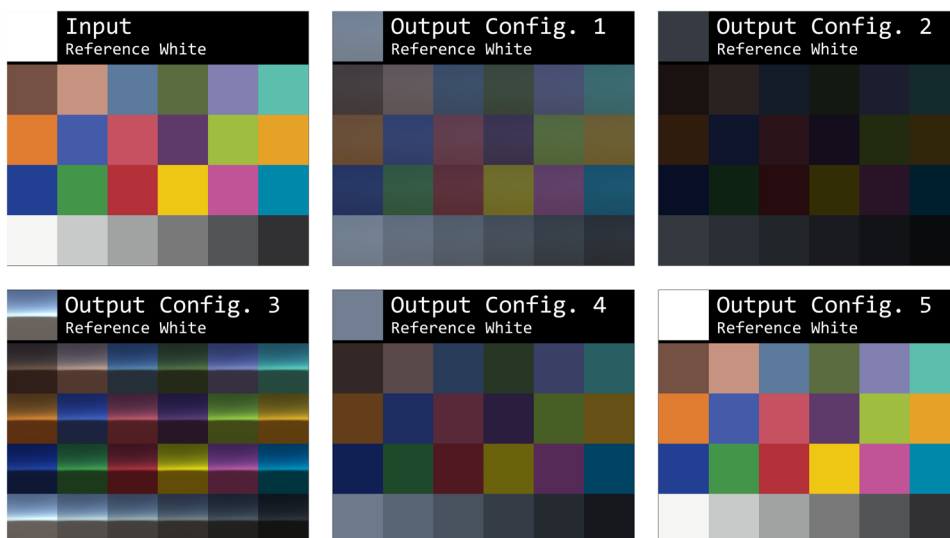
### Experiment 1: ColorChecker 24

The 24 RGB color values of the ColorChecker were entered as material color in the shader script and rendered through the five configurations as described in Table 2. Render result of the 24 color patches through the five configurations can be seen in Figure 4. The distribution of pixel value differences in  $\Delta E$  for the first experiment ranges from zero for Unlit Shader to 25.97 for default configurations. The configuration using Unlit Shader returned zero values across all 24 color patches. Excluding configuration 5 (Unlit Shader), mean render output values for all other configurations cover a range of the CIELAB color space varying from what was input through the Unity render pipeline (Fig. 5a) and show a distinct pattern. Specifically, the  $\Delta E$  values show an increasing pattern in the grayscale ramp from patches #19-24 for configurations 1-4. Configuration 1 (default) returned highest  $\Delta E$  values for all of the 24 color patches, followed by configurations 2 (legacy vertex lit rendering path), 4 (smooth, metallic material

**Table 2** Nine different texture import settings in experiment 3: TG18-QC

Setting	Filter Mode	Max Size	Resize Algorithm	Compression Format
1	Bilinear, Aniso Level 1 (Default)	2048 (Default)	Mitchell (Default)	Automatic: Normal Quality (Default)
2	Bilinear, Aniso Level 1 (Default)	512	Mitchell (Default)	Automatic: Normal Quality (Default)
3	Bilinear, Aniso Level 1 (Default)	512	Mitchell (Default)	Automatic: High Quality
4	Bilinear, Aniso Level 1 (Default)	512	Bilinear	Automatic: High Quality
5	Trilinear, Aniso Level 1	2048 (Default)	Mitchell (Default)	Automatic: Normal Quality (Default)
6	Point (no filter)	2048 (Default)	Mitchell (Default)	Automatic: Normal Quality (Default)
7	Point (no filter)	128	Mitchell (Default)	Automatic: Normal Quality (Default)
8	Point (no filter)	512	Mitchell (Default)	Automatic: Low Quality
9	Point (no filter)	512	Mitchell (Default)	Automatic: High Quality

**Fig. 4** Input and rendered output for 24 color patches of ColorChecker across five configurations described in Table 1. Reference white (255, 255, 255) can be seen on the upper left corner of each configuration



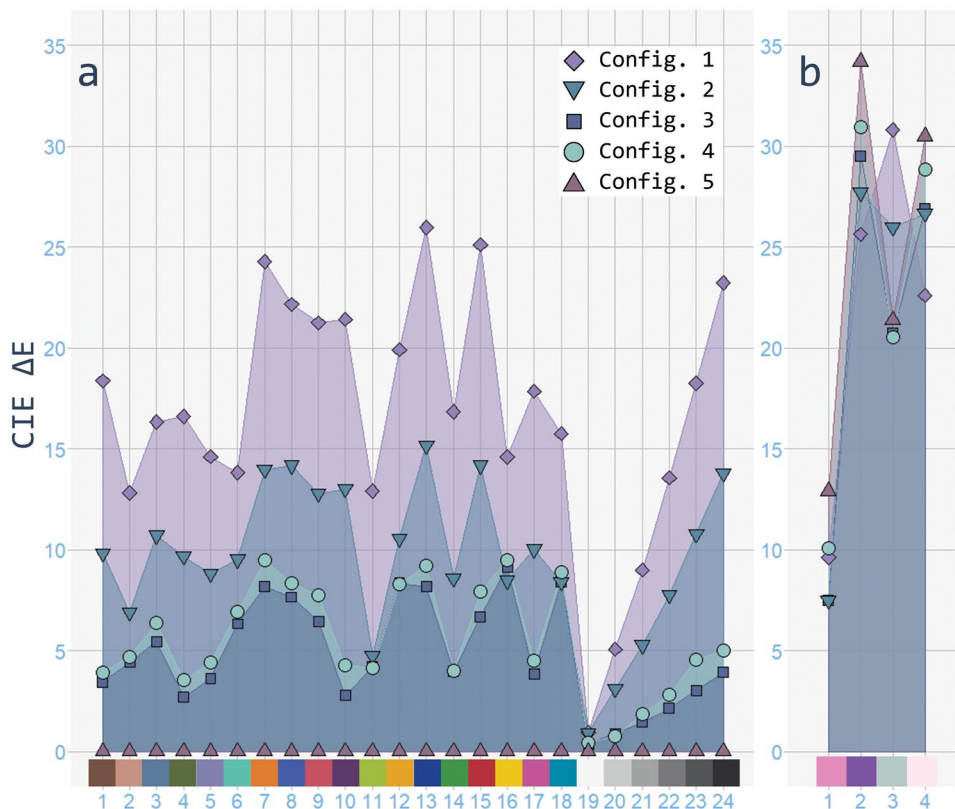
without Skybox reflections), and then 3 (smooth, metallic material). Configuration 5 returned a  $\Delta E$  value of 0 across all 24 color patches.

**Experiment 2: Tissue WSI**

The ground truth of a skin tissue slide sample obtained from multispectral whole-slide imaging (WSI)[8] was

imported into Unity as a 2D texture file. Import settings were kept at default values. The four chosen pixels varying in RGB values (Fig. 3) were then rendered through the five rendering configurations as in the first experiment.  $\Delta E$  values ranged from 7.49 to 34.18 (Fig. 5b). Unlike in the Experiment 1: ColorChecker results, the Unlit configuration in this experiment did not return zero values of  $\Delta E$ . Rather, the Unlit configuration resulted

**Fig. 5** Mean  $\Delta E$  for color patch outputs and selected pixels across the five configurations for experiments 1(a) and 2(b)



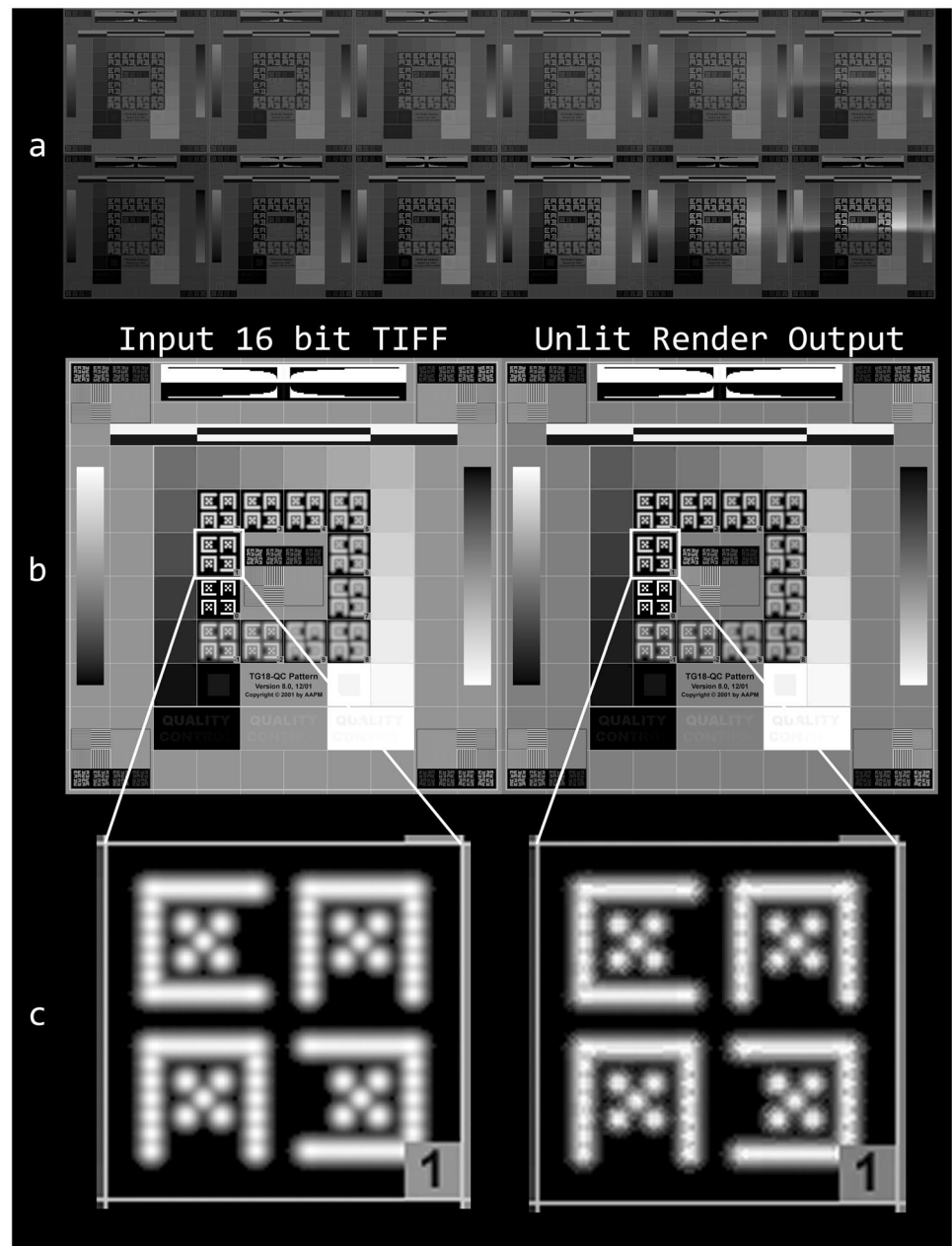
in the highest  $\Delta E$  values in three of four selected pixels in a skin tissue sample image. These results are unexpected and do not exhibit any of the patterns or increasing trends seen in the first experiment despite consistency in methods.

### Experiment 3: TG18-QC Test Pattern

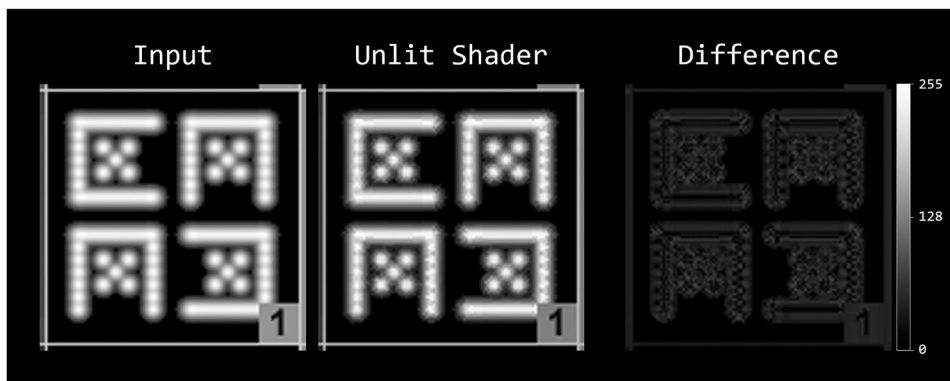
The TG18-QC image was imported as a 2D Texture file nine times with the import settings as detailed in Table 2. The image was then mapped onto the digital test object and rendered using an Unlit Shader, similar to configuration 5 of the first and second experiments. Figure 6(b)

and (c) show the render results of the test pattern using the default import settings, as compared with the original image. Differences between the two images are easier to distinguish once magnified (Fig. 6c). The output image of the Unlit Shader appears to be compressed as suggested by visible color banding. Pixel by pixel difference between the render output image and original test pattern can be visualized in Figure 7. The differences found from each pixel was then plotted in a histogram for each of the 9 import settings (Figure 8). Import setting 6 (Point Filter Mode) had both the lowest mean and standard deviation, followed by settings 1 (Bilinear Filter Mode) and 5 (Trilinear Filter Mode), which were identical. In

**Fig. 6** (a) TG18-QC test pattern render output with increasing incremental smoothness to the right. Top row is metallic 0, bottom row at metallic 1. (b) Rendered output next to the input image file. (c) Detailed view of input versus output

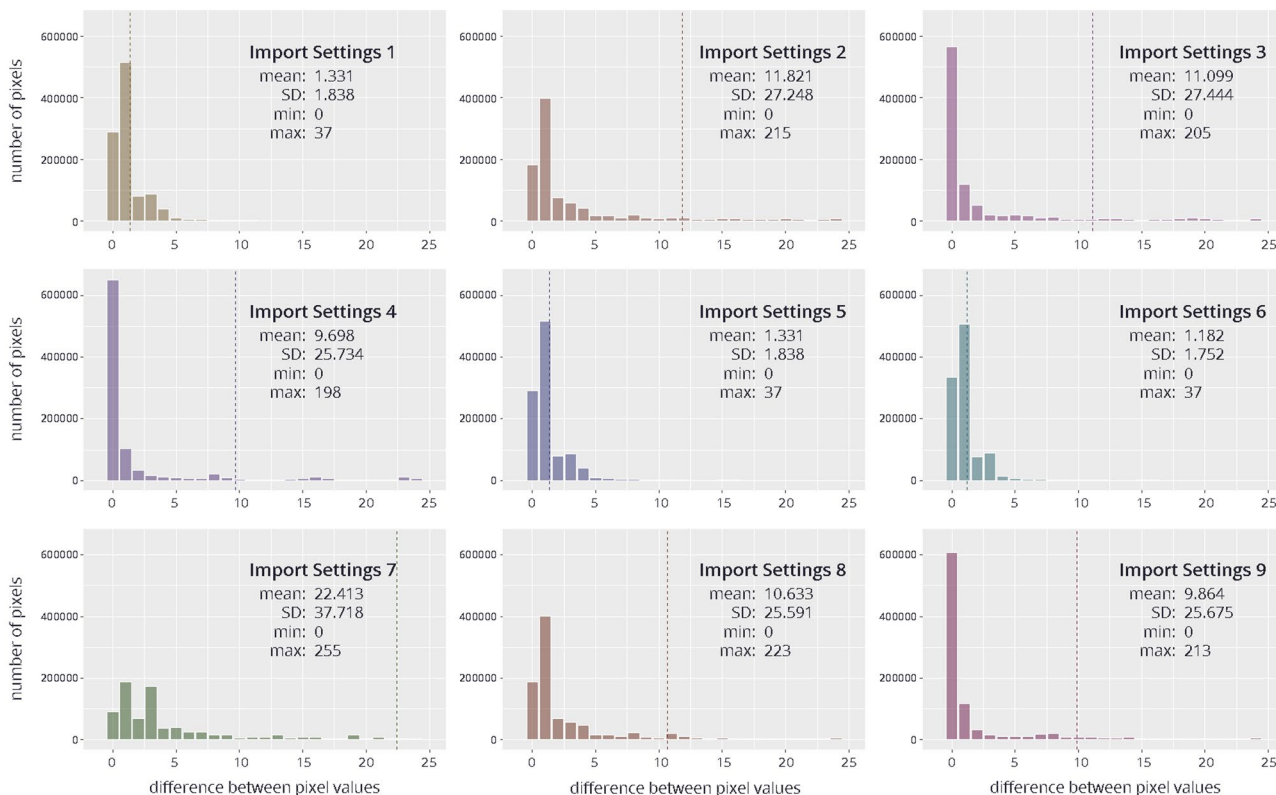


**Fig. 7** Pixel by pixel differences between the render output image and original TG18-QC test pattern



settings 1, 5, and 6, the set Max Size (2048) exceeds the image dimensions (1024) so the resulting textures after image import are not compressed, and Resize Algorithm is not applied. Thus, effects from only the Filter Mode is assessed for settings 1, 5, and 6. Of the three, the Point Filter Mode (no filter) resulted in the least pixel deviation from the original image. Setting 7 had the highest mean and standard deviation, which was expected as the

Max Size was set to 128, followed by settings 2 and 3. Import settings 4, followed by 9 and 3, had the highest number of pixels with difference values of 0. These three import settings used Automatic: High Quality Compression format. Import setting 4 (Bilinear Resize Algorithm) resulted in less pixel deviations than the Mitchell Resize Algorithm (settings 9 and 3).



**Fig. 8** Histograms of the number of pixels with the difference from 0-255 (0-25 shown here) between original 1024x1024 TIFF image TG18-QC pattern and the output 1024x1024 PNG images using 9 different import settings rendered using Unlit Shader



## Discussion

Results from the first experiment show mean render output values for configurations 1–4 covering a range of the CIELAB color space varying from what was input after processing through the Unity render pipeline (Fig. 5a). Once the reference white was calculated into the transfer between color spaces to account for luminance, the default rendering settings (configuration 1) interestingly returned highest  $\Delta E$  values for all of the 24 color patches. Based on the visualization of the render results seen in Figure 4, one may anticipate greatest deviations from the input RGB values with a smooth, metallic material that reflects lighting of the Skybox (configuration 3), or at least less so than the default rendering settings (configuration 1). Yet, configuration 3 resulted in the lowest  $\Delta E$  values out of configurations 1–4, with the default rendering properties resulting in the greatest color difference.

The default Skybox in Unity, which mimics an outdoor setting underneath a blue sky, is a source for environmental ambient lighting and reflections. The Standard Shader with increased smoothness and metallic levels will reflect the environment, as seen in configuration 3 render results (Fig. 4). Even without a defined light source, ambient lighting from the Skybox is calculated at every rendering. The Skybox can be removed by deleting it from the Skybox Material property under the Environment section of the Lighting Settings.

Experiment 1 results from the Unlit Shader script (configuration 5) of zero color difference across all 24 color patches is expected, as the shader is scripted to ignore influences from light sources and environmental effects. Deviations from the input RGB values are expected to occur with changes to material shader properties, light source properties, and camera properties as in configurations 1–4. The proposed framework helps to characterize the various rendering techniques used in exploratory visualizations, as is typically the case with XR. For example, the increasing pattern of  $\Delta E$  values for the grayscale ramp from patches #19–24 for configurations 1–4 show a greater color difference with decreasing RGB values. Users should keep in mind that color deviations in medical data may be greater than what they may anticipate strictly based on visualizations.

Results from the second experiment (Fig. 5b) were less predictable and did not reflect those seen in experiment 1. These results suggest that color transfer in the pixels is affected by factors beyond just rendering configurations. Based on our study design, the only difference in methods between experiment 1 and 2 was the import of an external image as a Unity-readable 2D Texture file. Thus, the image import process and the resulting texture that is displayed as a material was tested for additional color distortions. The third

experiment found texture import settings to add distortions to the pixels, even when using an Unlit Shader. Though the exact reason for the color banding effect is not identified in this investigation, experiment 3 highlights the effects of various import settings on the image.

When building an XR application for different platforms, developers should consider the file size with associated memory size requirements, pixel dimensions, and resolution of the image textures for each target platform[25], as the type of texture compression is dependent on the intended platform. For instance, a standalone XR HMD, Android mobile device, and a PC will each have their own unique compression formats that work with their specific hardware as some graphics devices only use certain compressed formats. Developers have the ability to designate specific compression settings for each platform in the import settings of the inspector window.

Thus, while the Unlit Shader yields predictable and consistent transfer of color throughout the rendering pipeline for input colors, it is not always the case when using a texture due to how the image is imported. Though the TG18-QC render result seen in Figure 6(b) may not be significantly noticeable to the human eye, image quality should be preserved as much as possible for imaging datasets to be used in medical XR visualizations.

## Limitations of the Study

While the characterization of color transfer in software remains important, user studies should be conducted to quantify how the different parameters ultimately affect user tasks and human perception. Variability in color across HMDs need to be addressed in future work. Furthermore, this study was limited to the software component and did not investigate the hardware contributions to this effect. For example, a field-programmable gate array (FPGA) circuit board may be used to capture the digital pixel data transmitted on high-definition multimedia interface (HDMI) to capture the pixel data altered by the color management, the graphics card driver, and the XR applications following rendering.

This investigation did not address the influence of tone mapping and post-processing effects. Unity offers several tone mapping options, and a custom tone mapper allows users to create their own transfer curves, which can heavily affect colors and saturation[26]. In addition, this investigation does not address the use of baked lighting as compared with real-time lighting.

Another limitation of this study is that various corrections including sampling strategies and inverse lens distortion that might be applied to HMDs are not addressed in our experiments. The nature and degree of inverse distortion implemented is dependent on the optical specifications of each

HMD as well as on the eye centering and interpupillary distance of the user. Future studies that evaluate complete software implementation of an XR system may benefit from adding these considerations to their measurement setups.

Lastly, anisotropic filtering takes into account that, due to the camera orientation, the output polygon may not be rectangular. This filter method applies additional transformations to compute the effect of camera angle on the dimensions of the output. As the rendering camera zooms in on a texture, the pixels of the texture need to be mapped onto multiple pixels in the output image. In this study, the rendering camera is faced straight-on toward the texture mapped onto the digital test object. Effects of texture filtering may have been better evaluated if the camera angle was not perpendicular to the digital test object, or if the digital object was slightly rotated.

## Conclusion

Color transfer of materials using the Unlit Shader is unaffected by the Unity rendering pipeline, except when using texture files due to image compression during import. Deviations from the input RGB values occur with changes to material shader properties, light source properties, and camera properties. The proposed framework addresses the characterization of the rendering techniques for exploratory XR visualizations. This research provides developers of medical XR applications with considerations regarding how to ensure color information displayed in the headset is accurate and consistent. While a custom shader script may be appropriate for medical data visualization, such a shader would not be universally applicable and standardized for all visualization tools. To investigate this issue, a standardized instrument for color measurement of virtual objects will be further implemented to fully characterize color transfer in rendering pipelines.

As the XR user community expands to include medical professionals and applications, so does the need for an interdisciplinary understanding and control over graphics rendering properties. The effects of various texture import settings in color difference are highlighted in this work. While these measures of change in color image properties are specific to the Unity engine, this methodology could be extended to address similar concerns for other software tools utilized in medical XR application development.

## Compliance with Ethical Standards

**Conflict of Interest:** The authors declare that they have no conflict of interest.

## References

1. Valcasara N, Unreal Engine Game Development Blueprints. Packt Publishing, 2015.
2. Kim SL, Suk HJ, Kang JH, Jung JM, Laine TH, Westlin J, Using Unity 3D to facilitate mobile augmented reality game development, 2014 IEEE World Forum on Internet of Things (WF-IoT), 2014. <https://doi.org/10.1109/WF-IoT.2014.6803110>
3. Burley B, Physically-based shading at Disney. In ACM SIGGRAPH 2012 Courses, ACM, SIGGRAPH 2012.
4. Cook RL, Torrance KE, A reflectance model for computer graphics. *ACM Trans. Graph.* 1, 5, pp.7-24, 1982.
5. Walter B, Marschner SR, Li HS, Torrance KE, Microfacet models for refraction through rough surfaces, The Eurographics Association, Proceedings of the 18th Eurographics Conference on Rendering Techniques, pp.195-206, 2007.
6. Blinn JF, Models of light reflection for computer synthesized pictures. Proceedings of the 4th Annual Conference on Computer Graphics and Interactive Techniques, Association for Computing Machinery, New York, NY, pp.192-198, 1977. <https://doi.org/10.1145/563858.563893>
7. Schlick C, An inexpensive BRDF model for physically-based rendering, *Computer Graphics Forum*, 13(3), pp.223-246, 1994.
8. Cheng WC, Saleheen F, Badano A, Assessing color performance of whole-slide imaging scanners for digital pathology, *Color Research & Application*, 44(3), pp.322-334, 2019. <https://doi.org/10.1002/col.22365>
9. Samei E, Badano A, Chakraborty D, Compton K, Cornelius C, Corrigan K, Flynn MJ, Hemminger B, Hangiandreou N, Johnson J, and others, Assessment of display performance for medical imaging systems: executive summary of AAPM TG18 report, *Medical Physics*, 32(4), pp.1205-1225, 2005.
10. Unity. <https://unity3d.com/>. Accessed 15 Jan 2020
11. Wheeler G, Deng S, Toussaint N, Pushparajah K, Schnabel JA, Simpson JM, Gomez A, Virtual interaction and visualisation of 3D medical imaging data with VTK and Unity, *Healthcare technology letters*, 5(5), pp.148–153, 2018. <https://doi.org/10.1049/htl.2018.5064>
12. Seif M, Umeda R, Uehara Y, Higa H, A Data Conversion for Medical Training System, *IEEJ 2016 Conference*, 2016.
13. Wang R, Yao J, Wang L, Liu X, Wang H, Zheng L, A surgical training system for four medical punctures based on virtual reality and haptic feedback, 2017 IEEE Symposium on 3D User Interfaces (3DUI), pp.215-216, 2017. <https://doi.org/10.1109/3DUI.2017.7893348>
14. Peters TM, Linte CA, Yaniv Z, Williams J, Mixed and augmented reality in medicine, CRC, 2018.
15. Escobar-Castillejos D, Noguez J, Neri L, Magana A, Benes B, A Review of Simulators with Haptic Devices for Medical Training, *Journal of Medical Systems*, 40(104), 2016. <https://doi.org/10.1007/s10916-016-0459-8>
16. Unity Technologies, Unity Manual 2019.2: Standard Shader. <https://docs.unity3d.com/Manual/Shader-StandardShader.html>. Accessed 15 Jan 2020.
17. Lagarde S, The High Definition Render Pipeline: Focused on visual quality. <https://blogs.unity3d.com/2018/03/16/the-high-definition-render-pipeline-focused-on-visual-quality/>. Accessed 15 Jan 2020.
18. Unity Technologies, Unity Manual 2019.2: Global Illuminance. <https://docs.unity3d.com/Manual/GIIntro.html>. Accessed 15 Jan 2020.
19. Khor WS, Baker B, Amin K, Chan A, Patel K, Woong J, Augmented and virtual reality in surgery the digital surgical environment: applications, limitations and legal pitfalls, *Annals of translational medicine*, 4(23), 2016.
20. Chen L, Day TW, Tang W, John NW, Recent developments and future challenges in medical mixed reality, 2017 IEEE International Symposium on Mixed and Augmented Reality (ISMAR), pp.123-135, 2017.

21. Vávra P, Roman J, Zonča P, Ihnát P, Němec M, Kumar J, Habib N, El-Gendí A, Recent development of augmented reality in surgery: a review, *Journal of healthcare engineering*, 2017.
22. Unity Technologies, Unity Manual 2019.2: Textures. <https://docs.unity3d.com/Manual/class-TextureImporter.html>. Accessed 15 Jan 2020.
23. International Electrotechnical Commission and others, Multimedia systems and equipment-Colour measurement and management-Part 2-1: Colour management-Default RGB colour space-sRGB, IEC 61966-2-1, 1999.
24. Ohta N, Robertson A, *Colorimetry: fundamentals and applications*. John Wiley & Sons, 2016.
25. Unity Technologies, Unity Scripting API 2019.2: Texture Importer Platform Settings. [docs.unity3d.com/ScriptReference/TextureImporterPlatformSettings.html](https://docs.unity3d.com/ScriptReference/TextureImporterPlatformSettings.html). Accessed 15 Jan 2020.
26. Dille S, Fuhrmann A, Fischer G, Real-time tone mapping- An evaluation of color-accurate methods for luminance compression, *Tagungsband des 22. Workshop Farbbildverarbeitung*, Ilmenau, Germany, 2016.