Scientific Visualization, 2023, volume 15, number 2, pages 90 - 101, DOI: 10.26583/sv.15.2.08

Scientific Visualization for Interrogation of Results in Lighting Simulation System

E.D. Biryukov[1], B.Kh. Barladian[2], E.Yu. Denisov[3], A.G. Voloboy[4]

Keldysh Institute of Applied Mathematics RAS

[1] ORCID: 0000-0003-4297-6813, birukov@gin.keldysh.ru
[2] ORCID: 0000-0002-2391-2067, bbarladian@gmail.com
[3] ORCID: 0000-0002-0614-9100, eed@spp.keldysh.ru
[4] ORCID: 0000-0003-1252-8294, voloboy@gin.keldysh.ru

Abstract

Lighting modeling software packages often include a tool for processing and investigation of two-dimensional images that are the results of modeling. Functionality of these tools is usually minimal one and often consists of mapping of resultant images into monitor screen only. There are general image viewers and image processing programs. But in practice they either do not contain required functionality, or are so superfluous that result postprocessing becomes inconvenient with them. Therefore we had to implement own viewer that processes high dynamic range images generated by the lighting simulation and realistic computer graphics system as well as visualizes supplementary simulation information needed for engineer analysis. Lighting simulation module transfers to our viewer the generated image in physical values together with supplementary data. These data has per pixel nature and form additional layers of images. Our image processing module provides brightness high dynamic range compression for rendering of images generated with lighting modeling methods including stochastic ones, analysis of the image generation algorithms using additional data layers, applying additional visual effects for improving images appearance, and export of processed images to commonly used formats.

Keywords: Simulation result visualization, Visualization modes, High dynamic range image, Tone mapping, Image filtering, Highlight desaturation.

1. Introduction

Photorealistic three-dimensional graphics and lighting modeling software packages often include a tool for processing and investigation of two-dimensional images that are the results of modeling. Functionality of these tools is usually minimal one and often consists of mapping of resultant images into monitor screen only. There are many well-known general image processing programs (for example, Adobe Photoshop [1], ACDSee [2], Corel PaintShop Pro [3], and others). But in practice they either do not contain required functionality, or are so superfluous that result postprocessing becomes inconvenient with them. So, what functionality is necessary for researcher or optical designer?

At first, there should be support of high-dynamic range (HDR) images [4]. Most of usual images have so called low dynamic range. They keep information in the way suitable for direct display on a computer monitor. Pixel brightness is there expressed in abstract values relative to maximal possible brightness of a monitor, with fixed number of linearly distributed intermediate steps. This number is called color depth and usually is equal to 256 (8 bit) per color channel. So the low dynamic range is two orders. But results of lighting simulation are physical values (of luminance or radiance) which dynamic range is more than two orders in most cases. Also human vision is capable to perceive luminance values from 10^{-6} to 10^{8} cd/m^2, so this range should be provided in HDR images [5]. While the low dynamic range im-

age often keeps picture in RGB color model (because it is suitable for fast rendering) the HDR image can have other color models like spectral one or CIE XYZ. Here the color model depends on lighting simulation purpose and researcher or designer tasks. Obviously, for displaying such images on a monitor screen, luminance or radiance values of each pixel should be mapped into color space of a monitor using algorithms of dynamic range compression and conversion into the RGB color model. Such conversion is often called Tone Mapping Operator (TMO).

Besides support of HDR format, image processing tool should provide additional display modes for showing auxiliary modeling results, for example, achieved accuracy or number of processed rays. The various display modes [6] can be used here but most usable is a heat map or visualization in false colors. Also sometimes modeling results are recorded in polar coordinates or represented as a spherical panorama. Such data should also be converted into form suitable for rendering on a screen. In some cases it is required to provide additional interaction between image processing module and main lighting modeling program, for example, when some area should be selected on the image and its coordinates are passed to the main program for further simulation.

There are several viewers of HDR images [7-9]. All they provide rich toolkit for investigation and modification of HDR images. Also work [10] proposes to use virtual and mixed reality tools for evaluation of lighting modeling results. However they cannot be used for analysis of auxiliary simulation result and supplementary information produced by lighting simulation system.

This is why we had to implement own viewer that processes HDR images generated by the lighting simulation and realistic computer graphics system as well as visualizes supplementary simulation information needed for engineer analysis.

2. Layer-based structure of image

Initially the image intended for postprocessing in separate program contained only luminance or illuminance values for each pixel. However, later it became necessary to record additional data. So possibility of transmission of new data was provided. These data have “per pixel” nature and therefore we call them layers of image. However, creation of a layer not attached to the image coordinate scale is also possible. For example, for each image in spectral color model the array of wavelengths for which the image was calculated must be also stored in layer. Layers are defined by names which are hard-coded for specific data, and data are being searched and read by these names during loading into the image processing program. Main layers which currently are recorded to files besides the image itself are the following: grid image, calculation inaccuracy matrix, matrix of numbers of ray hits for each pixel, three-dimensional coordinates of closest scene point corresponding to each pixel.

Historically, some layers were included in the main image file itself, while other layers were recorded into separate files with name corresponding to the main image file and differing by extension. In particular, the layer with three-dimensional coordinates of pixels and the layer with scene object indices are stored to separate files. Also, if several kinds of physical values (luminance, illuminance, etc.) are calculated during single lighting simulation for the same camera or virtual measuring device (observer), then these data also are written in separate files.

Structure of layers is different depending on what kind of image (virtual camera or measuring device/area) we intend to investigate. Fig. 1 shows the layer structure for camera image (i.e. rendering results).

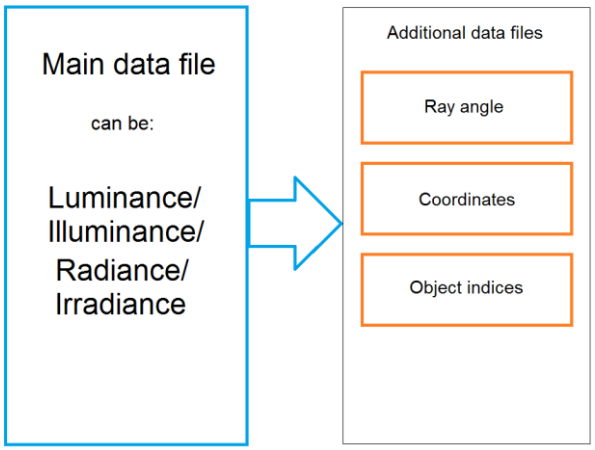

Fig. 1. Data layers for rendering results.

Here each layer is usually stored in a separate file. Any of these files can be recorded during rendering according to user settings. Then, in the image processing module all existing additional files (ray angle values, coordinates and object indices) will be opened automatically when user opens one of the main image files (luminance, illuminance, radiance, irradiance).

Virtual measuring devices or areas are called observers. For observer results layer structure is different as it is shown on Fig. 2.

Here all layers are stored in one file. Besides the main image layer, the layers with

- inaccuracy data,
- cell areas (useful if observer cells have different areas),
- projection coefficients (ratio of distance of point projections to surface from axis to distance of point from axis),
- ray hits numbers and inverse values of cosines for luminance gonio observer

can be optionally stored.

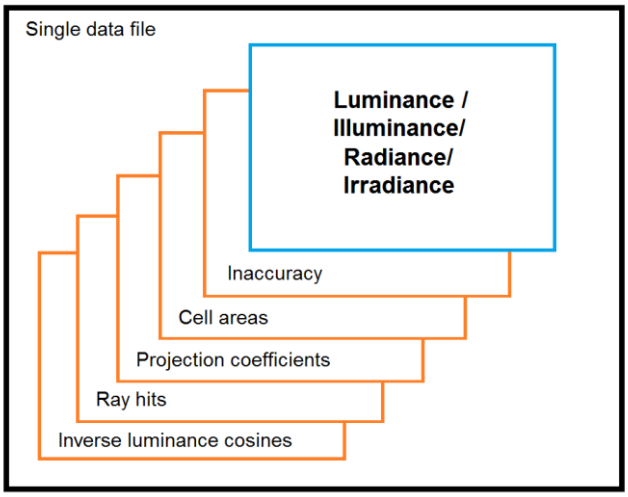

Fig. 2. Data layers for observer results.

Also we are using different types of data arrangement in files depending on rendering or observer data, image size, and some other parameters. It is possible to group recorded data by layers (so at first is recorded entire luminance layer for all pixels, then entire inaccuracy layer, and so on), or group it by pixels (record all existing data corresponding to the first pixel, then for second pixel, and so on). It is even possible to group data by small blocks corresponding to some rectangular part of an image. This way is used if rendering algorithm generates such

blocks sequentially. But it should be noticed that the main image layer (luminance / illuminance / radiance / irradiance) is always considered as the single layer, not depending on data arrangement type and not depending on number of color components (three for RGB or XYZ or possibly more for spectral color model). So all color component for each pixel are always grouped together.

3. Image visualization modes

3.1 Tone mapping with compression of high dynamic range

One of the main functions of the HDR image processing program is mapping of high dynamic range image into monitor color space and rendering it on monitor screen. Dynamic range compression task is necessary for optical modeling software package because images obtained with lighting modeling and rendering are represented with calculated physical values which dynamic range is arbitrary.

In principle, the history of the development of Tone Mapping Operator (TMO) has been going on for more than 20 years [11-13]. In the early 2000s, basic and improved TMO algorithms for a single image were developed. They take into account the specifics of perception of the human eye. We also developed the original TMO algorithm [14]. At present, the development of tone mapping is still relevant [15]. However, modern algorithms are being developed for a wide range of applications [16]. Among them, TMO for video [17], TMT for image preprocessing for recognizing key points and images, for image matching [18] can be noted. The direction of development of reverse TMT for converting an image with a low dynamic range into a high one has received great development [19].

The following three variants of dynamic range compression are currently used in our image processing module:

1. Nonlinear dynamic range compression operator [14];
2. Simple linear compression operator controlled by the single parameter of maximal brightness level;
3. Tabulated compression operator where tone mapping is defined by an editable mapping table.

Our nonlinear compression operator is based on the improved Tumblin-Rushmeir operator [11]. It is defined by the formula:

$$L_d = m(L_{wa}) * L_{da} * \left(\frac{L_w}{L_{wa}}\right)^{\left(\frac{\gamma_w}{\gamma_d}\right)} \quad (1)$$

where L_w are initial values of pixel physical brightness, L_{da} is screen adaptation brightness (typical values are 10-30 cd/m2), L_{wa} is scene adaptation brightness, $m(L_{wa})$ is a coefficient depending on the scene adaptation brightness which prevents abnormally grey nightly images, γ_w and γ_d are defined using the Stevens formula for contrast sensitivity [20]. In the general case, after using the improved Tumblin-Rushmeir compression operator, brightness of some pixels corresponding to the brightest part of a scene may exceed maximal brightness of a monitor, and so corresponding part of an image will be oversaturated. For overcoming this problem our operator uses the formula suggested in [12, 13] for compressing high brightness:

$$L_{df}(x,y) = \frac{L_d(x,y)\left(1 + \frac{L_d(x,y)}{L_{white}^2}\right)}{1 + L_d(x,y)} \quad (2)$$

where L_d is pixel brightness on the screen, obtained from the previous expression, and L_{write} is minimal brightness which is mapped to pure white color. For images with very high dynamic range, white level L_{white} can be any large enough value (usually it is close to maximal scene brightness). Controlling parameters for our operator are maximal luminance or illuminance value which won't be clipped, and scene adaptation luminance.

Linear compression operator is controlled by single parameter corresponding to maximal luminance or illuminance value. This parameter can be set explicitly as well as through a set

of photographic parameters such as exposure time, f-number (relative aperture) of the camera lens, and ISO speed.

User interface of the compression operators are shown on Fig. 3.

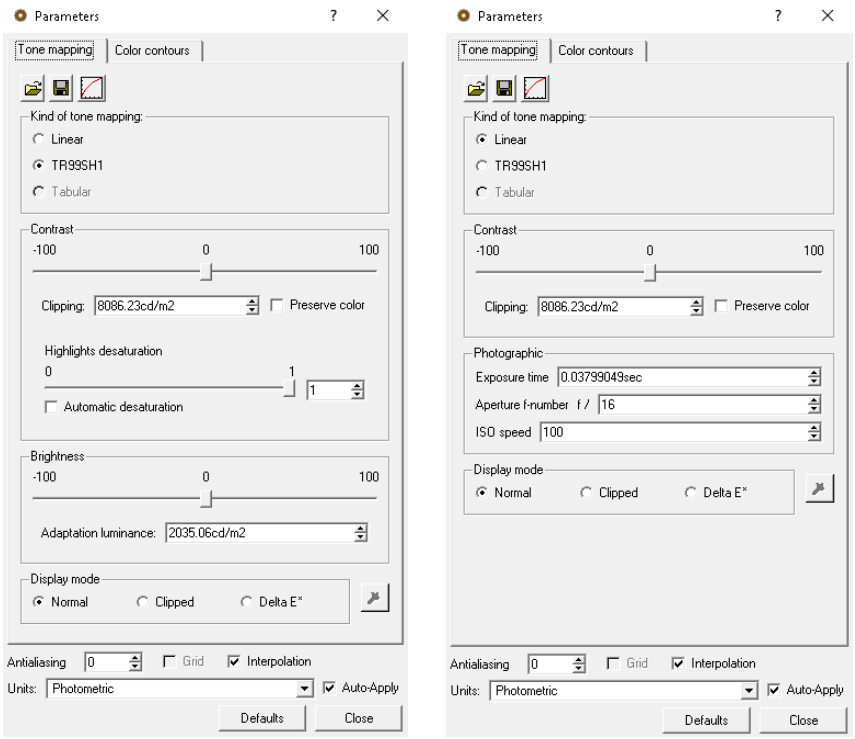

Fig. 3. User interface elements to control the dynamic range compression operators (left – nonlinear operator, right – linear operator).

Besides general controls for tone mapping setting, the Display mode switch is present (Normal, Clipped and Delta E* modes). Normal mode just shows original image after tone mapping. Clipped mode inverses luminance values and clips them before tone mapping. So in this mode negative luminance values are painted in the same way as positive values in the Normal mode and positive values are painted in black. This is useful for interrogation of negative luminance values which can be obtained, for example, during conversion of spectral data to RGB color model. And the Delta E* mode shows color distance between original luminance/illuminance values and the same values packed to compact HDR format. CIE94 formula is used for calculation of color distance [21]. This formula uses the CIE Lab color model, so RGB color components should be converted to *L*, *a* and *b* values.

Obtained value is displayed using the false colors (heat map) visualization mode.

3.2 Luminance, illuminance, Delta E visualization

Our viewer provides false colors visualization mode intended for showing physical values in a way suitable for human perception. This mode allows visualization with isolines as well as without them. User interface in this case can control visualization colors as well as luminance, illuminance or calculation inaccuracy values corresponding to these colors. Example of image display in the false colors mode (with dialog box for setting parameters) is shown on Fig. 4.

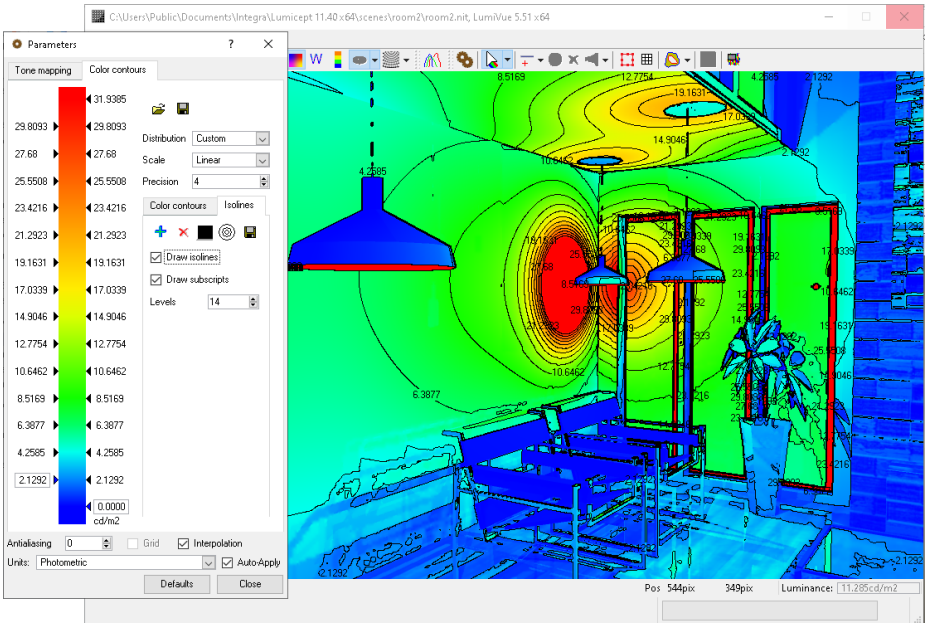

Fig. 4. Visualization of surface illumination values.

The same false colors mode is used for displaying color difference for Delta E* mode. Own color scale and setup dialog are used in this case.

3.3 Visualization of simulation inaccuracy and ray hit numbers

The layer containing simulation inaccuracy values corresponding to pixels and the layer with numbers of ray hits for each pixel gather data during bidirectional Monte-Carlo ray tracing for virtual measurement devices (so-called observers).

Inaccuracy values are displayed in false colors. A special mode is intended for their display. It provides possibility to find scene parts where geometry features do not allow achieving the required accuracy of calculations, and correct them. This mode can be turned on in the dialog of setting of main parameters. The same control as for luminance and illuminance visualization in false colors mode are available in this case. This mode and its user interface controls can be seen on Fig. 5.

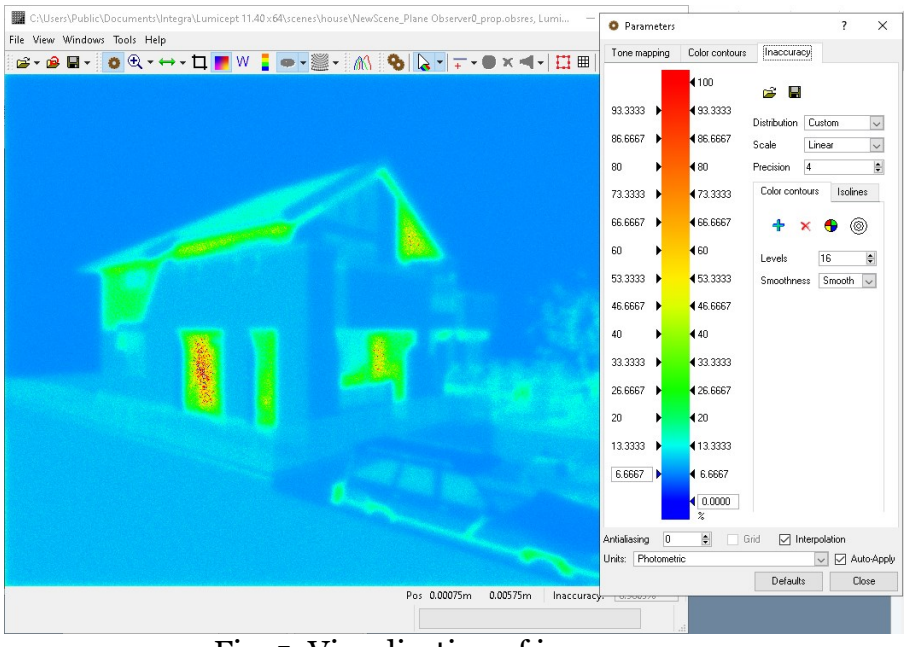

Fig. 5. Visualization of inaccuracy.

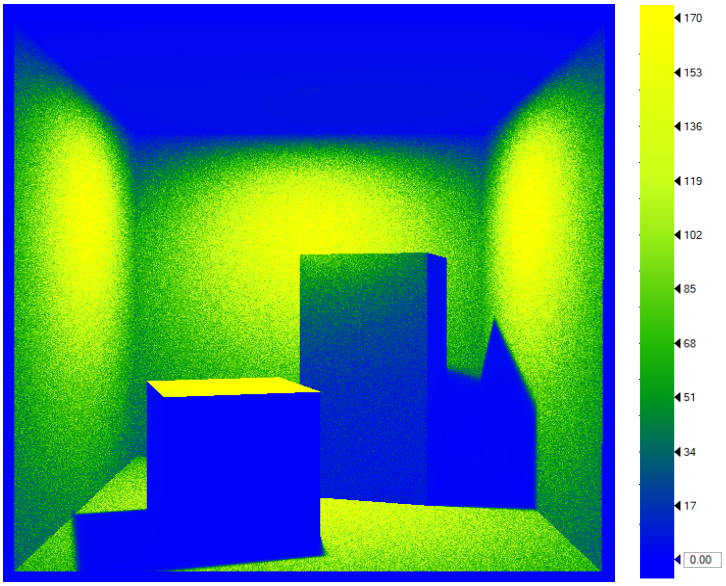

Fig. 6. Visualization of ray hits numbers.

Also it is possible to show layer with number of rays which hit each pixel. This functionality is useful for investigation of simulation method efficiency. The layer is also displayed in false colors like the previous one. Example of this layer is shown on Fig. 6.

3.4 Layer of coordinates of 3D scene points

One more layer recorded together with image contains coordinates of scene points in three-dimensional space corresponding to pixels. One of the cases when these data are necessary is measurement of the distance between points in scene space. This functionality allows selecting two pixels on the image and calculating distance between points in the scene three-dimensional space using previously recorded coordinate values. It is useful for virtual scene investigation.

Dialog box for distance measurements is shown on Fig. 7.

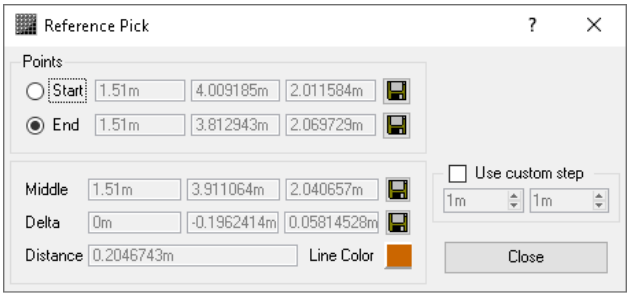

Fig. 7. Dialog of distance measurement between scene points.

Possibility of distance measuring between two points had already been provided in the main program of lighting simulation system. Workspace of this program represents itself a three-dimensional graphics editor which allows moving the viewpoint in the space. But adding measurement functionality also into image processing program can significantly improve usability because during analyzing of results user should not switch between programs each time and manually search for the scene point corresponding to required pixel. In some cases such manual search is even impossible. For example, when fisheye camera with view angle near to 180 degrees or even greater is used for rendering, scene preview in the workspace is impossible with such great view angles. Also such measurement functionality can be extremely useful in cases when user should measure distance between some light/dark spots, or size of these spots, i.e. in the case when measurement points became known only after simulation and visible only in the simulation results.

Also this functionality allows distance measuring on a plane. Generally it is needed during work with images representing result of a simulation on observers. As observer dimensions are set in real physical units (not just in numbers of pixels), calculated distance between points in this case is also measured in real physical units. Possibility of setting custom scale of coordinate axes on images (expressed as a percentage relative to the pixel numbers of the original image) and calculating distances between pixels with this scale is also provided.

4. Additional display functionalities

4.1 Layer of coordinates of 3D scene points

If simulation of light propagation is performed with stochastic methods then resulting images inevitably will be noisy. So image filtering algorithms are applied to reduce noise. Two simple denoising algorithms are currently implemented in our image processing program: average filter and median filter. Average filter calculates mean value of pixels adjacent to the current one and sets this mean value to the current pixel. Usage of this filter may cause blurring of the image. Median filter collects an array of adjacent pixels, sorts it, and selects value of the middle element of the array as the pixel color value. In most cases these two filters are enough for simple image denoising.

A filter can be implemented in different ways depending on what color components of point should be filtered. It can be:

- different arrays for each color channel; this approach results in very low performance (especially for median filters) [22, 23];
- usage of vector median (only for median filter); this also is time consuming [24];
- usage of some scalar value corresponding to pixel (for example, luminance or illuminance).

In our program we use the last variant.

The following parameters can be set to control image filtering: filter type (average or median), filter size in pixels and number of passes of the algorithm (Fig. 8).

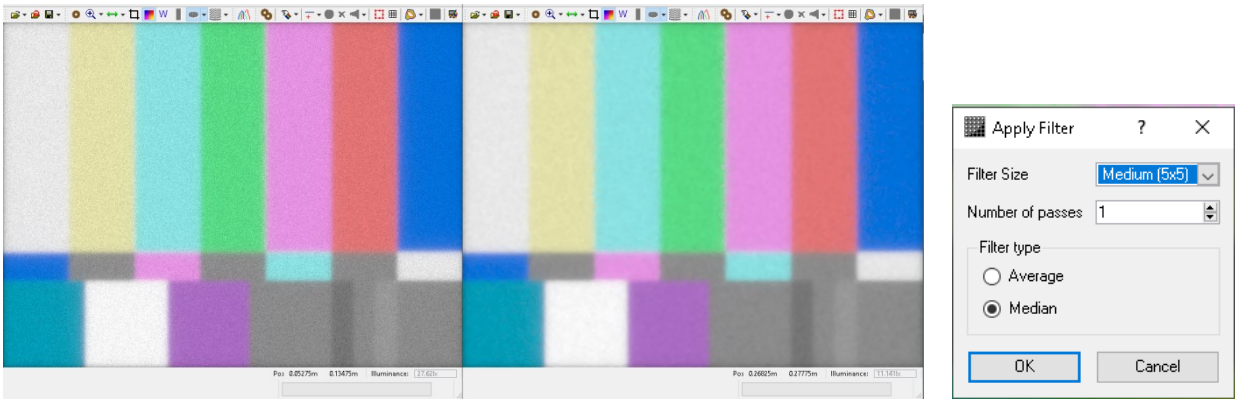

Fig. 8. Applying median filter (left – original image, right – resulting image) and parameters dialog for this filter.

Total working time for median filter algorithm with 5x5 size is about 1 second for image with resolution 1366x768 pixels and about 2.5 minutes for 15000x15000 image using AMD Ryzen 9 3900X 12-Core processor and 32 GB RAM.

4.2 Highlights desaturation

The brightest image parts look simply white for human, i.e. overexposed areas “lose” coloring in our perception [25, 26]. It should be taken into account for simulation of human vision or photographic equipment specifics. But sometimes the image dynamic range is not enough high for compression algorithm to take this desaturation into account. So an addi-

tional saturation decreasing for brightest parts of image (highlights) is advisable. Such functionality was implemented in our image processing program.

Shortly, the algorithm of desaturation method is the following:

1. Mean value of color channels for particular pixel is calculated;
2. Difference between each particular color channel value and the mean value is obtained;
3. Pixel brightness is divided by highest brightness of the image. This value will be considered as individual desaturation factor of this particular pixel.
4. From each color channel, its difference with the average value of the channel multiplied by the individual desaturation factor is subtracted.

Thus, values of color channels become closer to each other and saturation decreases. The algorithm details are in [27].

For better automatic tuning of the algorithm, it was bound to the statistical parameters of the image. Bright highlights for which desaturation is required have area significantly less than the whole image area. Final variant of the algorithm got the brightness value so that only 10% of pixels have greater brightness. Then this value was doubled and used as a threshold value for desaturation. Individual pixel desaturation factor is equal to zero for the threshold and lower values, and equal to 1 for the maximal brightness value of the image. Linear interpolation is used for all intermediate values. Result of applying this method is shown in Fig. 9. From human visual perception point of view the desaturated image looks more plausible.

Fig. 9. Result of the highlight desaturation algorithm (on the left – the original image, on the right – after applying the desaturation).

Control of the highlights desaturation system consists in switching between automatic and manual modes. In manual mode user should set threshold brightness value by himself, in automatic mode it is set according to the described algorithm.

5. Conclusion

Described images analyzing and processing module is currently integrated in realistic computer graphics and lighting simulation system Lumicept [28]. It provides brightness high dynamic range compression for rendering of images generated with lighting modeling methods including stochastic ones, analysis of the image generation algorithms using additional data layers, applying additional visual effects for improving images appearance, and export of processed images to commonly used formats.

Enhancement of the image processing module continues persistently. Currently main trends of its development are improvement of dynamic range compression operators together with additional desaturation algorithms for brightest highlights. Also new modes of image analysis can be implemented depending on designer's and engineer's needs. Structure of image layers elaborated by us is so flexible that allows adding new ones and modifying existent ones with low development cost.

References

1. Adobe Photoshop User Manual, URL: https://helpx.adobe.com/photoshop/user-guide.html (accessed on 15.08.2022).

2. ACDSee Photo Studio Software, URL: https://www.acdsee.com/en/products/photo-studio-ultimate (accessed on 15.08.2022).

3. Photo Editing Software – PaintShop Pro, URL: https://www.paintshoppro.com/en/products/paintshop-pro/ultimate/#features (accessed on 15.08.2022).

4. Reinhard, E., Heidrich, W., Debevec, P., Pattanaik, S., High Dynamic Range Imaging: Acquisition, Display, and Image-based Lighting. Morgan Kaufmann, Burlington, 2010.

5. Voloboy, A.G., Galaktionov, V.A., Kopylov, E.A., Shapiro, L.Z., Simulation of natural daylight illumination determined by a high dynamic range image, Programming and Computer Software, 32(5), 2006, pp. 284-297. DOI:10.1134/S0361768806050057

6. Birukov, E.D., Voloboy, A.G., Denisov, E.Yu., Elaboration of Visualization Modes for Lighting Simulation in CATIA CAD System,. Scientific Visualization, 12(4), 2020, pp. 123 - 132, DOI: 10.26583/sv.12.4.11

7. HD Shop, URL: https://vgl.ict.usc.edu/HDRShop (accessed on 11.03.2023).

8. Wirth, M., "Advanced HDR image viewer". 2017, Charles University, Faculty of Mathematics and Physics, Praha, Czech. URL: https://dspace.cuni.cz/handle/20.500.11956/2088 (accessed on 27.03.2023).

9. Kumaragurubaran V., Inanici M., Hdrscope: high dynamic range image processing toolkit for lighting simulations and analysis //Proceedings of the BS2013: 13th Conference of International Building Performance Simulation Association, 2013, pp. 25-28. http://www.ibpsa.org/proceedings/BS2013/p_1194.pdf (accessed on 27.03.2023).

10. Krupinski, R., Virtual Reality System and Scientific Visualisation for Smart Designing and Evaluating of Lighting, Energies, 13(20), 2020, 5518, DOI: 10.3390/en13205518

11. Tumblin, J., Three Methods For Detail-Preserving Contrast Reduction For Displayed Images. PhD thesis, Georgia Institute of Technology, 1999.

12. Reinhard, E., Stark, M., Shirley, P., and Ferwerda, J., Photographic Tone Reproduction for Digital Images, ACM Transactions on Graphics, 21(3), 2002 (Proceedings of SIGGRAPH 2002), pp. 267-276.

13. Reinhard, E., Parameter estimation for photographic tone reproduction. Journal of graphics tools, 7(1), 2002, pp. 45-52.

14. Barladyan, B.H., Voloboy, A.G., Galaktionov, V.A., Kopylov, E.A., An Effective Tone Mapping Operator for High Dynamic Range Images, Programming and Computer Software, 30(5), 2004, pp. 266-272.

15. Rana, A., Singh, P., Valenzise, G., Dufaux, F., Komodakis, N., Smolic, A., Deep Tone Mapping Operator for High Dynamic Range Images, IEEE Transactions on Image Processing, 29, 2020, pp. 1285-1298, DOI: 10.1109/TIP.2019.2936649

16. Debattista, K., Application-Specific Tone Mapping via Genetic Programming, Computer graphics forum, 37(1), 2018, pp. 439-450.

17. Melo, M., Bessa, M., Debattista, K., Chalmers, A., Evaluation of Tone-Mapping Operators for HDR Video Under Different Ambient Luminance Levels, Computer Graphics Forum, 34(8), 2015, pp. 38-49.

18. Rana, A., Valenzise, G., Dufaux, F., Learning-based tone mapping operator for efficient image matching, IEEE Transactions on Multimedia, 21(1), 2018, pp. 256-268.

19. Zhang, Y., Zhou, W., Xu, J., A dynamic range adjustable inverse tone mapping operator based on human visual system, The Visual Computer, 39, 2023, pp. 413–427, DOI: 10.1007/s00371-021-02338-5

20. Stevens J.C., Stevens S.S., Brightness function: Effect of Adaptation, Journal of the Optical Society of American, 53(3), pp. 375-385, 1963.

21. McDonald, R., Smith, K.J., CIE94-a new colour-difference formula, Journal of the Society of Dyers and Colourists, 111(12), 1994, pp. 376-379.

22. Gonzalez, Rafael C., Woods, Richard E., Eddins, Steven L., Digital Image Processing Using Matlab. Pearson Education, 2004.

23. Tavse, S.S., Jadhav, P.M., Ingle, M.R., Optimized Median Filter Implementation on FPGA Including Soft Processor, International Journal of Emerging Technology and Advanced Engineering, 2(8), 2012, p. 236-239.

24. Liu Y., Noise reduction by vector median filtering, Geophysics, 78(3), 2013, pp. V79-V87.

25. Rizzi A., Gatta C., Marini D., A new algorithm for unsupervised global and local color correction, Pattern Recognition Letters, 24(11), 2003, pp. 1663-1677.

26. Guo, D., et al., Correcting over-exposure in photographs, 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, IEEE, 2010, pp. 515-521.

27. Birukov, E.D., Kopylov, M.S., Khlupina, A.A., Correction of color saturation for tone mapping operator, CEUR Workshop Proceedings, 2485, Proc. of the 29th International Conference on Computer Graphics and Vision, Russia, 2019, pp. 58-61, DOI: 10.30987/graphicon-2019-2-58-61

28. Lumicept – Hybrid Light Simulation Software, URL: http://www.integra.jp/en (accessed on 03.04.2023).