Techniques for Real-Time Spectral Rendering

Pedro Jose Perez, Nestor Monzon, Adolfo Muñoz

Affiliation: Graphics & Imaging Lab (GILAB)
Instituto de Investigación en Ingeniería de Aragón (I3A)
Universidad de Zaragoza, Mariano Esquillor s/n, 50018, Zaragoza, Spain.
Tel. +34-976762707, e-mail: p.perez@unizar.es

Abstract

We present a real-time, physically-based rendering pipeline that enables the usage of traditional RGB textures for spectral rendering in real-time contexts by leveraging pre-existing spectral upsampling techniques for reflectances. We combined our approach with other techniques, such as real-time underwater rendering, to prove its extensibility.

Introduction

RGB and Spectral Rendering

RGB rendering generates images by discretizing the entire visual spectrum into just 3 colours: red, green, and blue. It has been the standard for years due to its simplicity, speed, and the decent quality of its results.

Traditional RGB rendering presents problems during image synthesization upon encountering wavelength-dependent phenomena, such as participating media scattering or iridescence, introducing error. Spectral rendering takes into account the entire visual spectrum, and thus, generates more accurate colours in the final images. However, the former is still the dominant choice due to how expensive it is to produce spectral materials and assets based on real world data, usually relegating the latter to niche applications.

Considerations for Real-Time Rendering

For real-time applications, such as video games, architectural visualizations or virtual reality settings, high rates of around 60-120 generated images per second must be achieved, in contrast to offline rendering, where synthesis of a single image can take hours or days. Given how heavy spectral assets can be compared to RGB in both memory and disk, memory bottlenecks are common, making spectral assets unusable for real-time rendering.

Spectral Upsampling

There exist several techniques that aim to enable using RGB assets for spectral rendering, trying to get both the low difficulty of employing RGB assets and the accuracy of spectral rendering, while keeping the temporal costs as low as possible. Most consist in being able to recover spectral data from RGB triplets, via pre-optimized coefficients. The first work that employed optimization for this goal was the one by Smits [1]. Usually, spectral upsampling is performed only on reflectances, since it is an ill-posed problem and has barely been explored for volumetric coefficients.

Our real-time approach

We take Jakob and Hanika's work for reflectance upsampling from 2019 [2] as our starting point. There, three look-up tables are generated, with each containing three different coefficients, c_1, c_2, c_3 that are used by a second degree-polynomial and a sigmoid function to encode spectra in a smooth manner. We noticed that most of the necessary values were precomputed in tables that could be easily loaded with a small memory footprint, and the required operations could leverage vectorized shader instructions, like *fma* (full multiplication and addition). Thus, we adapted this technique for real-time rendering, using a minimal, custom rendering pipeline written in OpenGL. This pipeline is composed of 4 different passes. Two for deferred rendering, with the upsampling taking place in the second pass, one fully customizable (unused in our implementation), and the last one for post-processing, where we perform gamma correction. We get our wavelength samples in a naive way, equally spacing samples across a user-specified range, and later performing a middle Riemann sum.

Extending it with Underwater Rendering

To show that our work is compatible with other real-time spectral rendering methods and usable in industrial applications, we implemented an adapted version of Monzón et al.'s underwater rendering approach [3], where an approximation for multiple scattering is proposed, taking into account several Jerlov water types, a classification of natural water types depending on their optical properties, and different possible observer spectral sensitivities.

Results

Finally, we can confidently state that our method works in real-time (144 FPS for our testing environment), and introduces less error than both naive upsampling strategies and traditional RGB rendering when compared to Ground Truth path-traced images, as we show in Figures 1 and 2 (underwater), and Figure 3, with testing cases for extreme lighting setups, where metamerism comes into play. We used MAE (Mean Absolute Error) and CIE **Δ**E*2000 (a metric based on human perception) as our error metrics. We also characterized the linear influence of the number of wavelength samples on the framerate (See Figure 4).

Conclusions

We have successfully devised a way to perform spectral upsampling of reflectances in real time that can be used jointly with other techniques for spectral rendering, but several other works are waiting to be put out there, especially for volumetric coefficients and emissions. Eventually, we might get full upsampling to perform real-time spectral rendering with only RGB assets, getting the best of both worlds. This is just a small step toward that ambitious goal.

REFERENCES

[1]. SMITS, B. An rgb-to-spectrum conversion for reflectances. Journal of Graphics Tools, 4(4):11–22, 1999.

[2]. JAKOB, W., and HANIKA, J. A low-dimensional function space for efficient spectral upsampling. Computer Graphics Forum (Proceedings of Eurographics), 38(2), March 2019.

[3]. MONZON, N., GUTIERREZ, D., AKKAYNAK, D., and MUÑOZ, A. Real-time underwater spectral rendering. Computer Graphics Forum, page e15009, 2024

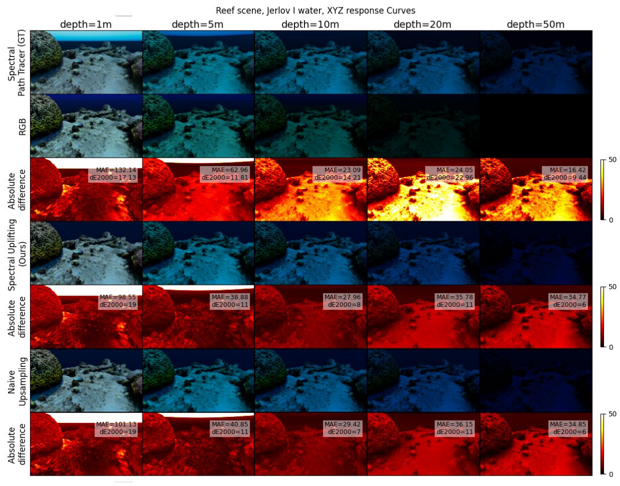
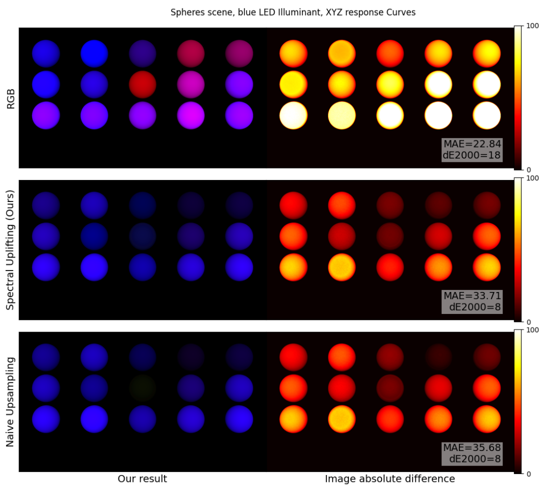


Figure 1: Image differences for several instances of an underwater scene. CIE **Δ**E*2000 and MAE error metrics included for each pair of Ground Truth and rendered images. Ours is in the middle row.



Spheres scene, blue LED Illuminant, XYZ response Curves
RGB
Spectral Uplifting (Ours)
Naive Upsampling
MAE=22.84 dE2000=18
MAE=33.71 dE2000=8
MAE=35.68 dE2000=8
Our result
Image absolute difference
100
0

Figure 3: Test scene for a highly spiked emission spectrum. Results on the left, image difference with path-traced GT on the right. CIE **Δ**E*2000 and MAE included. Ours is in the middle row.

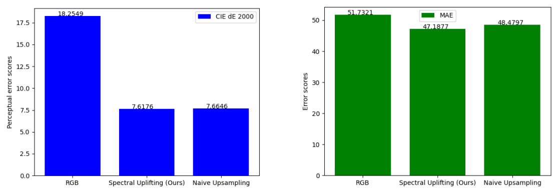


Figure 2: Mean error metrics for Figure 1. On the left, CIE **Δ**E*2000. On the right, MAE.

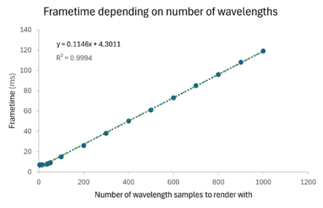


Figure 4: Frame time characterization with respect to the number of wavelength samples.