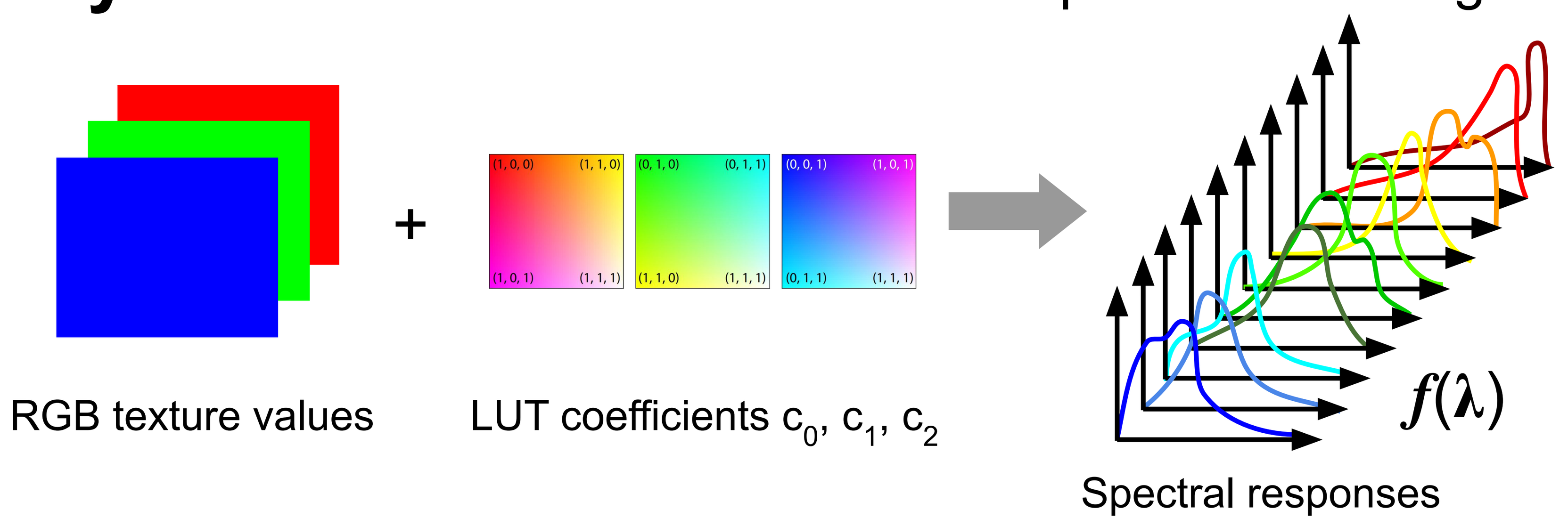
Techniques for Real-Time Spectral Rendering

Pedro Jose Perez[1] Nestor Monzon Adolfo Muñoz
Universidad de Zaragoza

- **RGB rendering** is the most common and popular way to generate images in Computer Graphics. It **discretizes** the entire visual spectrum into three colours: **Red**, **Green**, and **Blue**, matching our visual system. This traditional RGB rendering oftenly struggles simulating **wavelength-dependent** phenomena, such as iridescence or participating media scattering**, introducing error.**
- **Spectral rendering**, on the other hand, takes into account the **entire visual spectrum** (~400 nm to ~800 nm) to synthesize images, at the cost of having to deal with many **more samples**, slower than compared to just three colours.
- While good for **offline rendering** (VFX, marketing, architecture, etc.) where time constraints are not relevant, spectral rendering's need for more samples introduces **prohibitive costs** in both memory usage and computational time for **real-time rendering** (videogames, virtual reality, etc.), where 30-120 images need to be generated **every second**.
- We devise a method that enables the usage of spectral rendering in real-time contexts minimally affecting time and memory consumption by adapting previously existing techniques, such as **reflectance upsampling,** into real-time contexts. We also prove its **compatibility** with similar rendering methods.

Key idea: Can we use RGB assets in spectral rendering?



We take the work of Jakob and Hanika (2019) as our starting point. They **upsample the entire sRGB gamut into spectral responses.**

$$f(\lambda) = S(c_o\lambda^2 + c_1\lambda + c_2)$$
$$S(x) = \frac{1}{2} + \frac{x}{2\sqrt{1+x^2}}$$

```
function f(c0, c1, c2, wl)
   # Evaluate polynomial for wavelength 'wl'
   x = fma(fma(c0, wl, c1), wl, c2)

   # Evaluate nonlinear map
   return fma(.5 * x, rsqrt(fma(x, x, 1)), .5)
```

They achieve so via **lightweight look-up tables** (~6MiB) and cheap formulas that require **only 6 floating-point operations.** We recognized the potential that their technique had for real-time contexts.

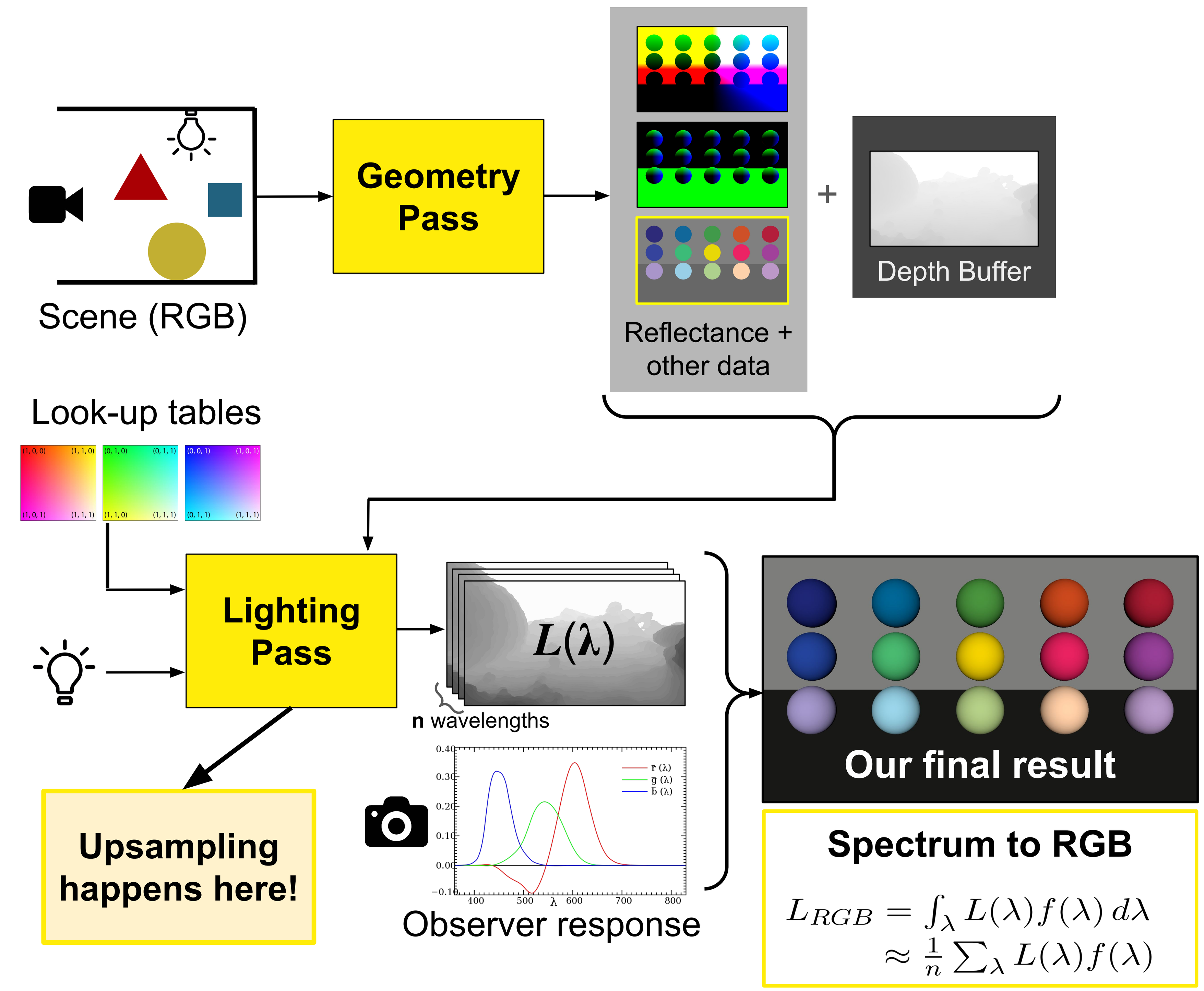
GITHUB REPO

SLIDES

FULL THESIS

Our implementation

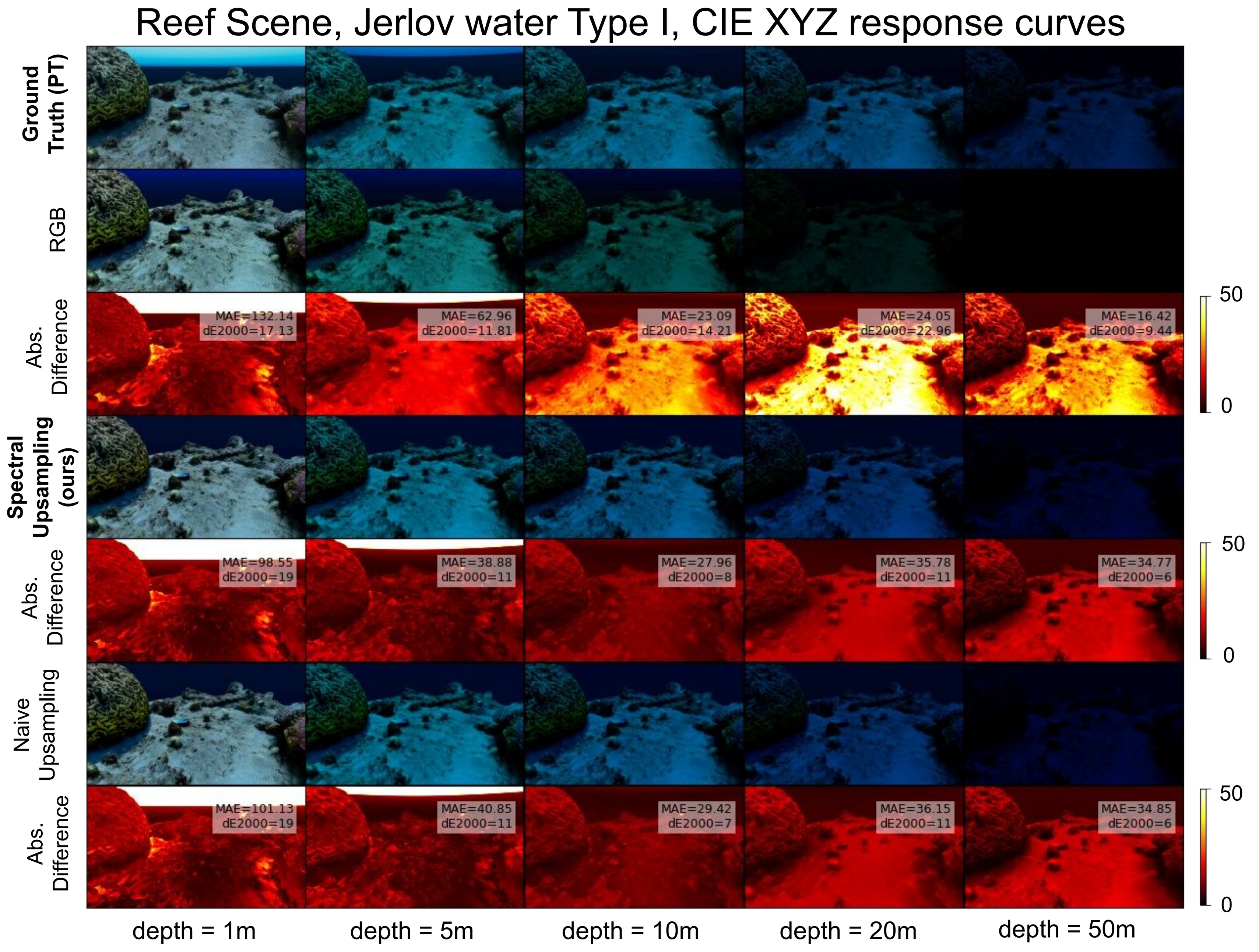
We implement our real-time version as an OpenGL renderer from scratch. We decided to implement a **multipass deferred pipeline**, illustrated here:



Additionally, we support the simulation of several **observer sensitivity curves**, being able to simulate several **camera types**, or the **human eye** observer. Same applies to lights, we can **load real world data** for the Spectral Power Distribution **(SPD)** of several measured light emitters.
An adaptation Monzon et al.'s underwater rendering method is supported too.

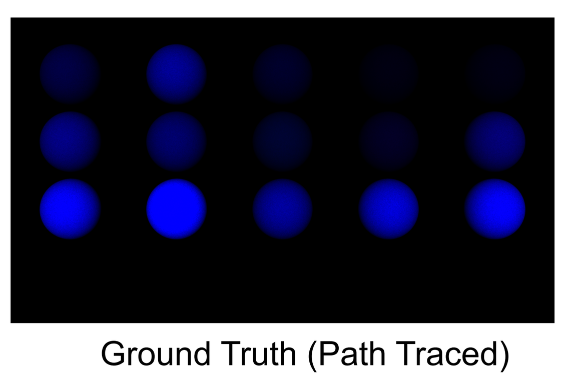
Results and evaluation

Reef Scene, Jerlov water Type I, CIE XYZ response curves

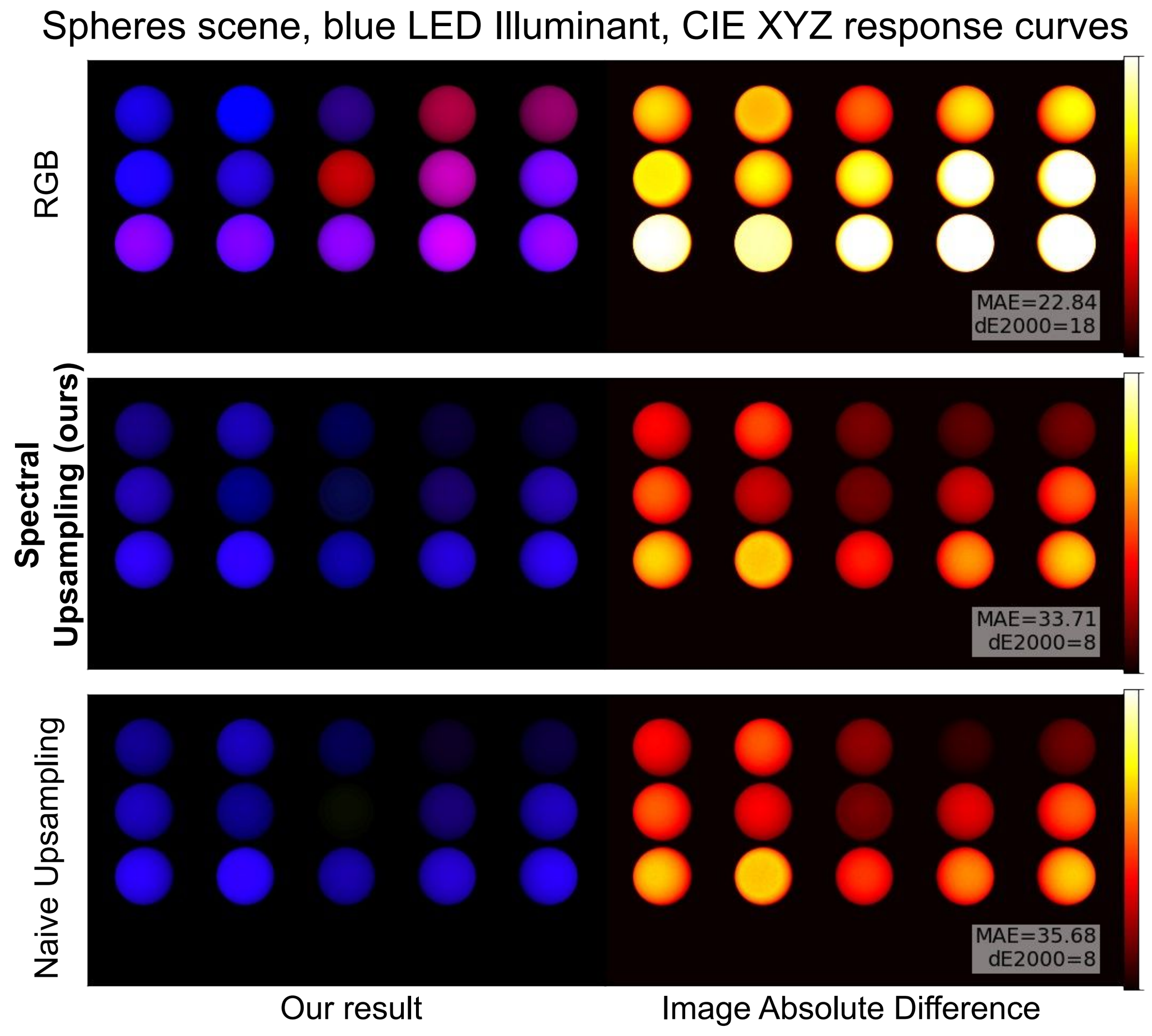


- **We improve error** with respect to baseline comparisons in scenes with **wavelength-dependent phenomena (left)** and scenes with **highly spiked emission spectra (below).**
- We **use RGB** reflectance textures for **spectral rendering!**
- **120+ FPS** for all tested use cases, real-time ready.

Spheres scene, blue LED Illuminant, CIE XYZ response curves



Ground Truth (Path Traced)



Future work: Upsample correctly other coefficients to fully achieve an RGB material→Spectral result workflow.

ACKNOWLEDGEMENTS: This work has received funding from the Government of Aragon's Departamento de Ciencia, Universidad y Sociedad del Conocimiento through the Reference Research Group "Graphics and Imaging Lab" (ref T34_23R).
Néstor Monzón was supported by a Gobierno de Aragón predoctoral grant (year 2023-2027).

[1]: p.perez@unizar.es