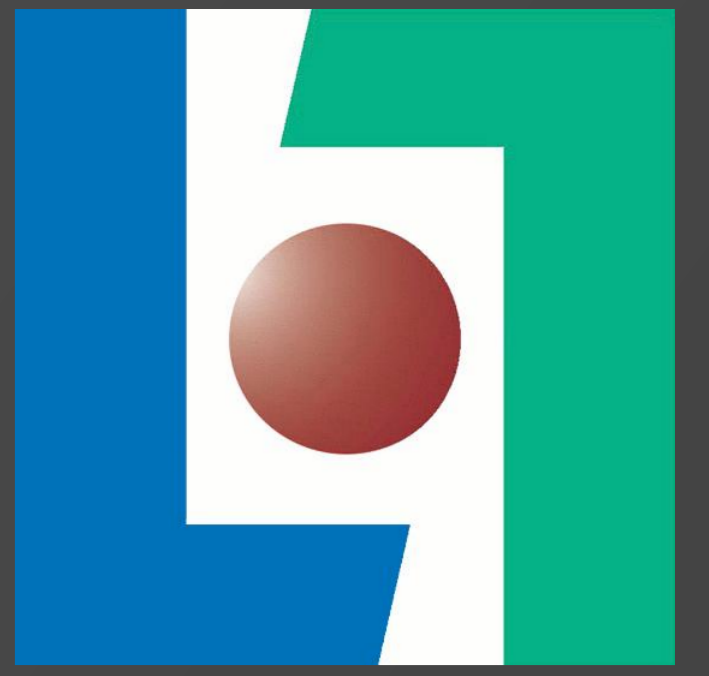




« Hash Tree Textures »



EGSR 2010

Jérôme Baril and Christophe Schlick

University of Bordeaux – LaBRI / INRIA - IPARLA Team

Overview

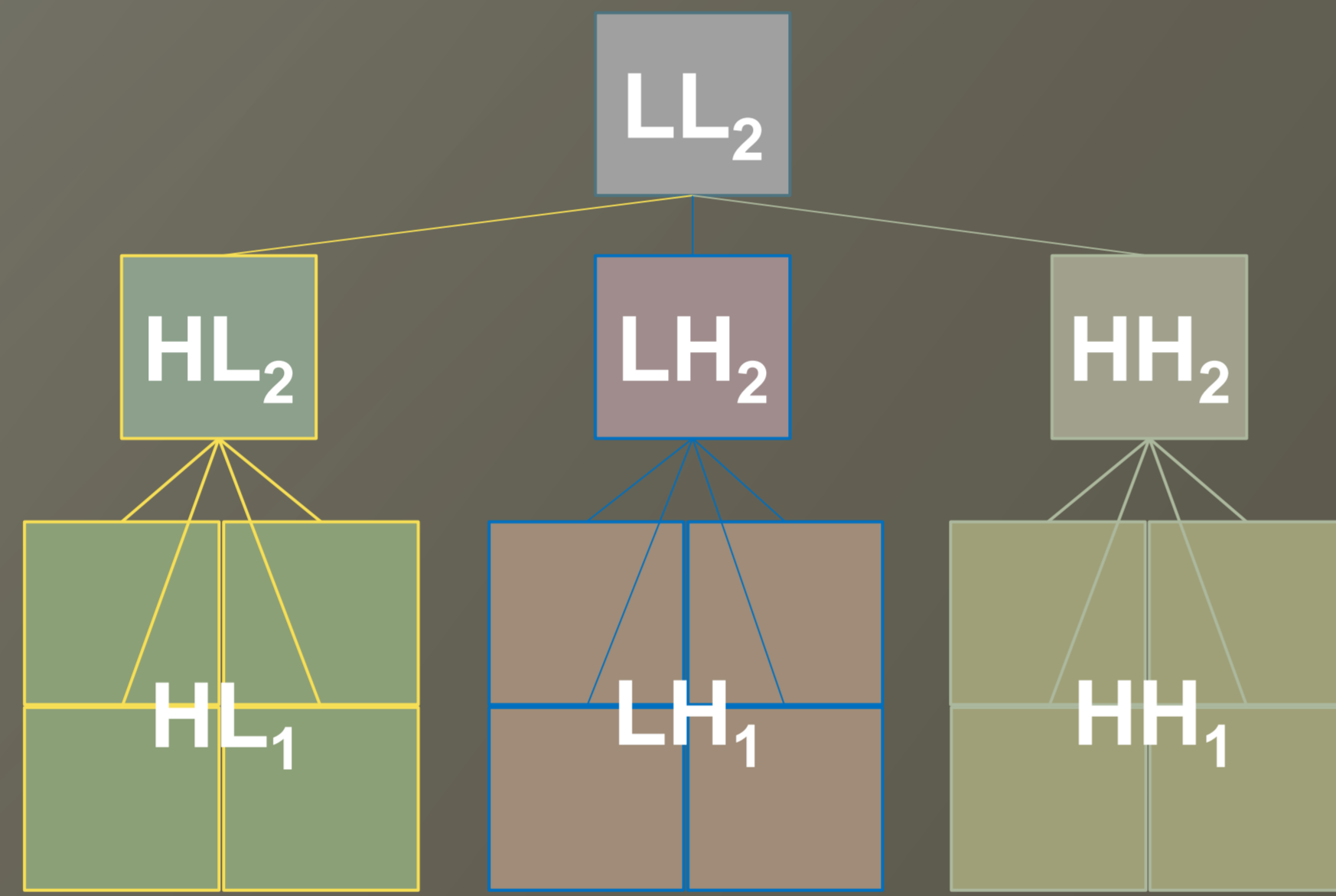
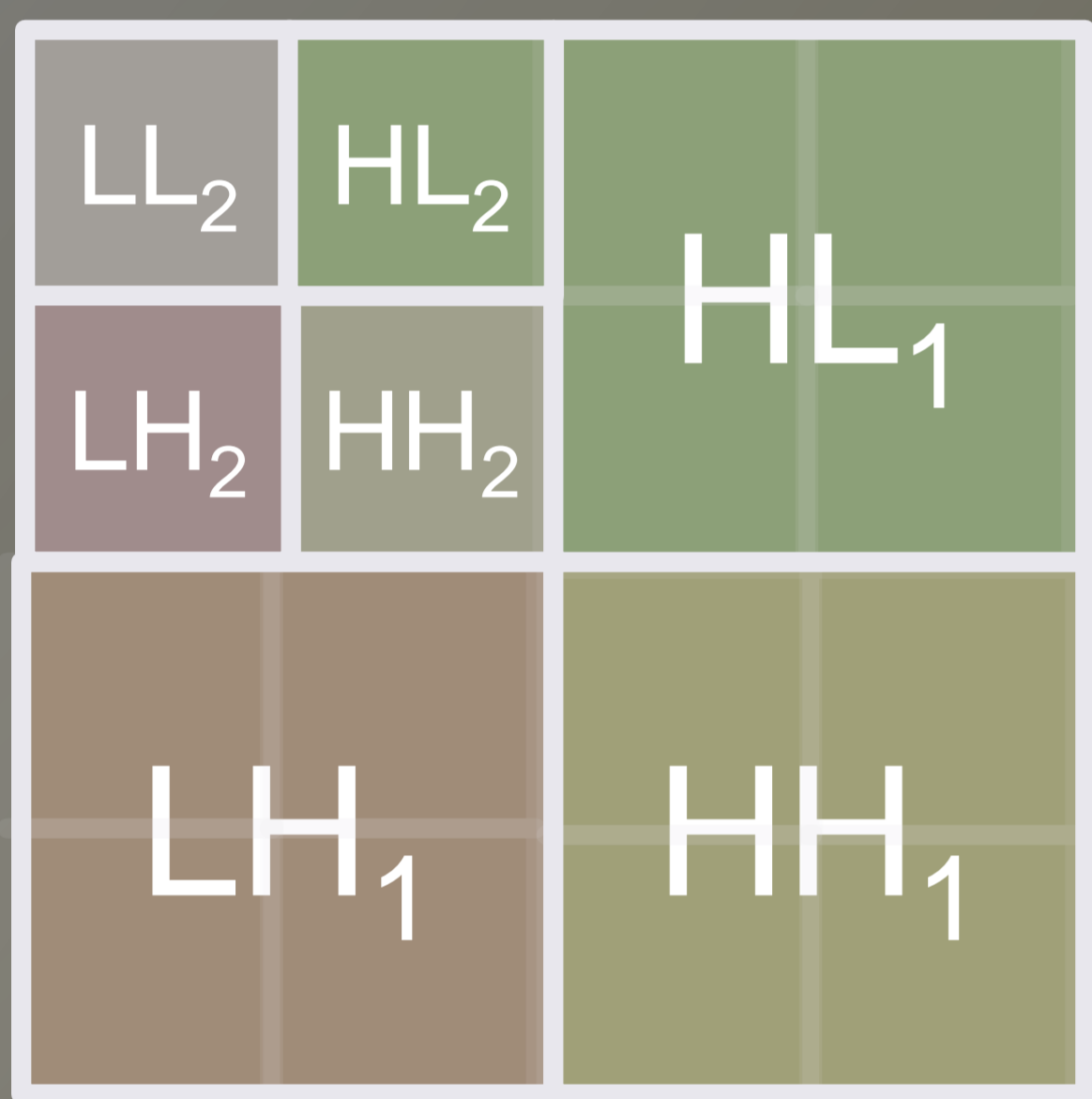
The work presented here tries to built a bridge between CPU and GPU texture compression methods, by proposing some innovative ways to import several principles used by the better CPU-based image compression techniques into the world of GPU-based texture compression. The keystone of our approach is to use hashing techniques as a general-purpose tool for coding arbitrary bitcodes. Hash Tree Textures can thus be seen as a generalization of the EZW to arbitrary number of dimensions, combined with an efficient GPU-friendly data structure based on hashing tables. Our work has basically tried to identify important features that make ZeroTree coding work so well and find a equivalent process that offers similar features, but with a bitcode that can be transformed into a GPU-friendly data structure.

Input Textures

Any kind of multi-dimensionnal textures (RGB, Normal Map, PTM, RNM, SVBRDF, BTF, ...).

Discrete Wavelet Transform

The DWT projects spatial coefficient into a set of sub-bands (LL_i, HL_i, LH_i, HH_i). Each sub-band is an oriented scale space with sparse floating-point values.



Wavelet Texture Space

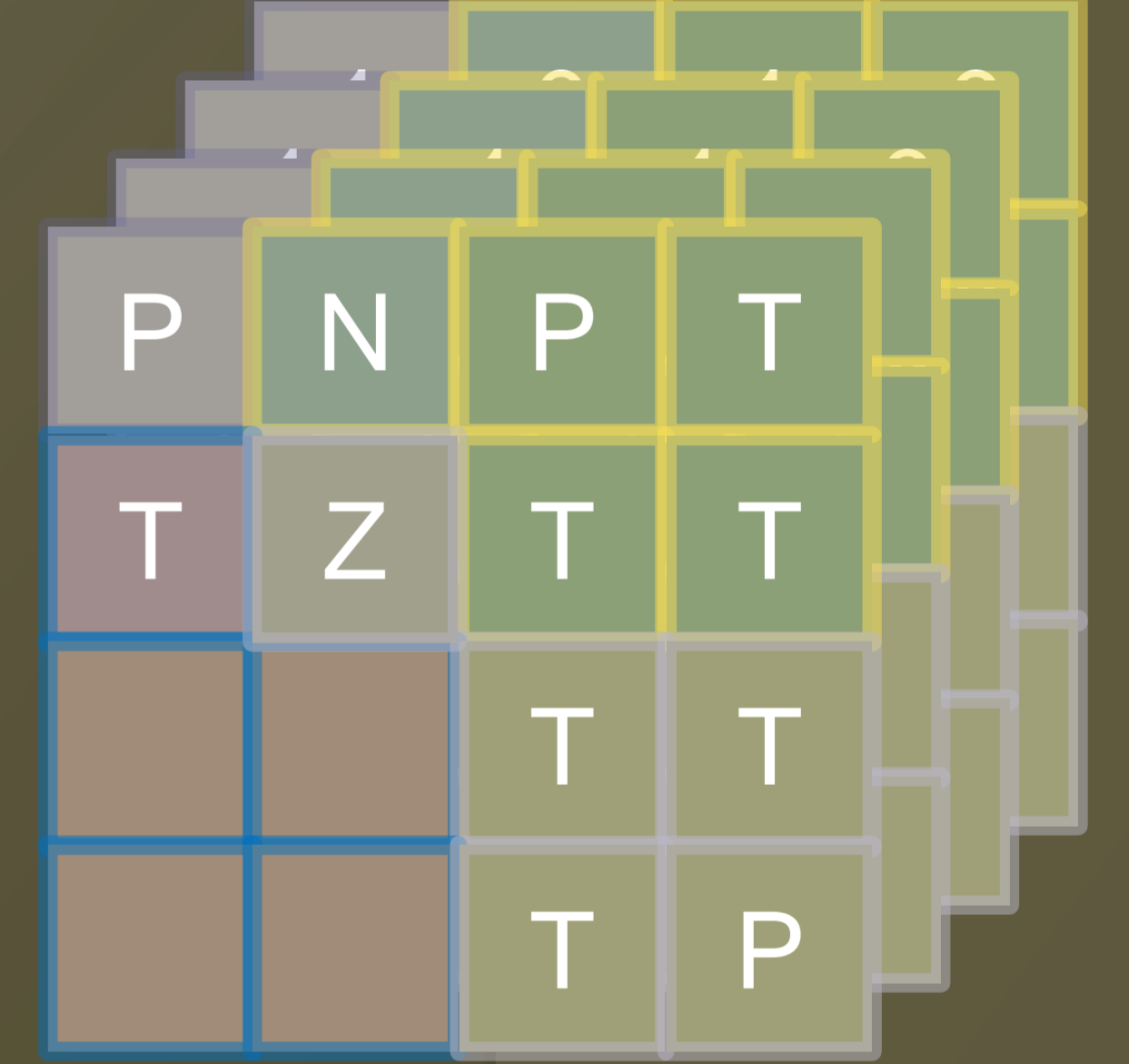
Wavelet Hierarchy

Successive Approximation Quantization

The SAQ is a binary finite-precision representation of a dataset by iteratively applying a sequence of thresholds. The process is repeated n times until the precision of the approximation is considered as good enough. The quality of the approximation is computed using **SSIM**, a local quality metric, useful for **ROI**. The SAQ is then tagged with { P, N, Z, T } symbols such as ZeroTree coding bitcode.

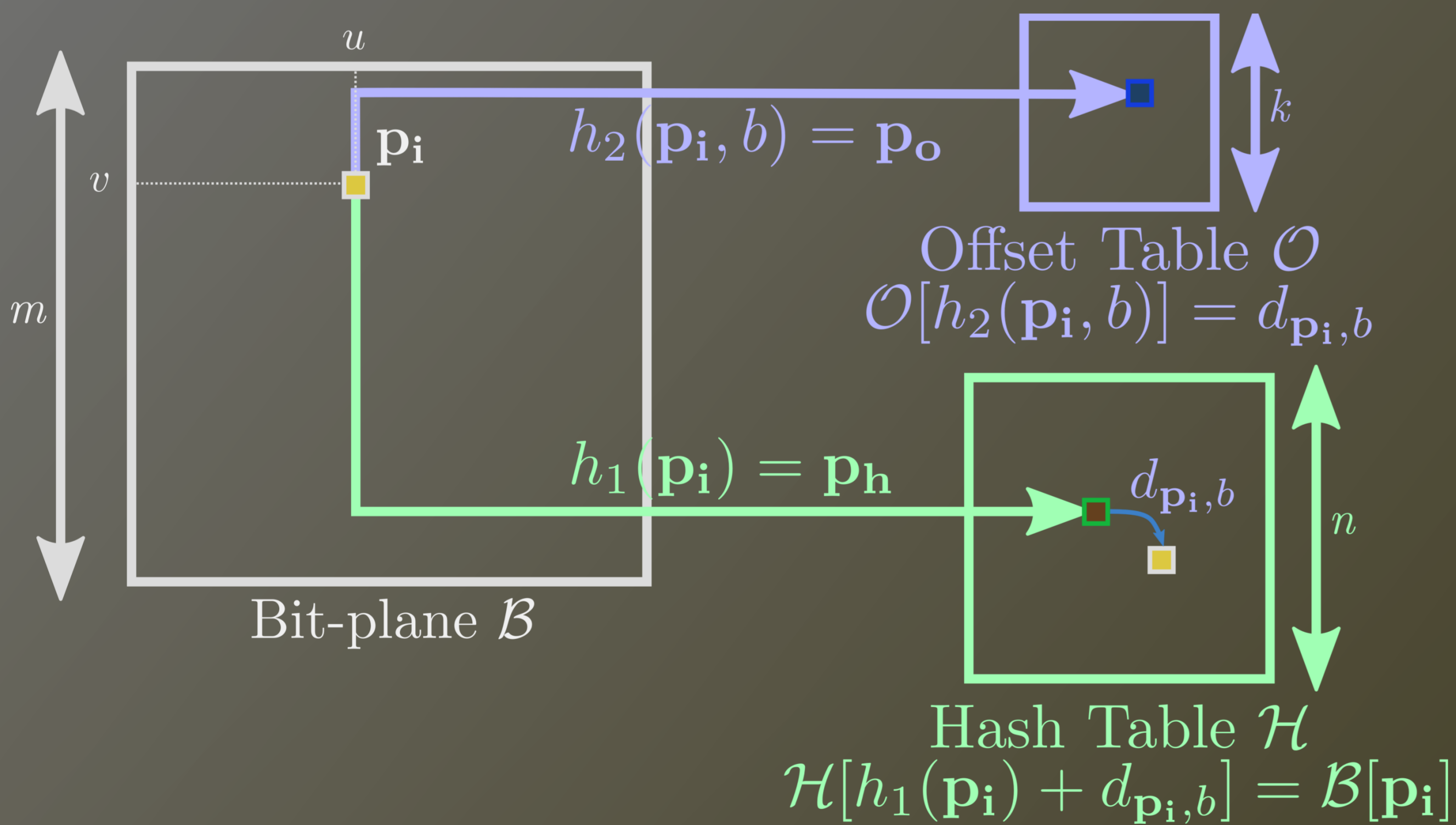


Bitplane Quantization



Tagged Tree Texture

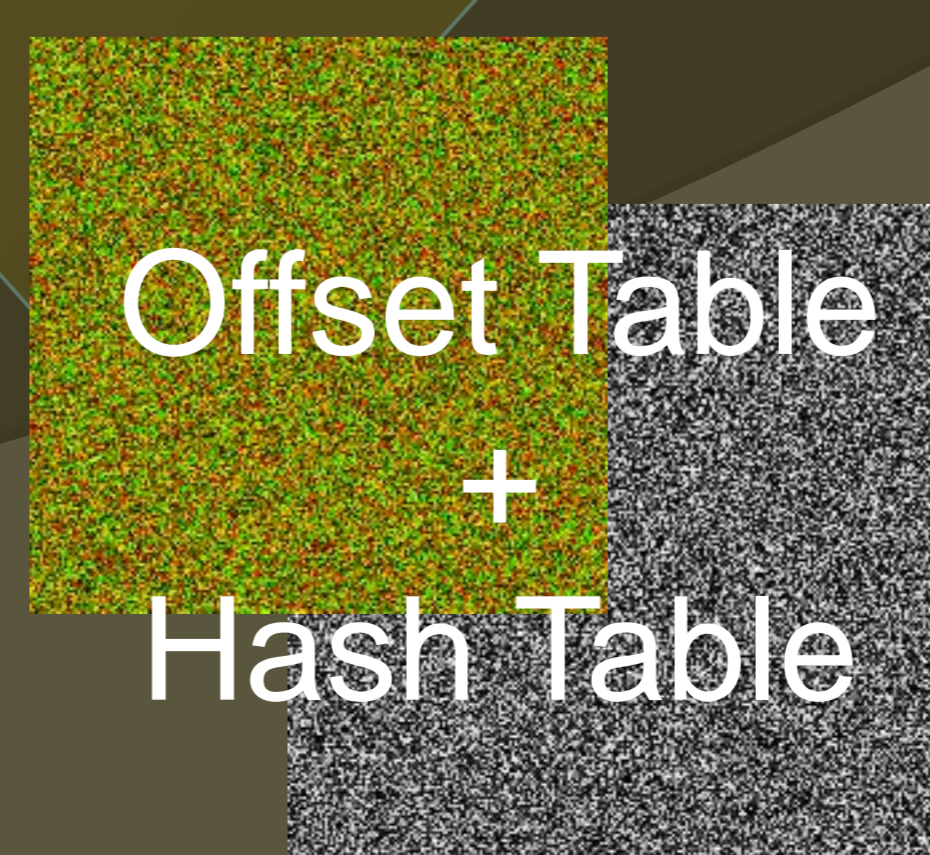
ImPerfect Spatial Hashing



Perfect spatial hashing is a way to store sparse data into a dense data structure called *hash table*. To avoid collision in the *hash table*, an *offset table*, a shared 2D offset value, is used. The value associated with **each key is one of the four symbols PNZT**. Therefore, there is a high probability that the values associated to two different keys are actually the same. Having a collision between these two keys is thus not a problem, it is even a highly desirable side-effect as it avoids to store the same value at another place in the table. We call them **friendly-collisions** for the ImPerfect Spatial Hashing. Our scheme can be **global** for **high compression ratio** or **local** to maintain **cache coherency** during GPU rendering.

GPU Rendering

The rendering process consists in **traversing the bitplane hierarchies** in the *hash table* to **regenerate the values of the wavelet coefficients**. Then, the inverse wavelet transform is applied to rebuild the pixel value at a given **Level Of Detail**.



Fragment Shader

