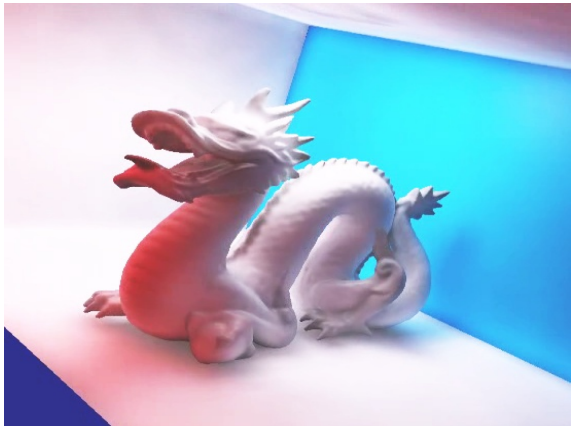
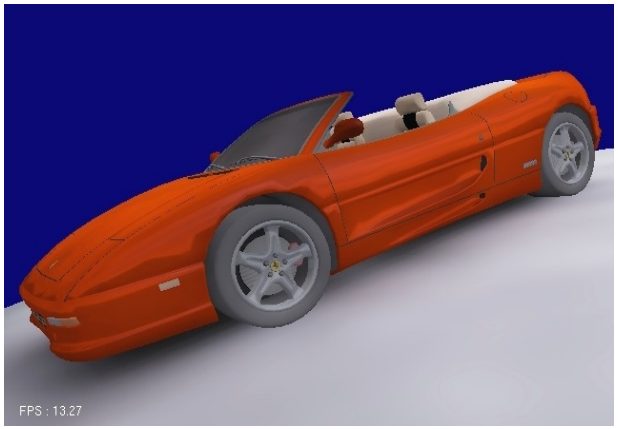
Light Octree : global illumination reconstruction for realtime navigation

Vincent Vivanloc, Jean-Christophe Hoelt, Coong Binh Hong and Mathias Paulin
{vivanloc,hoelt,paulin}@irit.fr

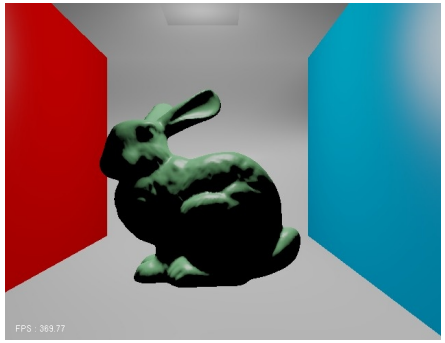
Institut de Recherche en Informatique de Toulouse
Université Paul Sabatier Toulouse III – France

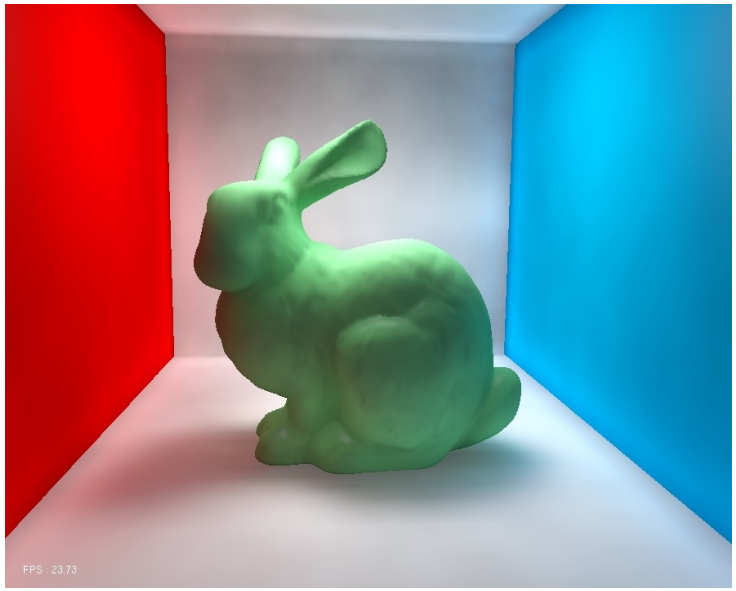


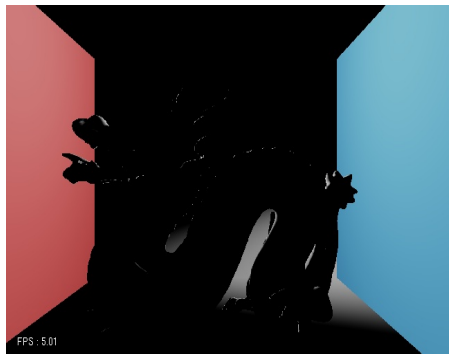


Layout

- Introduction
- State of the art
- Implementation
- Results
- Conclusion







OpenGL

Light Octree

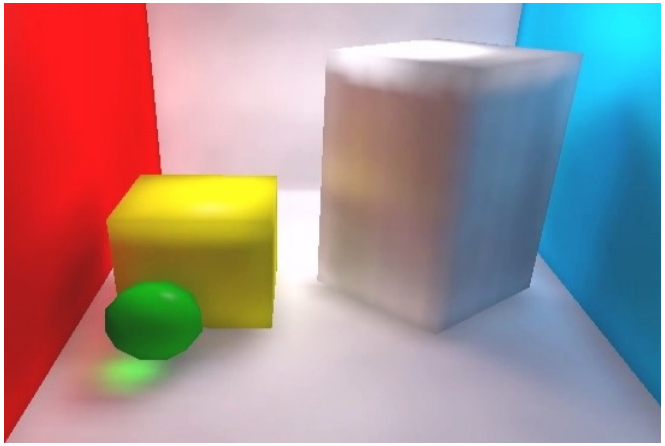
Purpose

Provide a real-time navigation
in a complex global illuminated environment

- a lot of polygons
- all-frequency lighting

Ferrari F355 spider (500k polys)

Renault Megane (1.5M polys)



Cornell box (50 polys)

Creating a GI cache

Radiance cache = 5-dimension scalar field

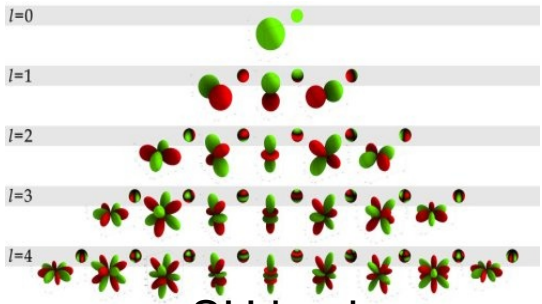
- compression is needed

– How to compress ?

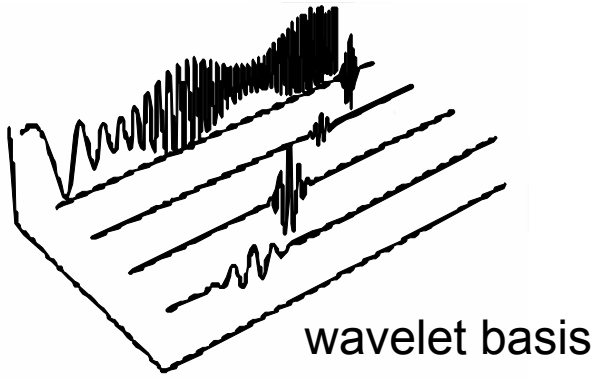
- continuous fct basis: SH[SKS02],wavelets [WTL06]
- discrete clusters virtual lights [Kel97,ZSP98]

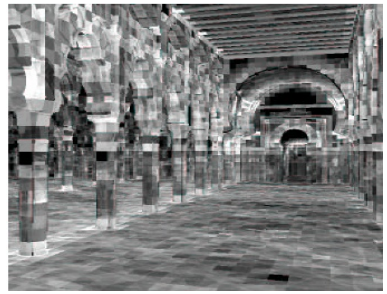
– Where to compress ?

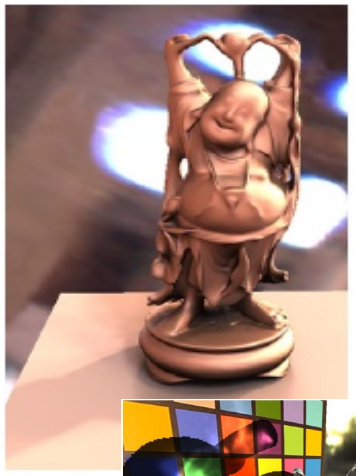
- assume lights have only a local influence [FBG02]
- per pixel, vertex, photon, voxel or mesh ?



SH basis


wavelet basis


[FBG02]



Rendering a GI cache

World GI cache

- per pixel (offline): ✓ hi quality ✗ final gathering
- per vertex (PRT): ✓ hi quality ✗ expensive
- splats: ✓ localised/low freq ✗ overdraw

Local GI cache

- environment maps: ✓ hi perf ✗ visibility constraints
- OpenGL lights: ✓ legacy ✗ precomputation
- per voxel: ✓ no parametrization ✗ not yet on GPU

[LSSS04,WD06,LP03,SKALP05]

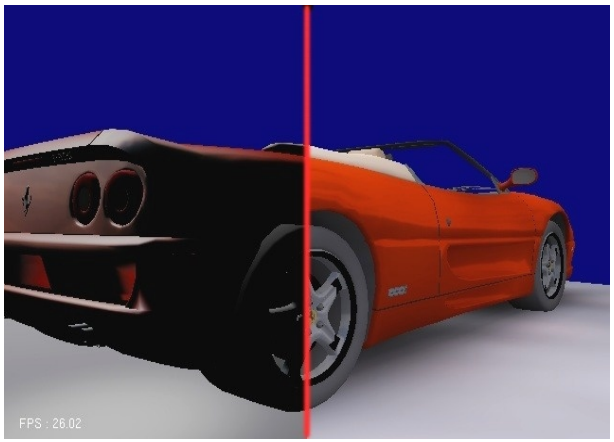
Contribution

Purpose

- Provide a real-time navigation in a *complex global illuminated environment*

Our contribution

- Light octree : a GI cache for GPU
- Fast simplification of a photon map
- Suitable for GPU Real time rendering

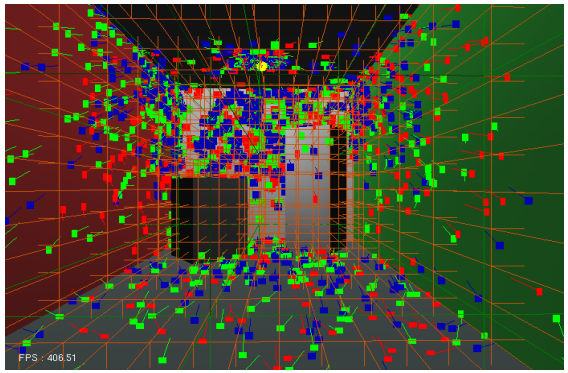


Ferrari F355 spider (500k poly)

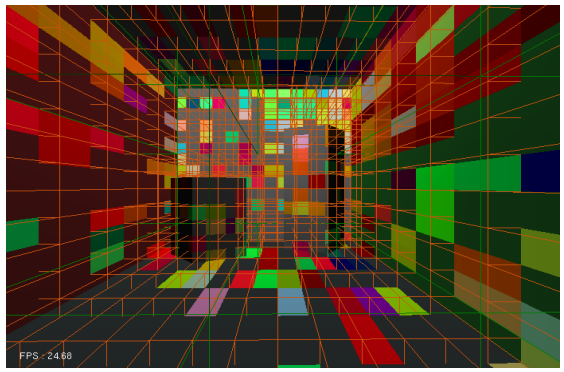
Renault Megane (1.5M poly)

Approach

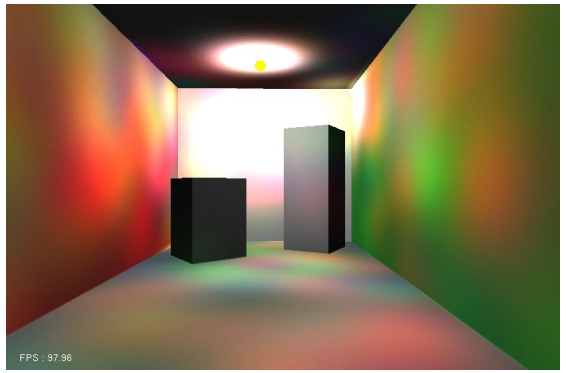
- Per voxel local illumination environment
- Clustering into virtual directional lights (VDL)
- VDL integrated on GPU



CPU :photon casting

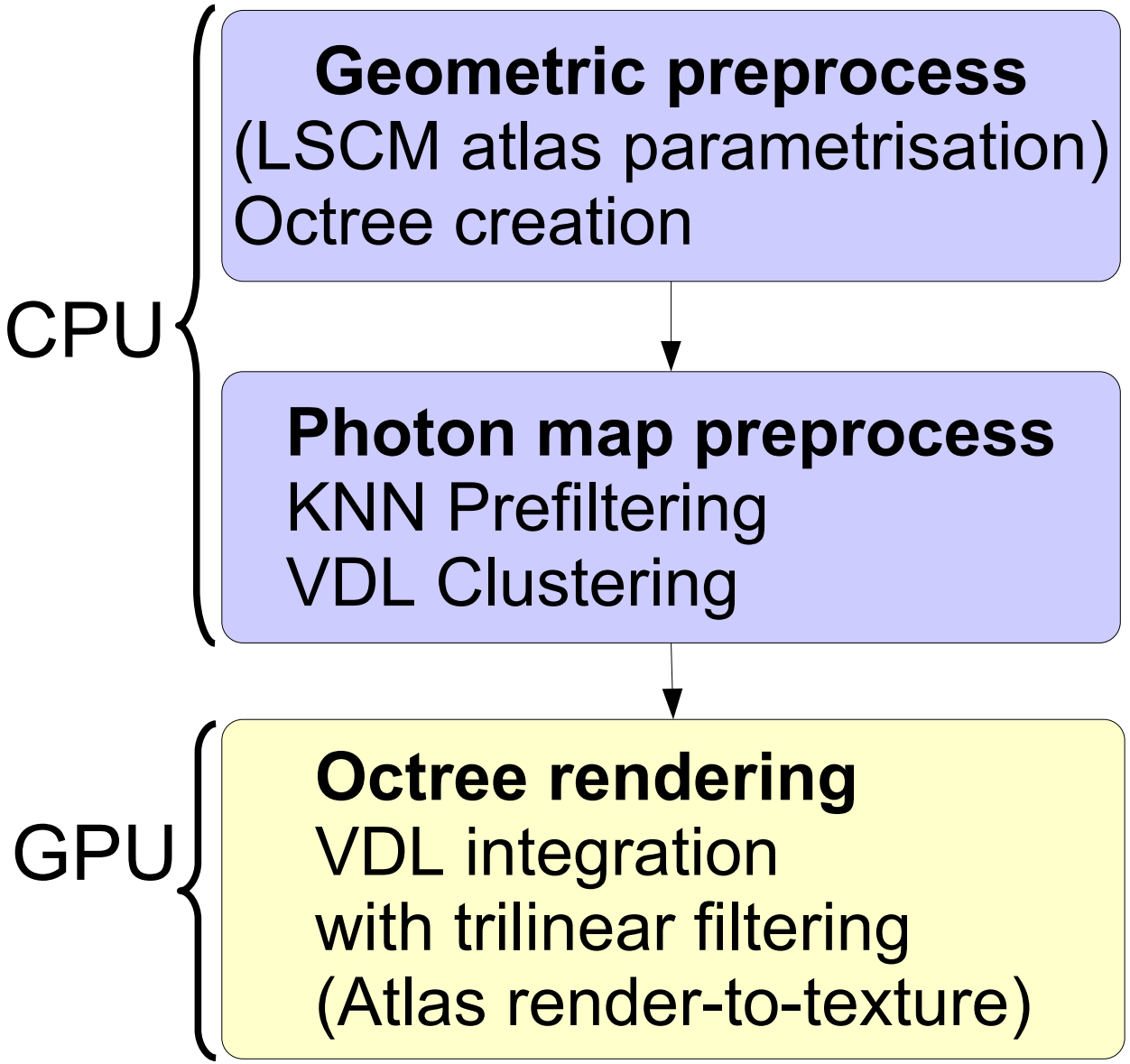


CPU: photon clustering into VDLs

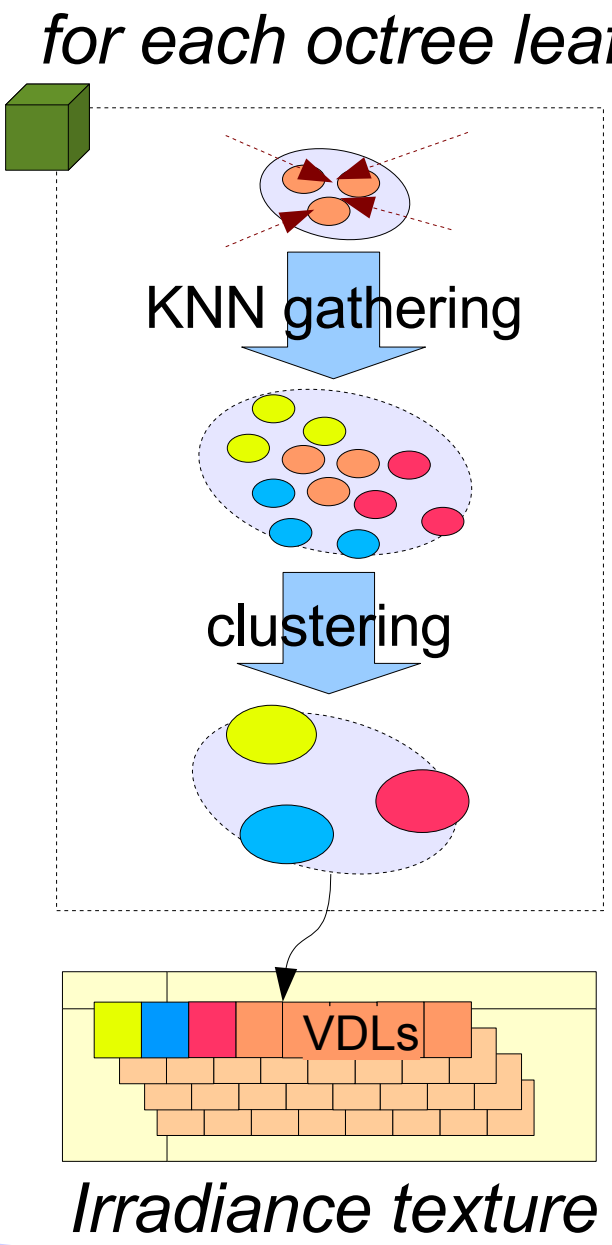


GPU : VDLs integration

Pipeline


CPU
Geometric preprocess
(LSCM atlas parametrisation)
Octree creation
Photon map preprocess
KNN Prefiltering
VDL Clustering
GPU
Octree rendering
VDL integration
with trilinear filtering
(Atlas render-to-texture)

Photon map preprocess



per voxel

create irradiance discrete clusters from a photon map

KNN gathering

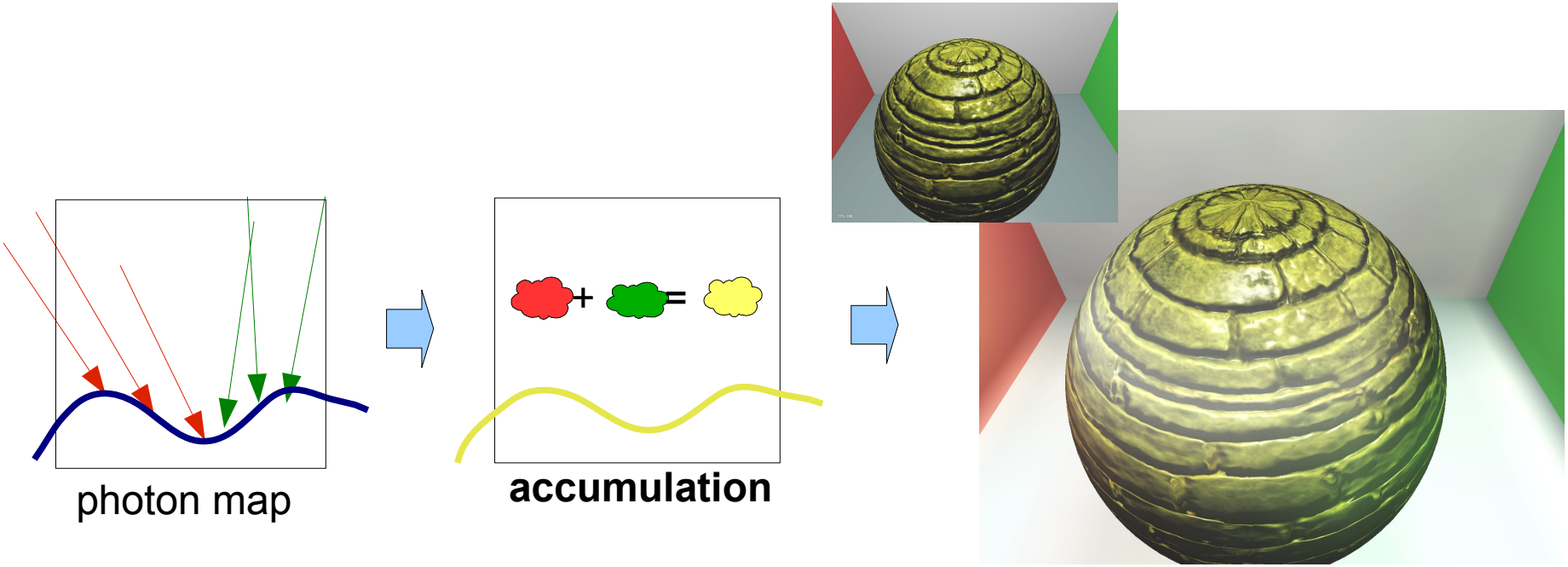
density estimation
\+ space voxelization
compulsory KNN

Clustering

PCA, Hclust, Kmeans ?

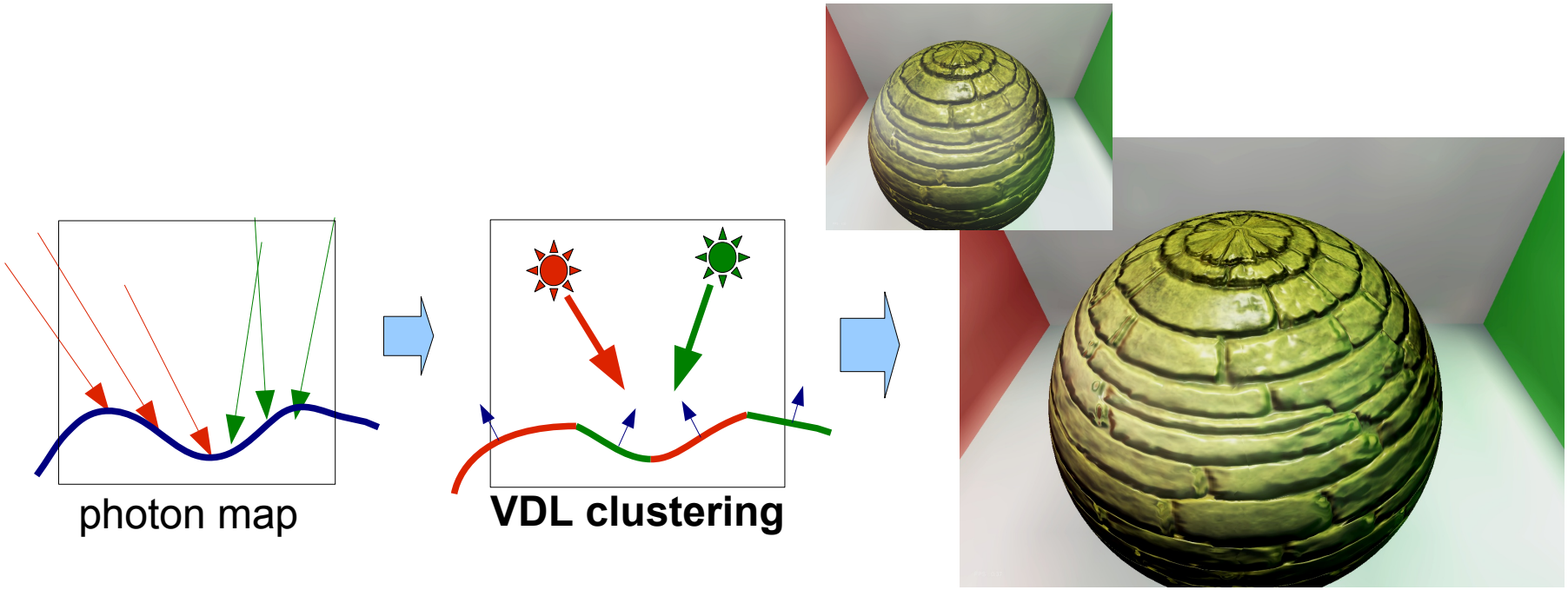
A GI cache on GPU

- based on Octree Textures on the GPU [LHN05]
- provides a compact data structure
- color octree
  - per voxel lighting : one color per voxel is not enough



Virtual directional light

- light octree
  - per pixel lighting : add a directional information to color
- VDL efficiently coded into a 4x16bit texel (R,G,B,{θ,Φ})
- up to 8 VDLs per voxel using an irradiance texture

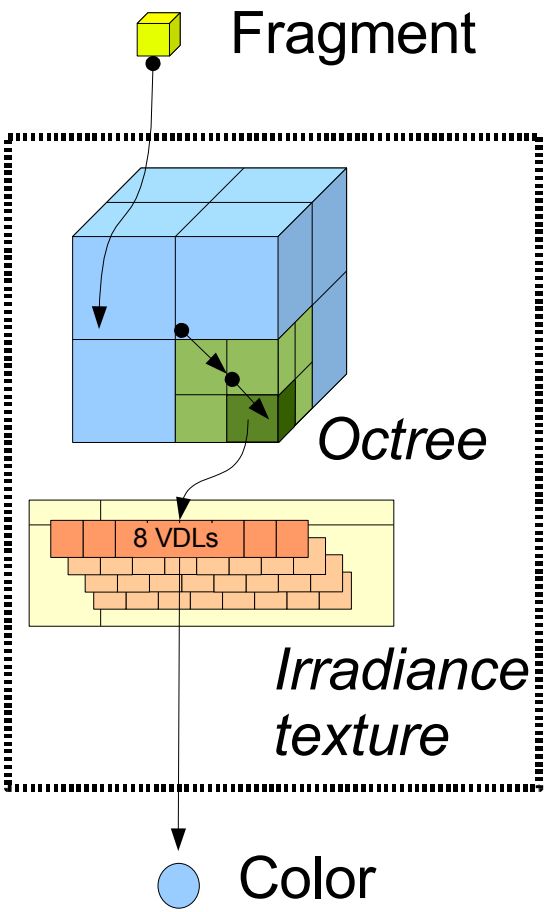


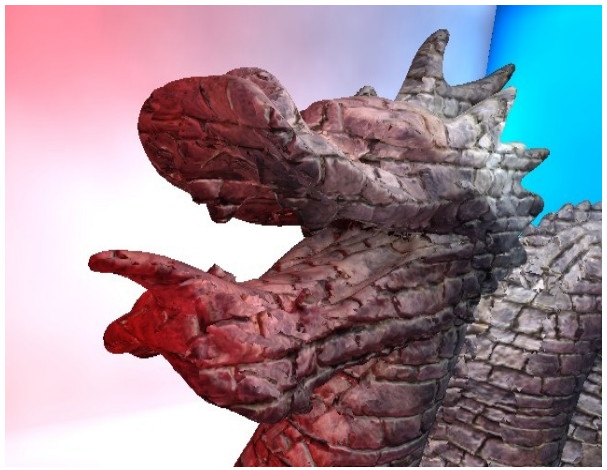
Rendering VDL on a GPU

- VDL as an OpenGL directional light

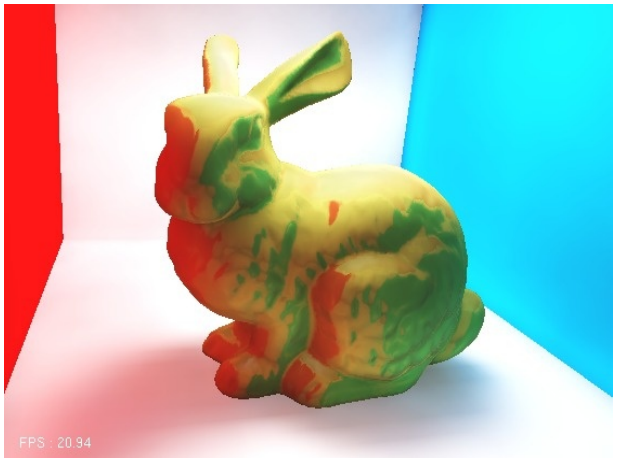
$$\sum_{i=1}^{nVDL} brdf(p, n, d_i).c_i$$

- easily integrated with legacy renderer





parallax bump mapping

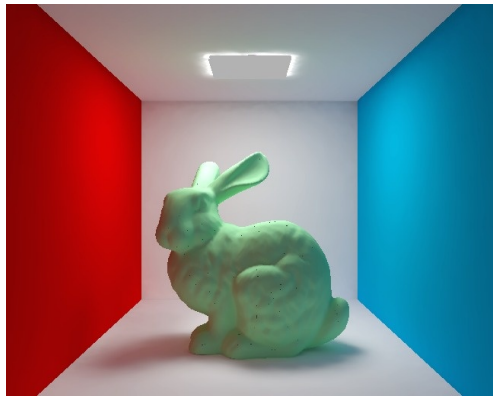


environment mapping

Results

Visual aspect

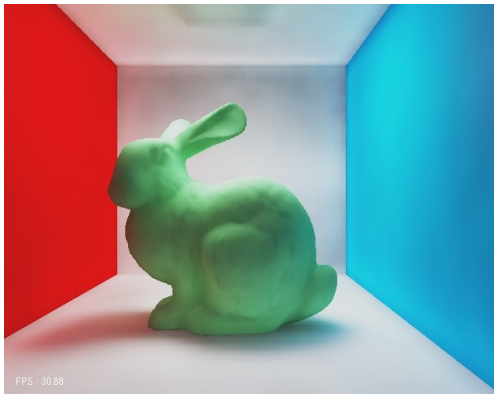
- Cannot compete with offline rendering
- but provides a good approximation (minus octree aliasing)
- integrable with legacy shaders



Yafray : 5 mins per frame

Timing

- Offline processing on CPU
  - not fast enough for dynamic scene but highly parallelizable
- Rendering on GPU
  - 40 to 10 fps (viewport size)

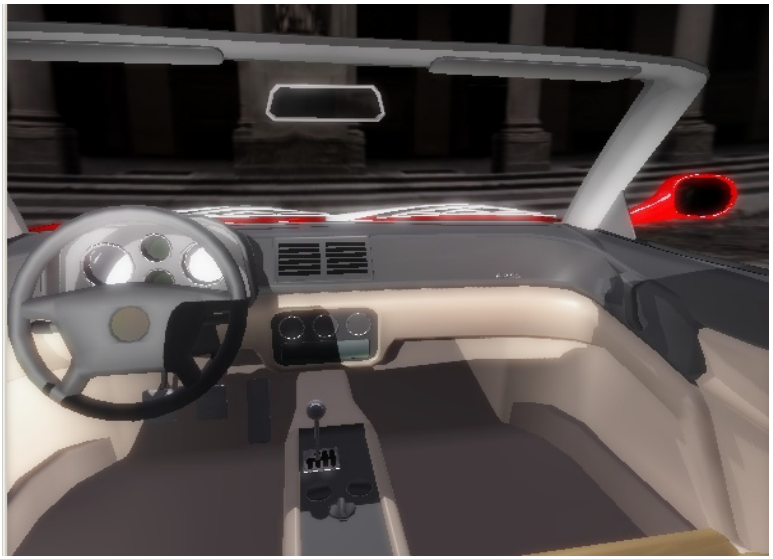


Light Octree : also 5 mins of pre processing... but navigation at 30 fps

Conclusion

Light Octree

- ✔ pros
  - parallelizable cache compression
  - compact representation on GPU
  - integrated to legacy shaders
- ✘ cons
  - octree aliasing
  - not (yet ?) ready for dynamic scene



Light Octree integrated with shadow map, bloom and image based lighting

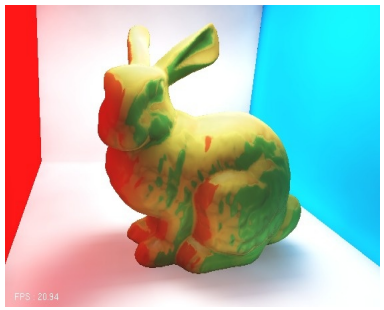
Future works

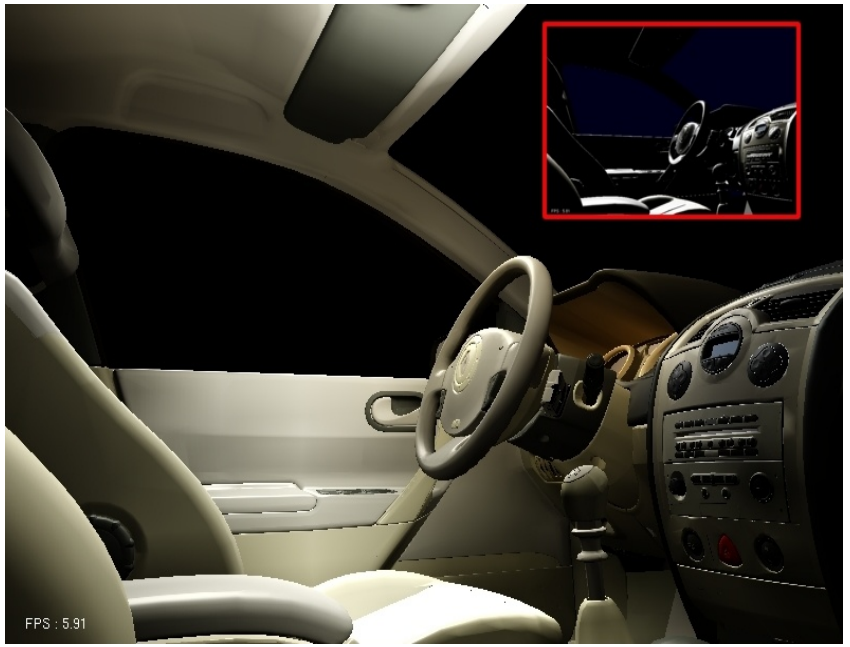
- per texel computation rather than per voxel
- incremental update of light octree

Questions ?

Light Octree

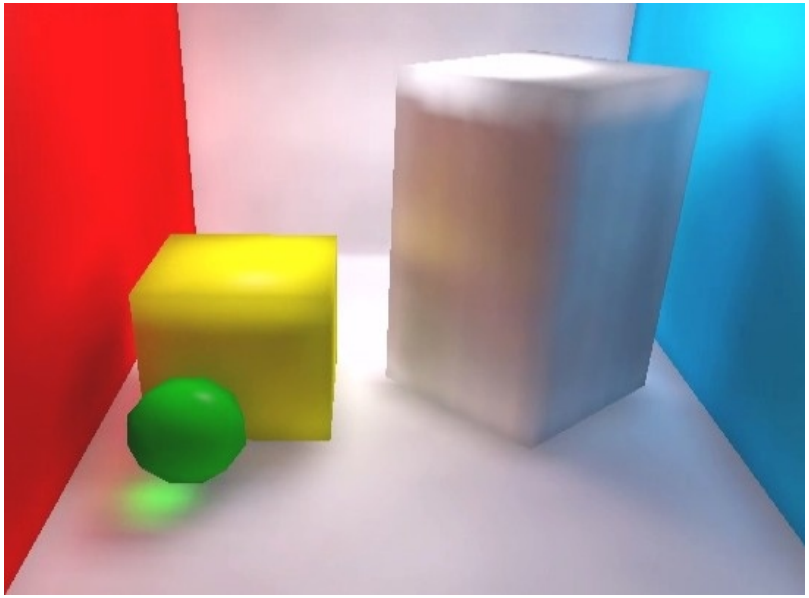
- ✔ pros
  - parallelizable cache compression
  - compact representation on GPU
  - integrated to legacy shaders
- ✘ cons
  - octree aliasing
  - not (yet ?) ready for dynamic scene

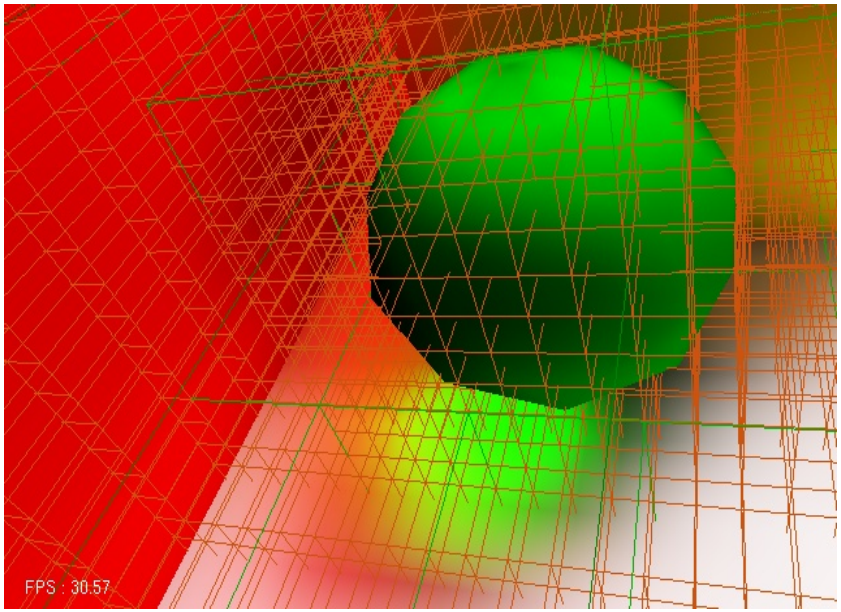




Octree construction

- geometric subdivision criteria
- leaves have always the same volume





Limitations

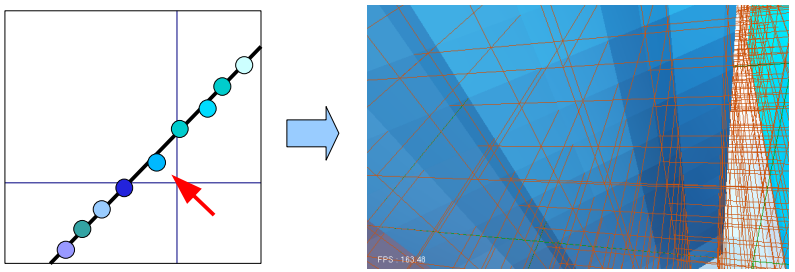
1)Limited number of VDLs

2)Voxel size matters

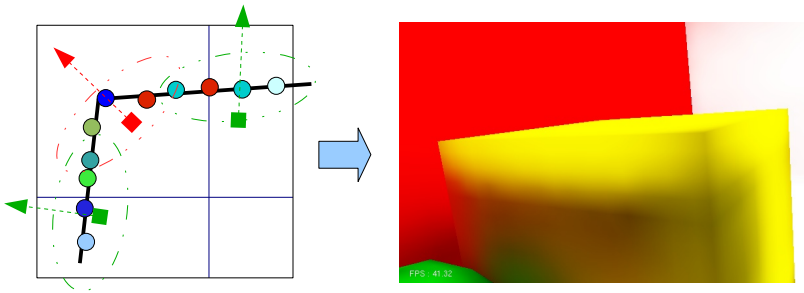
3)Octree resolution

4)Octree aliasing

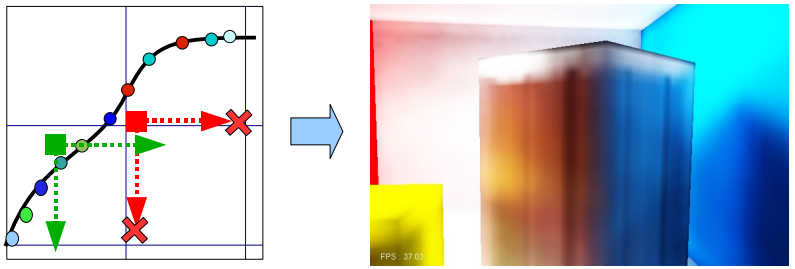
- Density underestimation
- Color overbleeding
- Banding artifacts



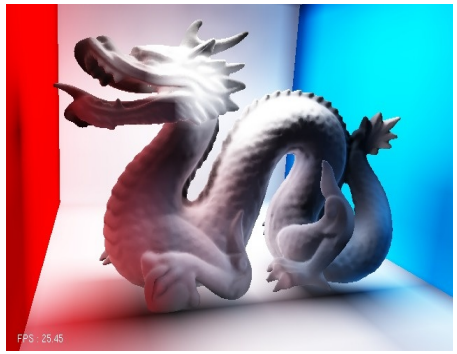
density understimation (a)



color overbleeding (b)



interpolation holes (c)

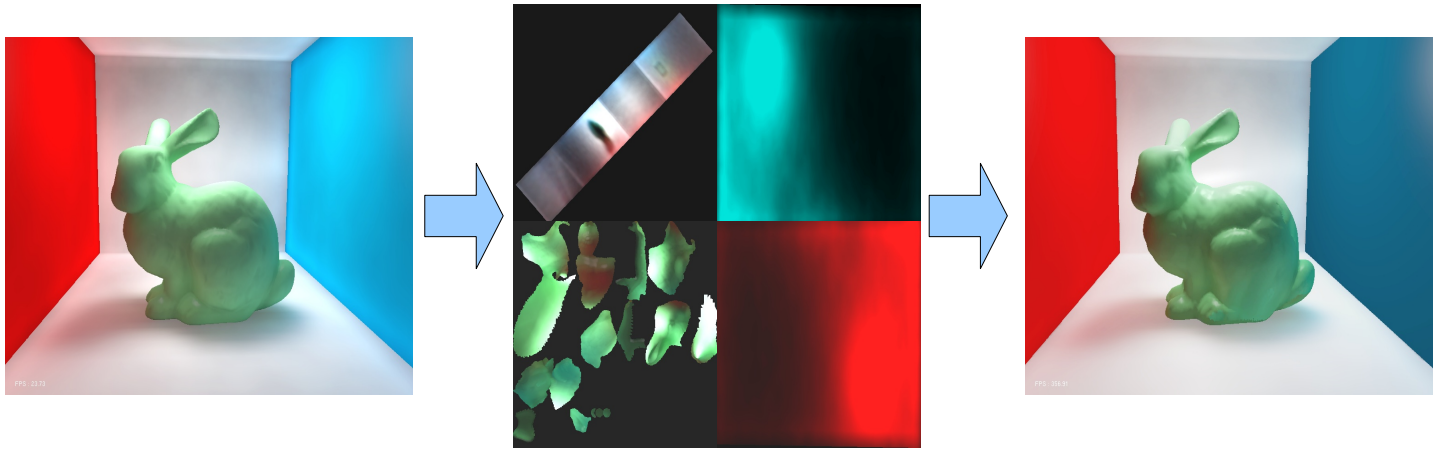


depth=4

depth=6

Rendering : Render to altas

- Create enhanced light maps
  - Improve the framerate
  - For legacy GPU
- Build a LSCM parametrization [LPRM02]
- Render : 2 pass algorithm
  - Render light octree into an texture atlas (one time)
  - Apply this atlas as a light map



Render to atlas

Apply as tex2D