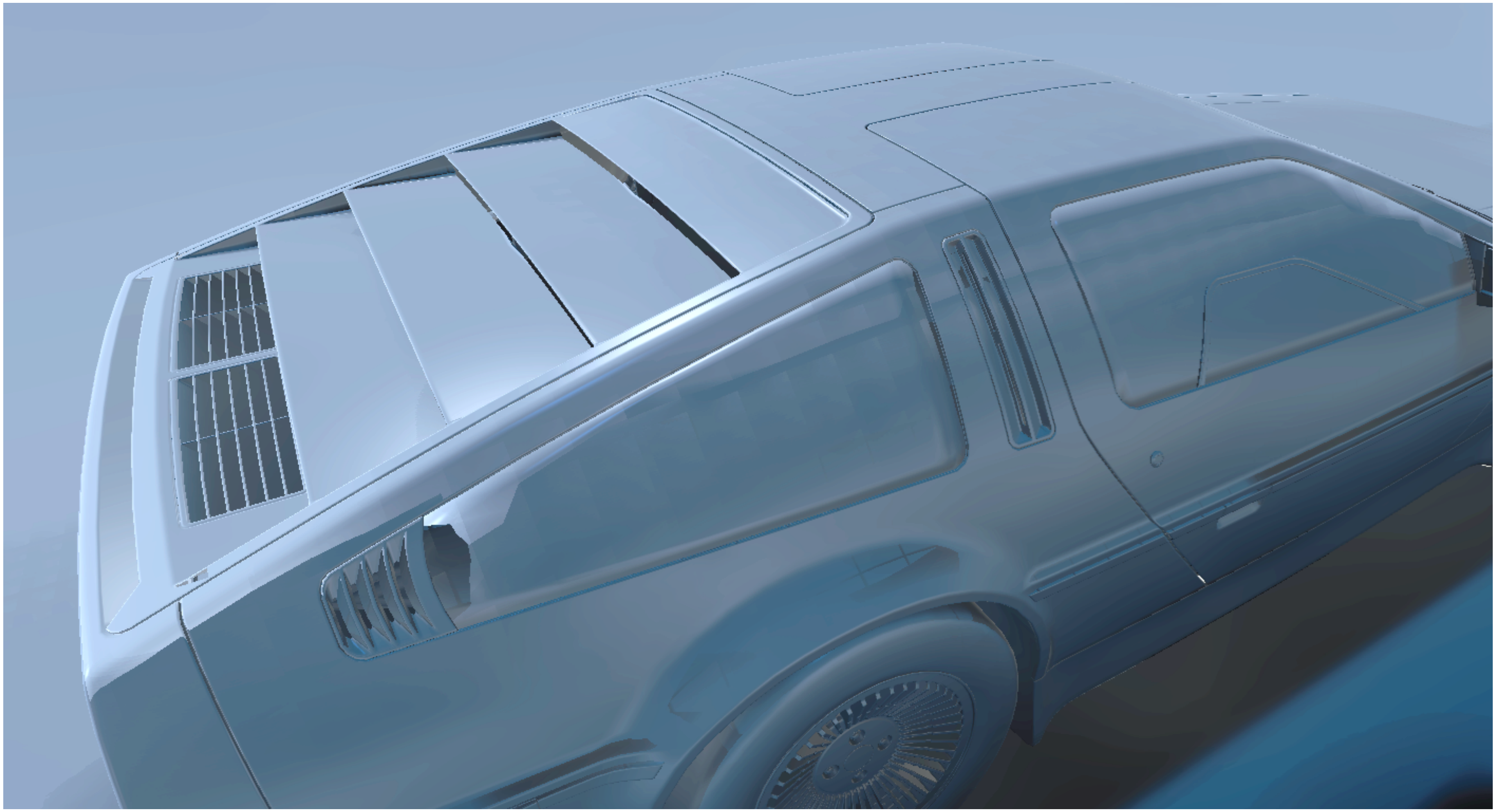
# Clustered Pre-Convolved Radiance Caching

**Hauke Rehfeld, Tobias Zirr, Carsten Dachsbacher**

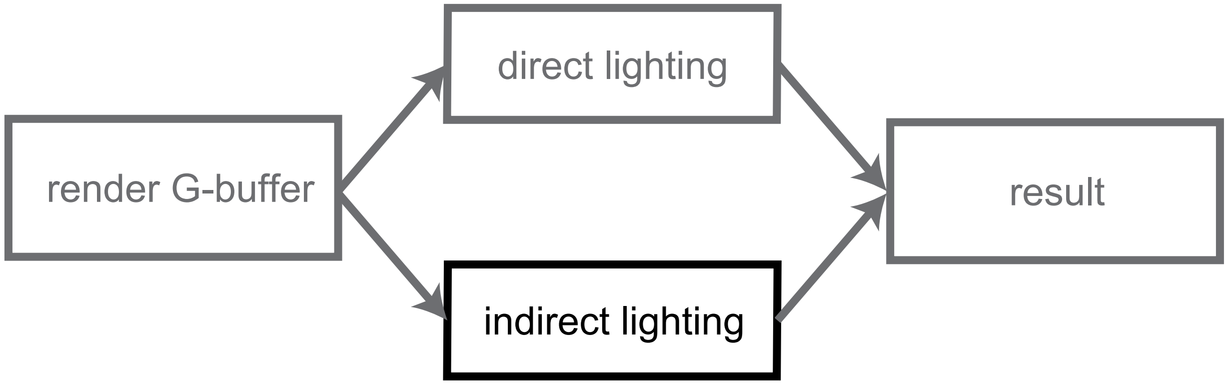
Karlsruhe Institute of Technology

# Problem

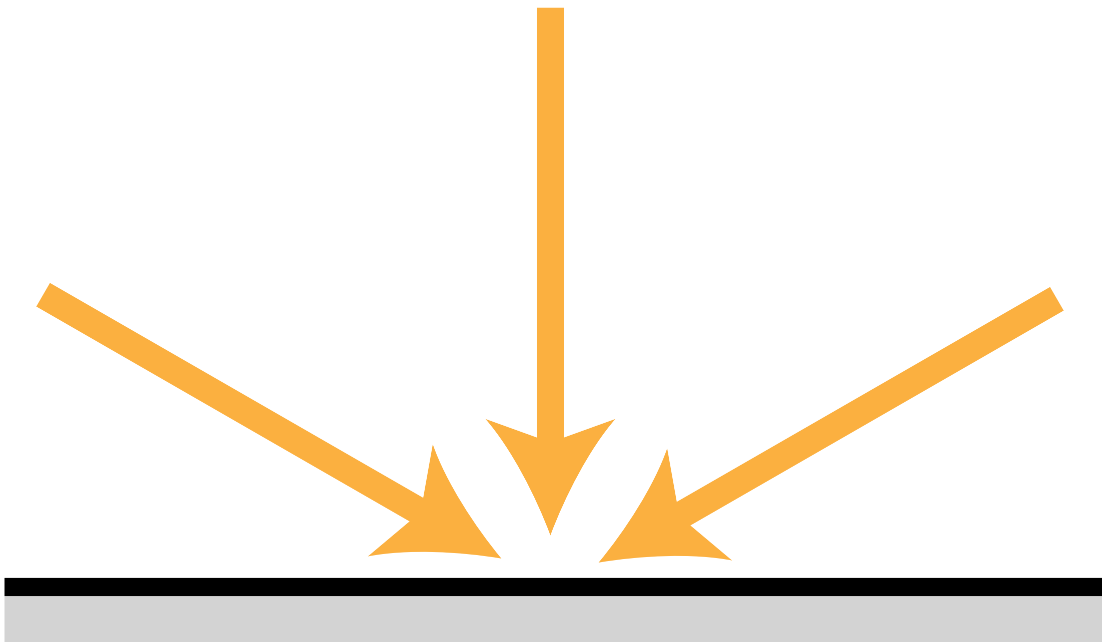


- indirect illumination
- interactive previews → single bounce GI!
- complex meshes and glossy materials

# Our Approach

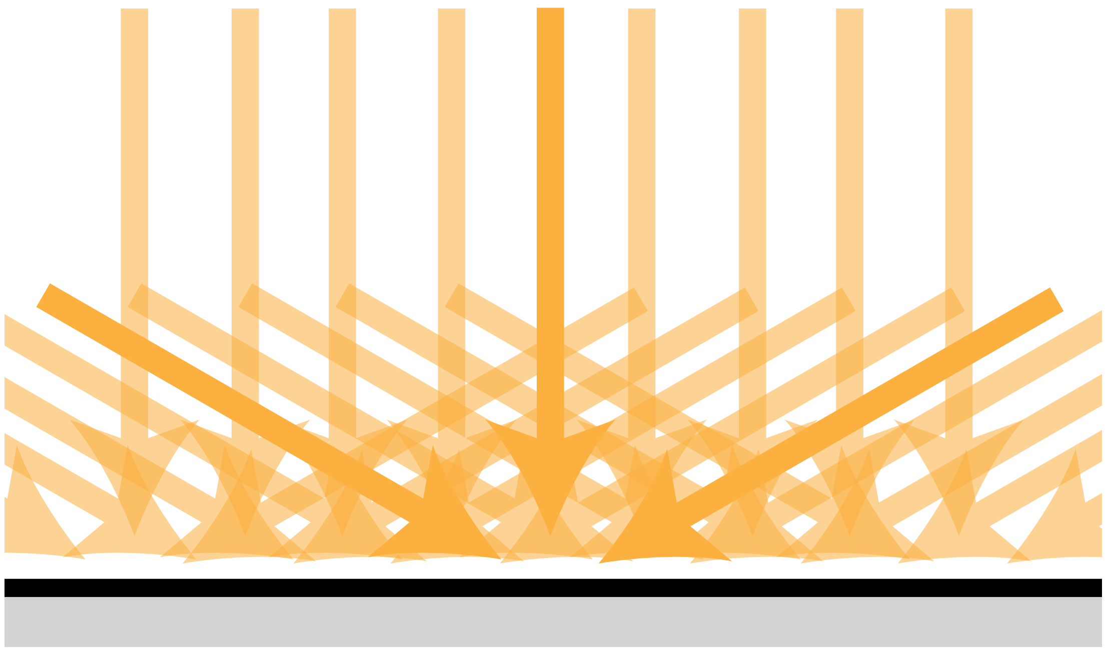


# Indirect Illumination



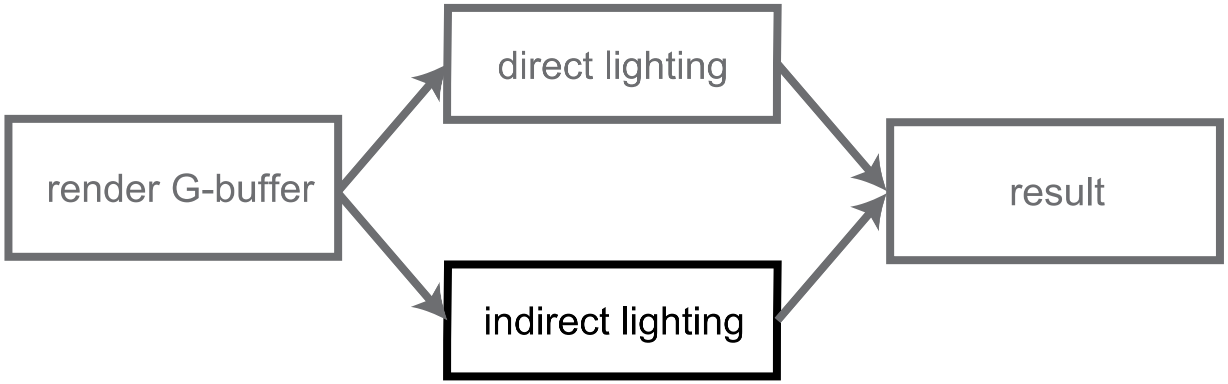
- Collect incident radiance

# Indirect Illumination



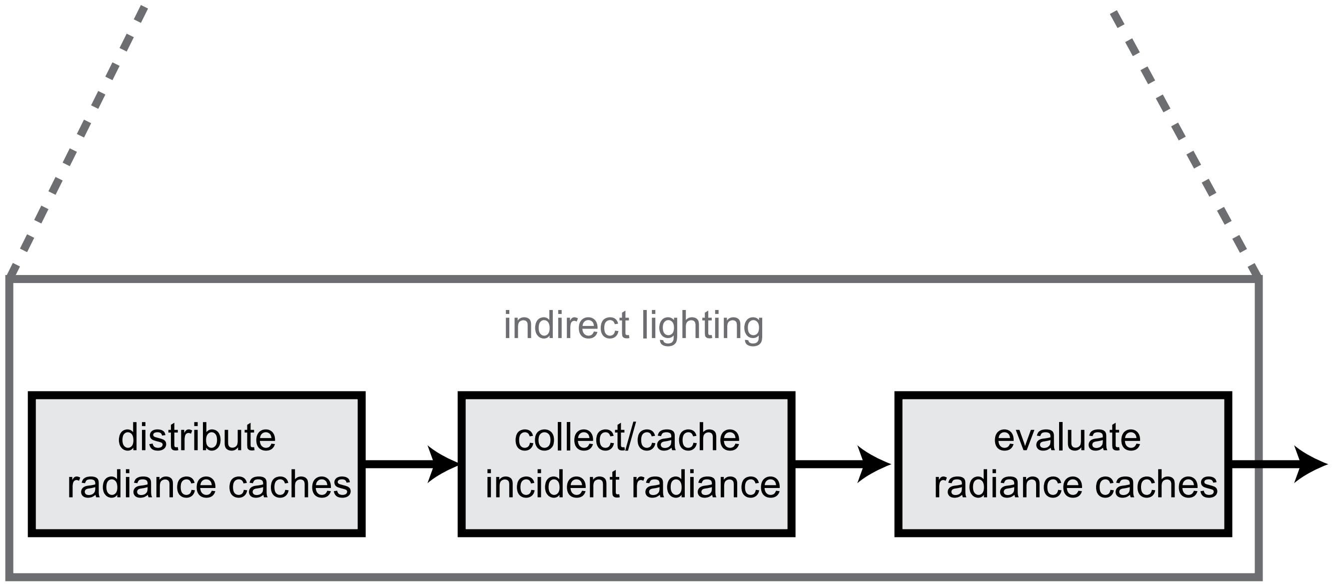
- Collect incident radiance
- For all surface points

# Our Approach




- Radiance Caching reduce number of indirect illumination samples
- Pre-convolved Radiance Caching for interactive use
  - pre-integrates glossy and diffuse
  - gpu-friendly
- deferred shading (G-buffer pass + shading passes)

# Contributions




- radiance cache distribution
  - based upon Clustered Deferred Shading
- employ Voxel Cone Marching to collect incident radiance
- enable a gathering approach to evaluate radiance caches

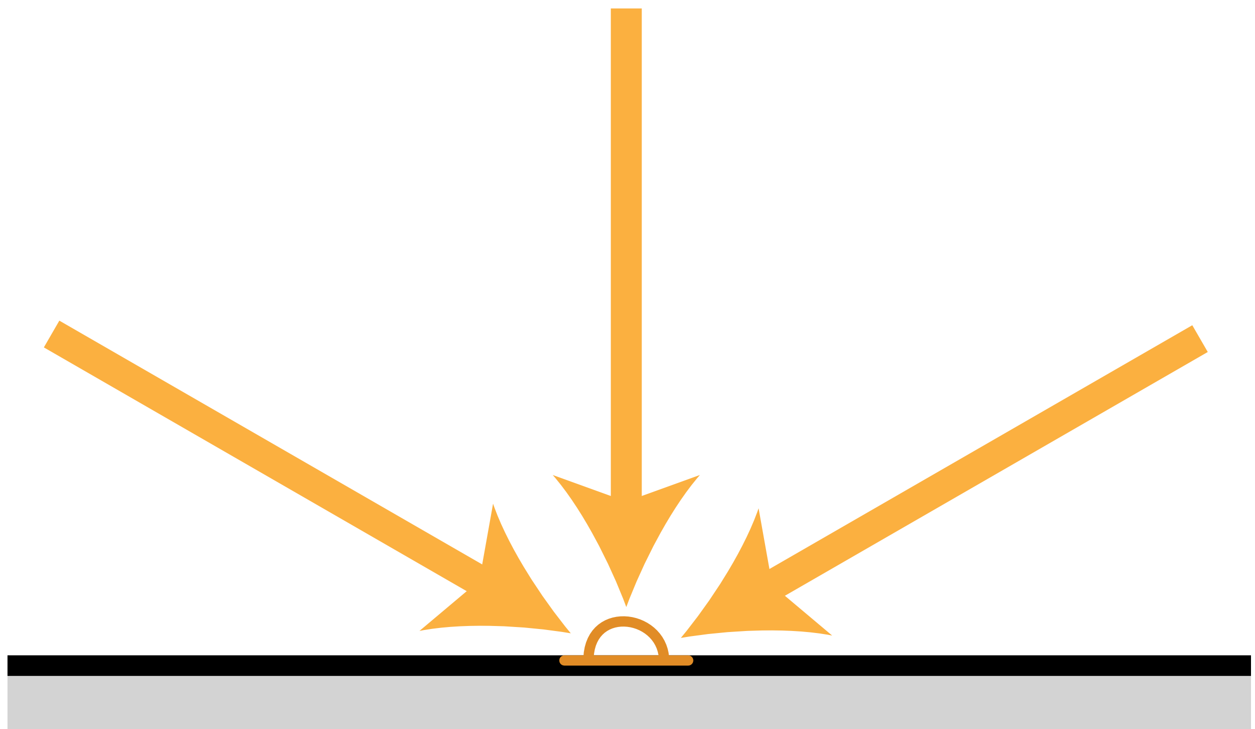
# Previous Work

- many approaches
  - screen-space
  - voxel-based
  - interactive raytracing
  - many-light methods
- here: only building blocks

# Radiance Caching [Křivánek et al. 05]

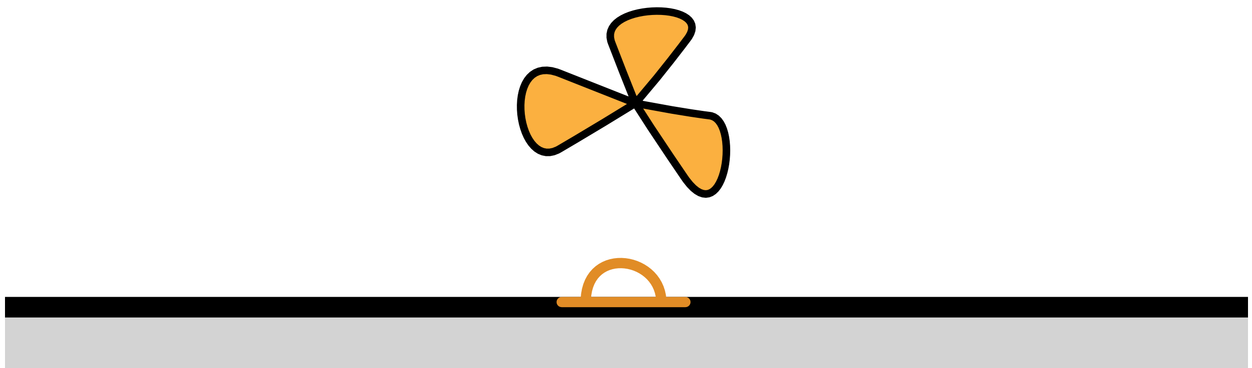
- place radiance cache

# Radiance Caching



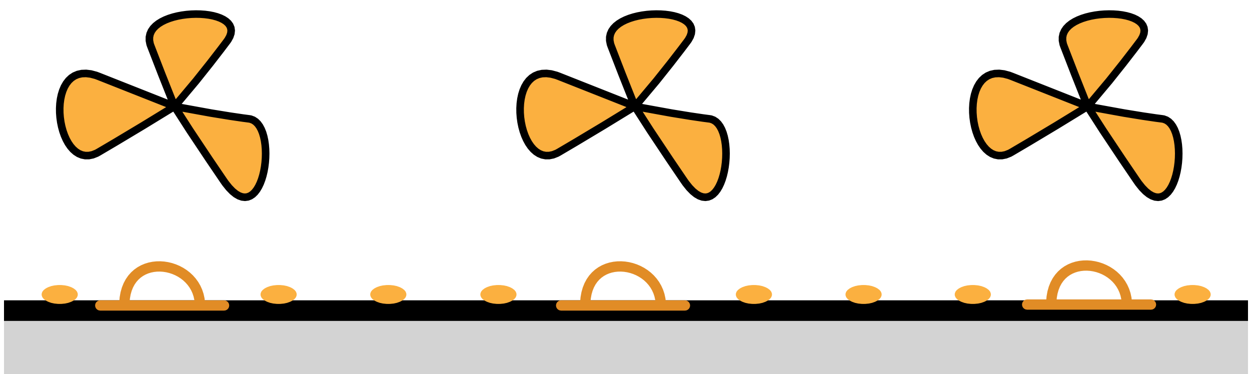
- place radiance cache
- collect incident radiance for the radiance cache

# Radiance Caching



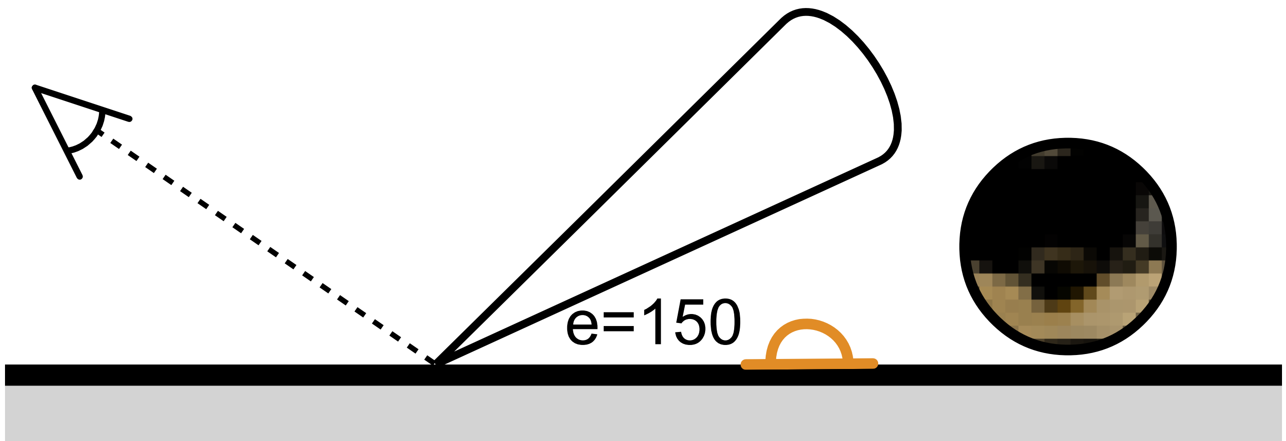
- place radiance cache
- collect incident radiance for the radiance cache
- store as spherical harmonic coefficients

# Radiance Caching



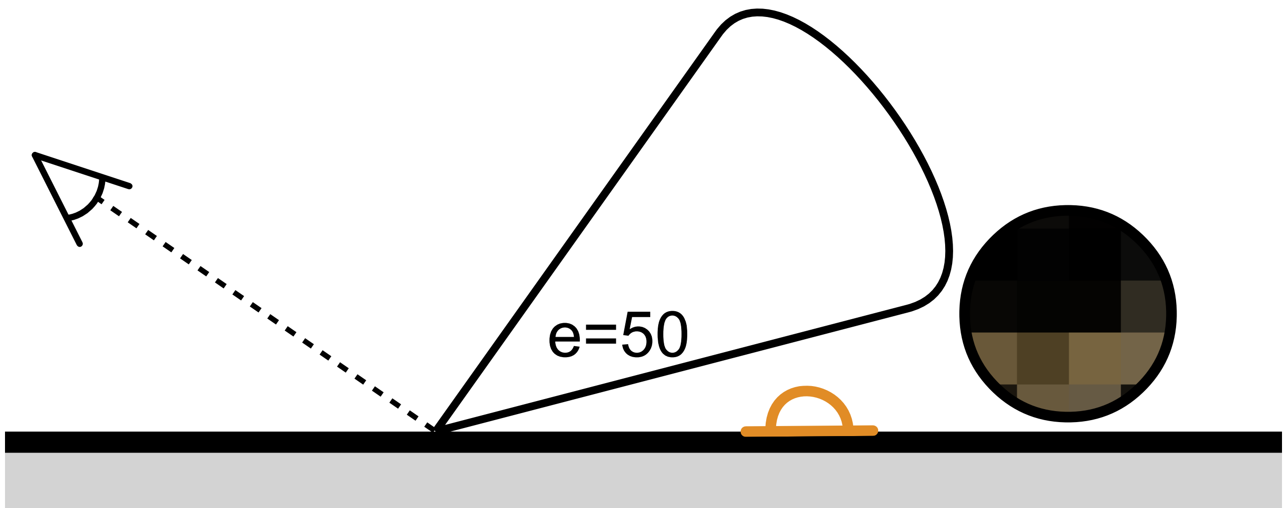
- place radiance cache
- collect incident radiance for the radiance cache
- store as spherical harmonic coefficients
- evaluate for surface points

# Pre-Convolved Radiance Caching



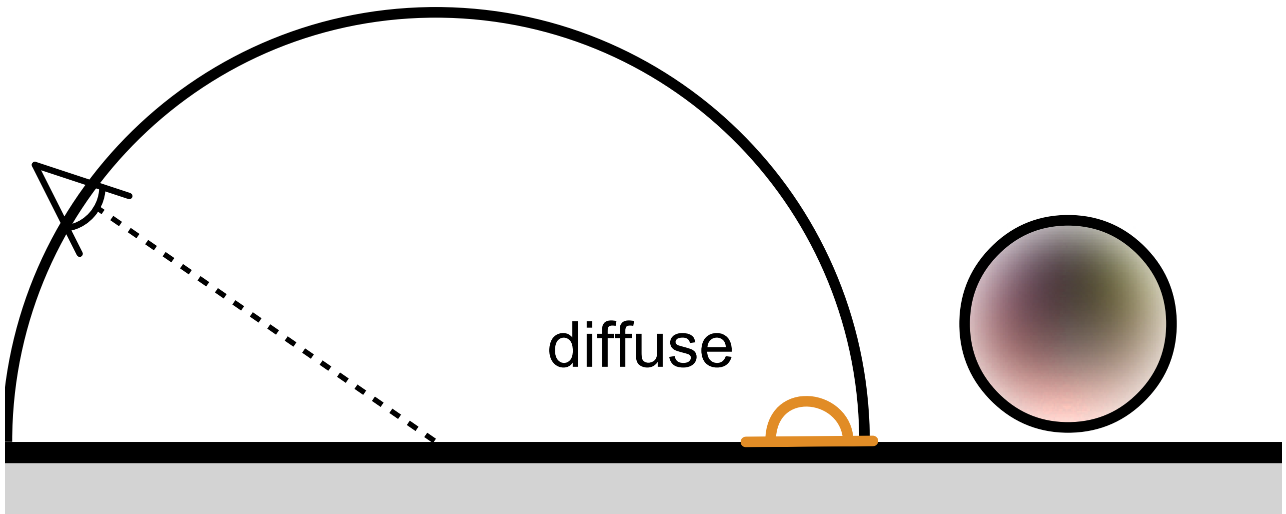

- phong-model: glossy lobe + diffuse
- store incident radiance in pre-convolved envmap [Scherzer et al. 12]

# Pre-Convolved Radiance Caching



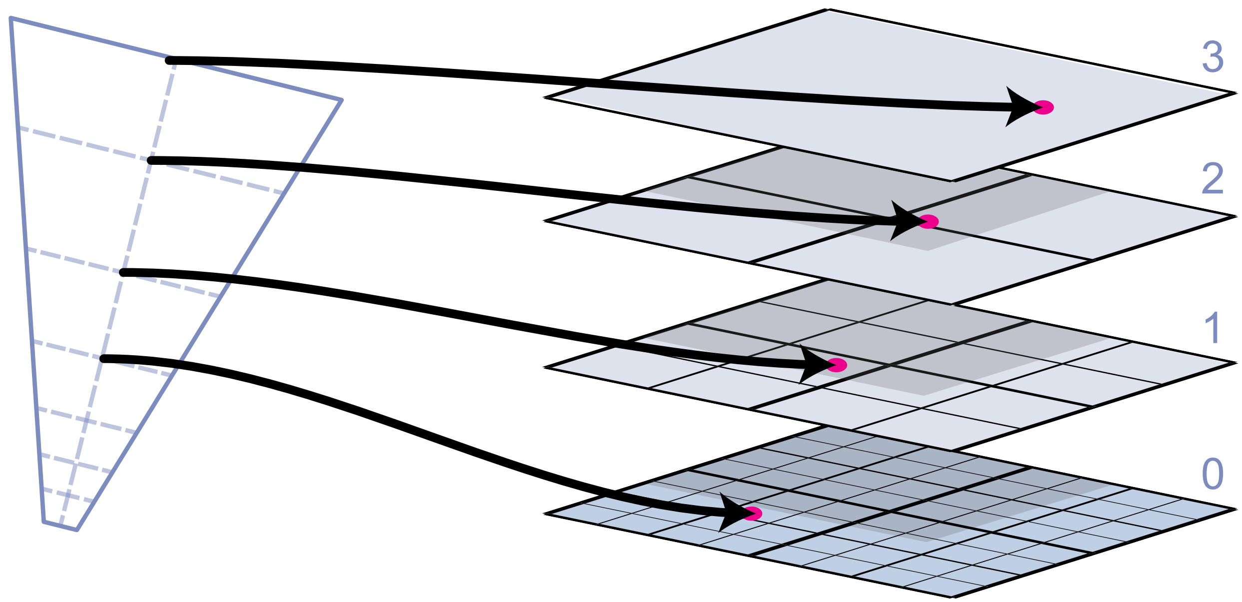

- phong-model: glossy lobe + diffuse
- store incident radiance in pre-convolved envmap

# Pre-Convolved Radiance Caching




- phong-model: glossy lobe + diffuse
- store incident radiance in pre-convolved envmap

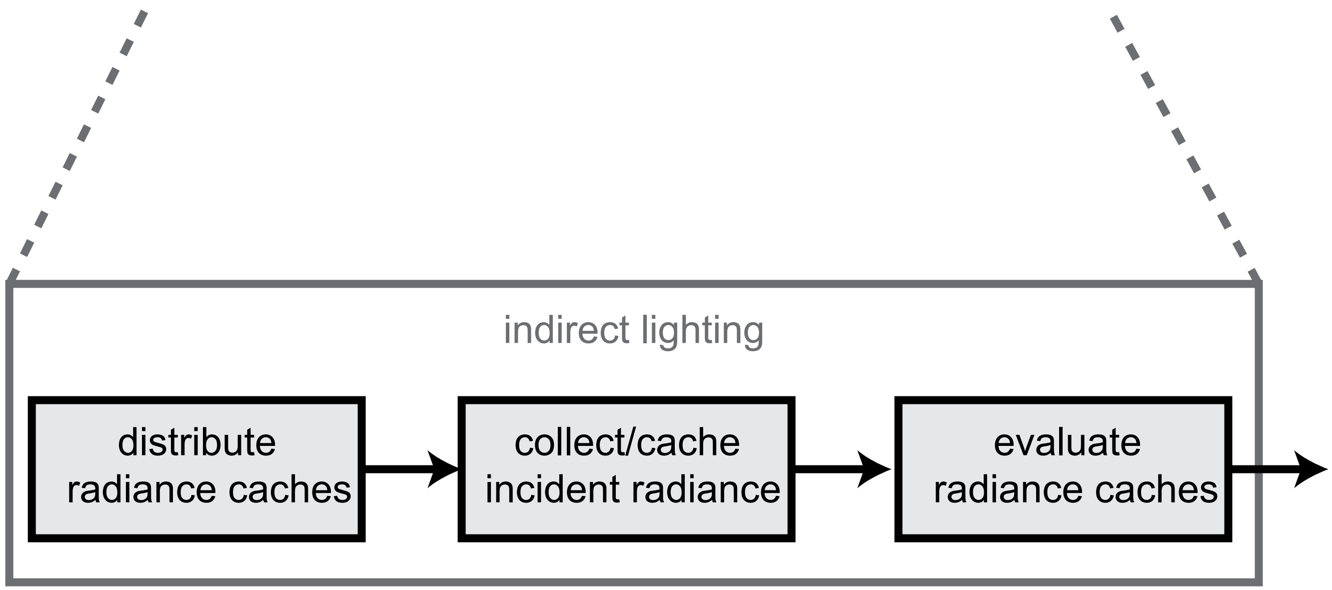
# Voxel Cone Marching




[Crassin et al. 11]

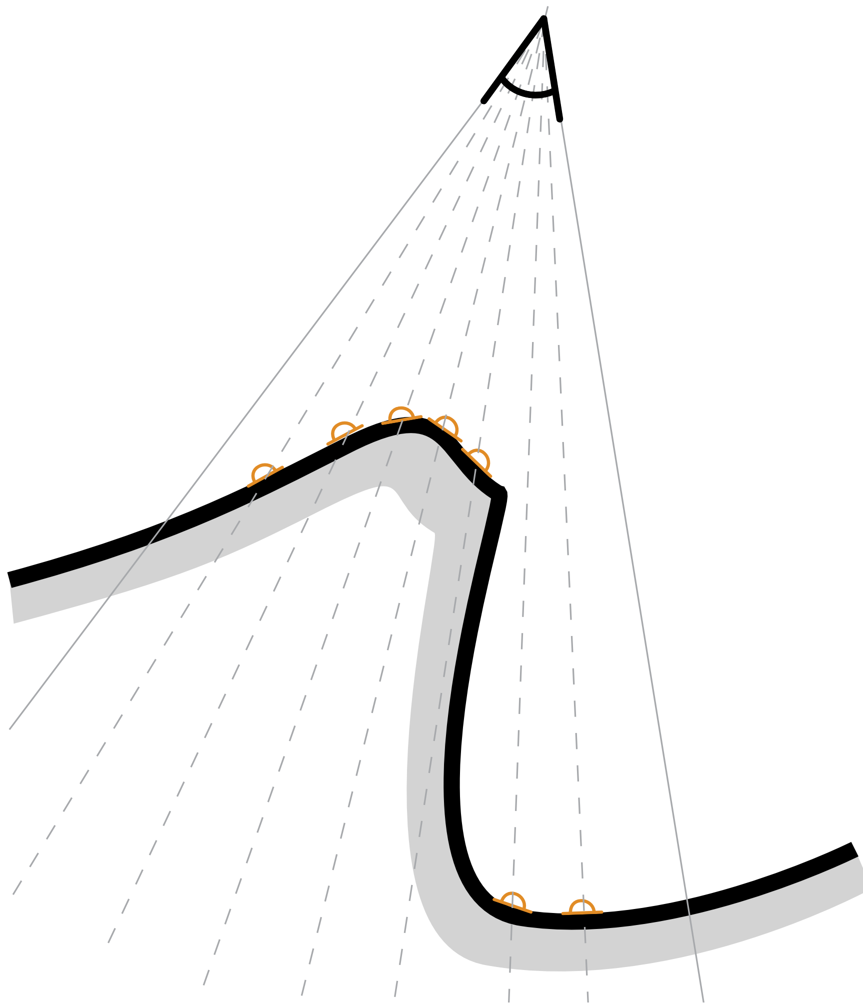
- cone-march pre-filtered hierarchy
- larger cone → lower resolution layer
- very fast retrieval

# Clustered Pre-Convolved Radiance Caching (CPCRC)



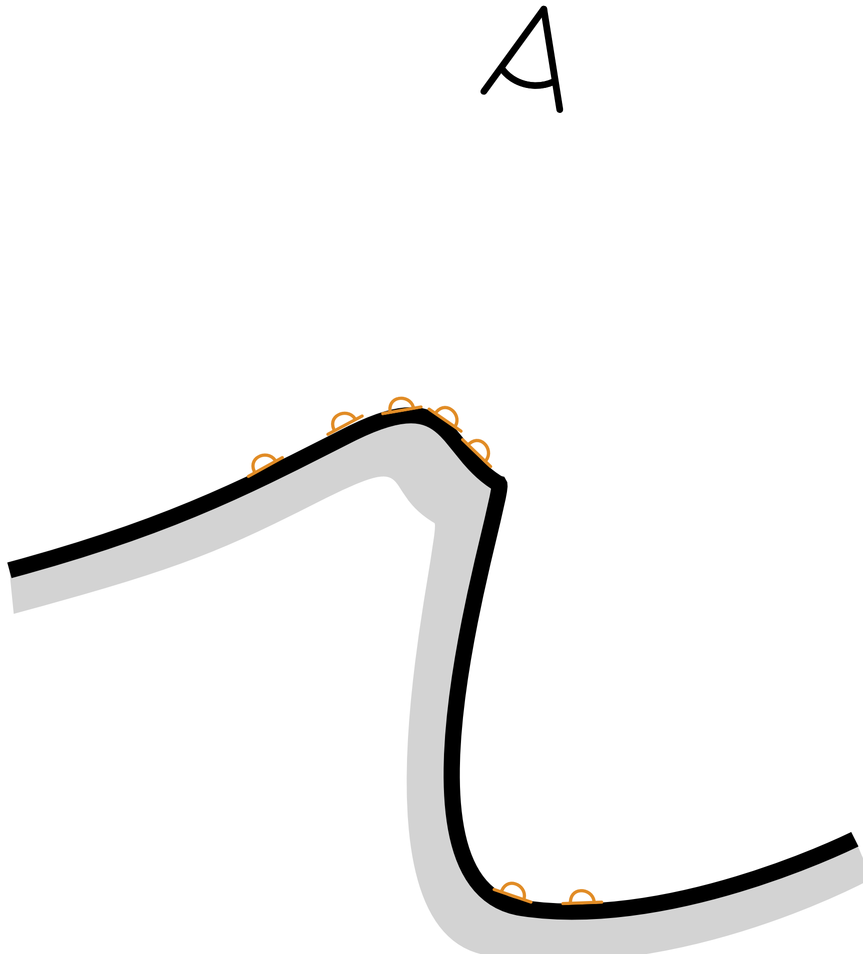

- distribute radiance caches

# Radiance Cache Distribution



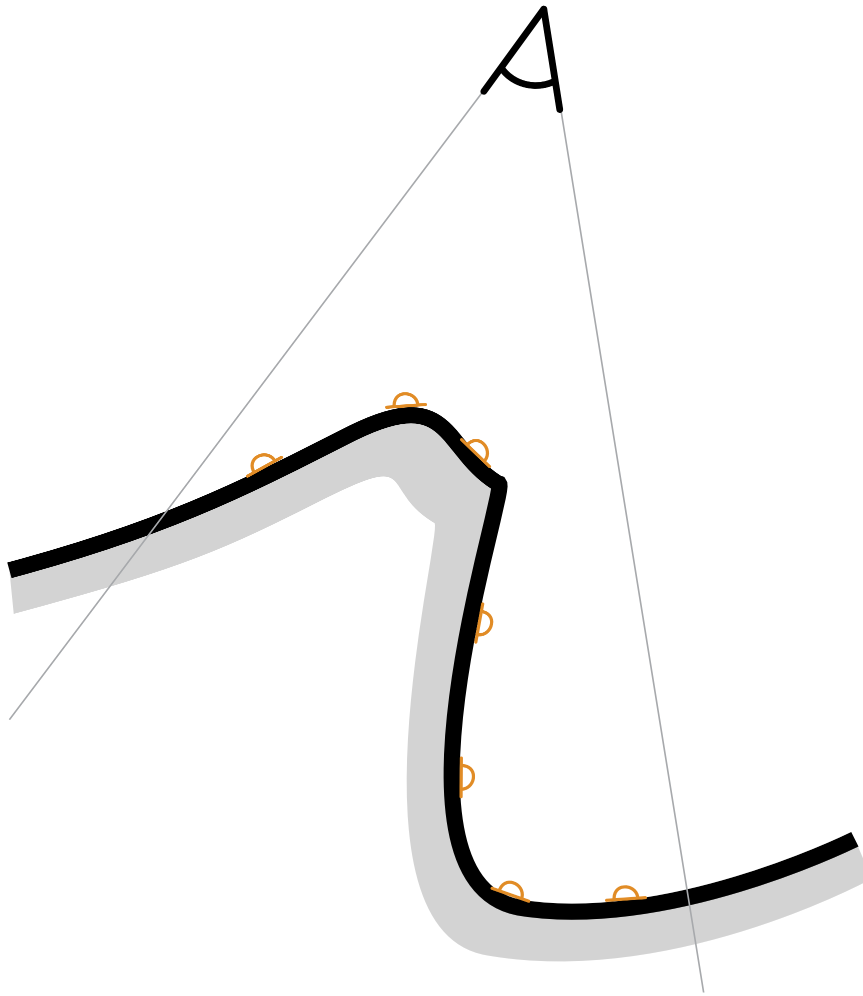
equi-distant distribution

# Radiance Cache Distribution



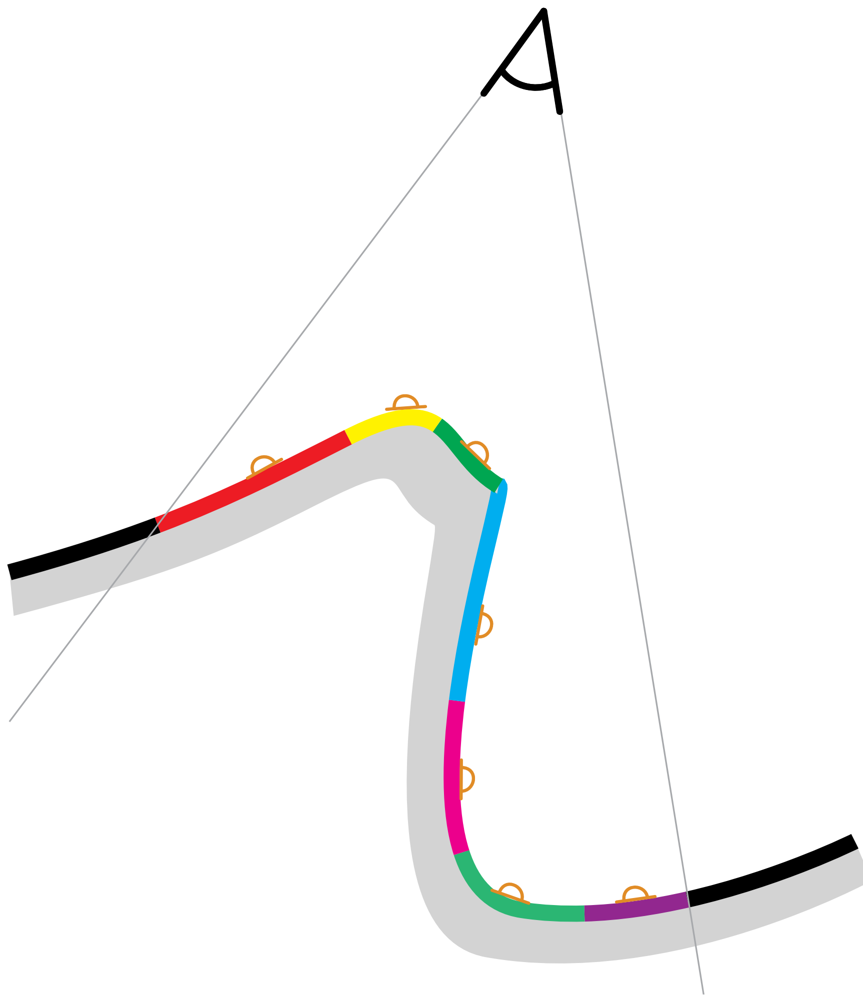

equi-distant distribution

# Radiance Cache Distribution



better distribution

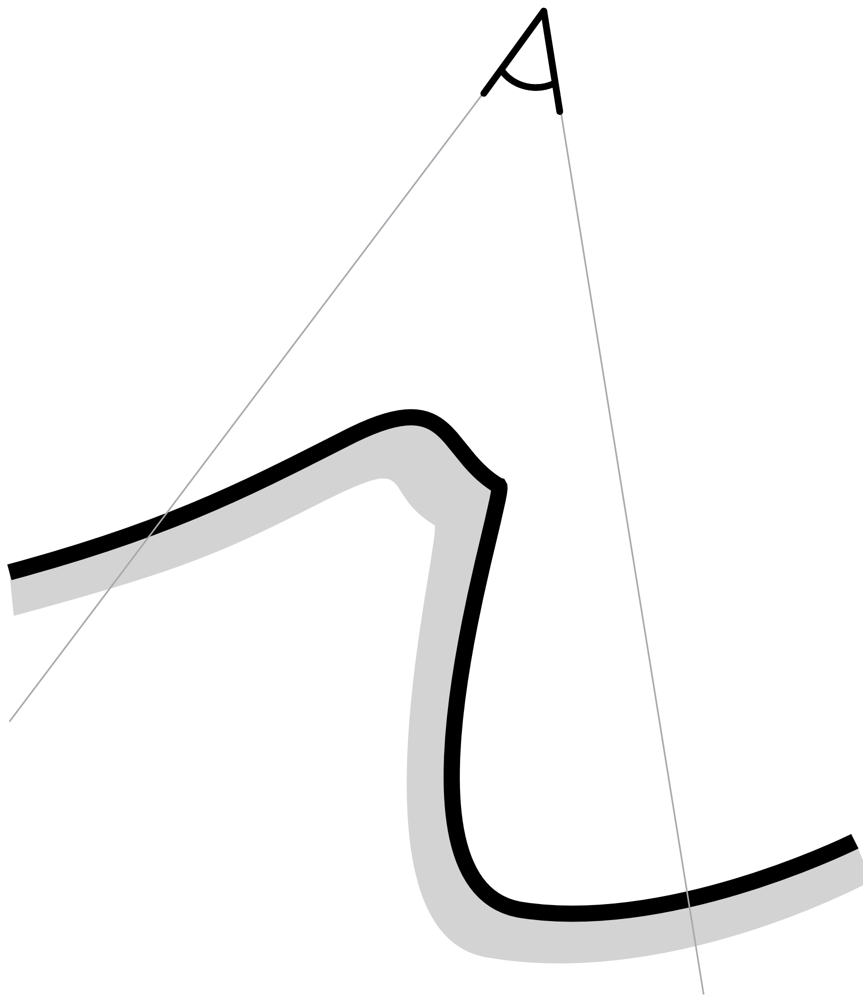
# Radiance Cache Distribution



better distribution

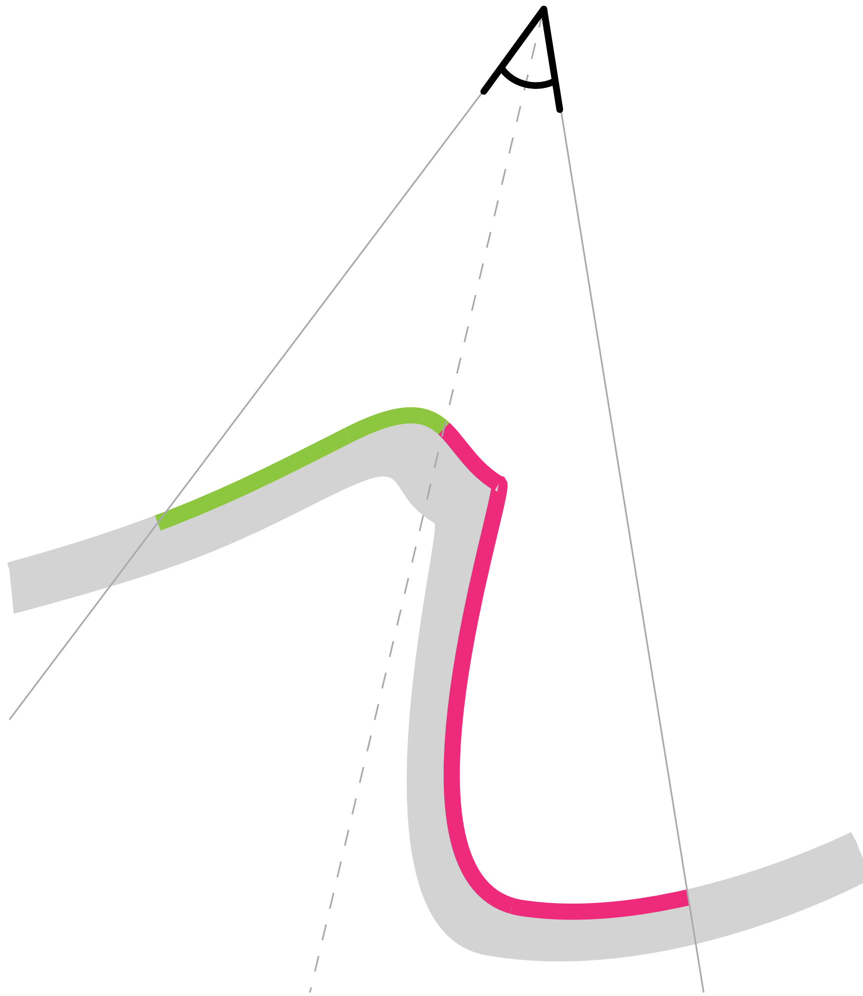
- cluster G-buffer

# Clustered Deferred Shading



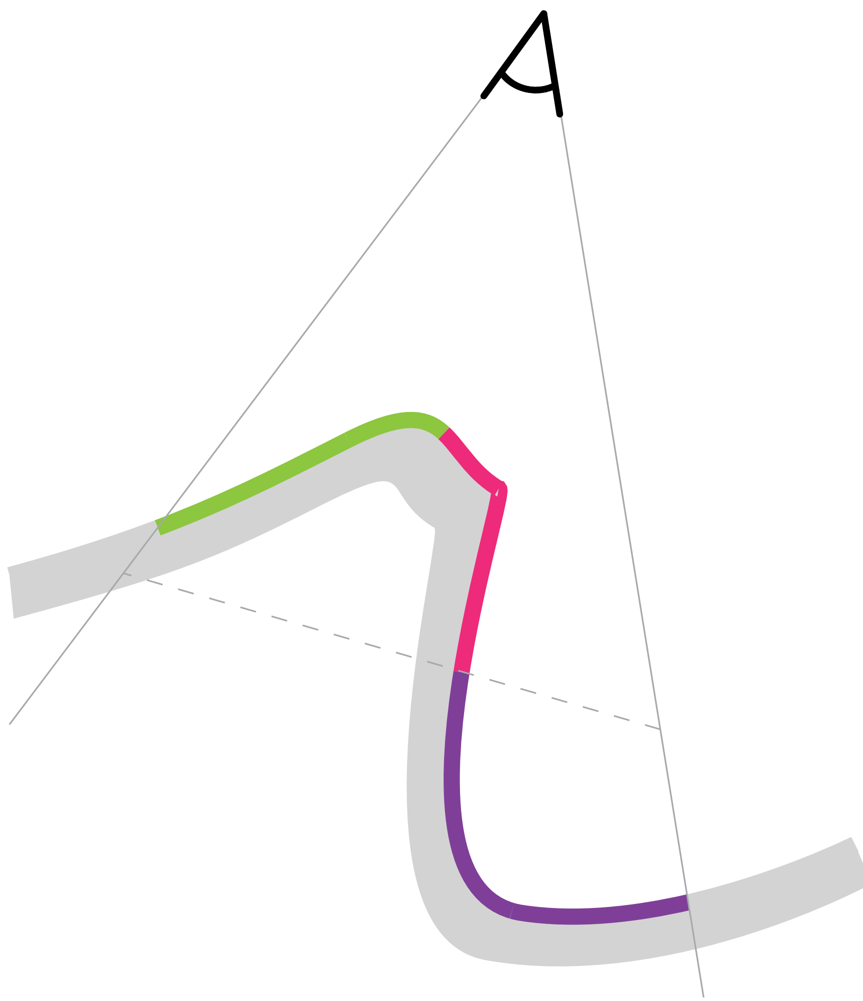
[Olsson et al. 12]:

# Clustered Deferred Shading



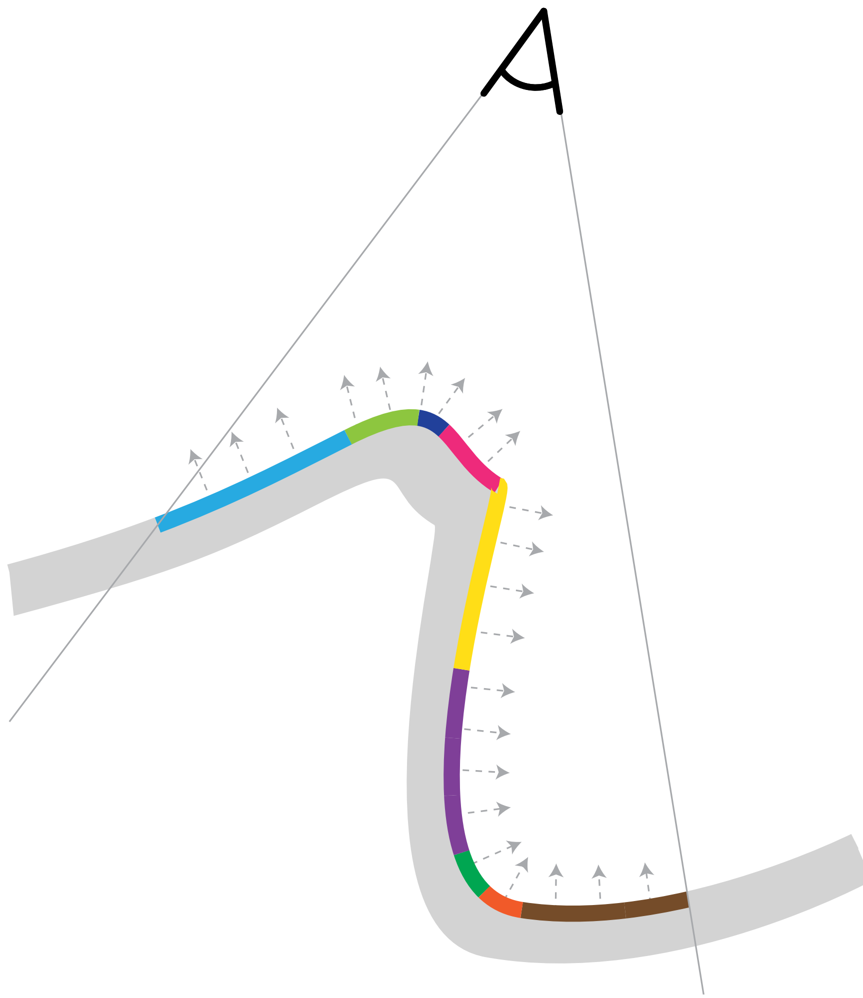
cluster by: tiles

# Clustered Deferred Shading



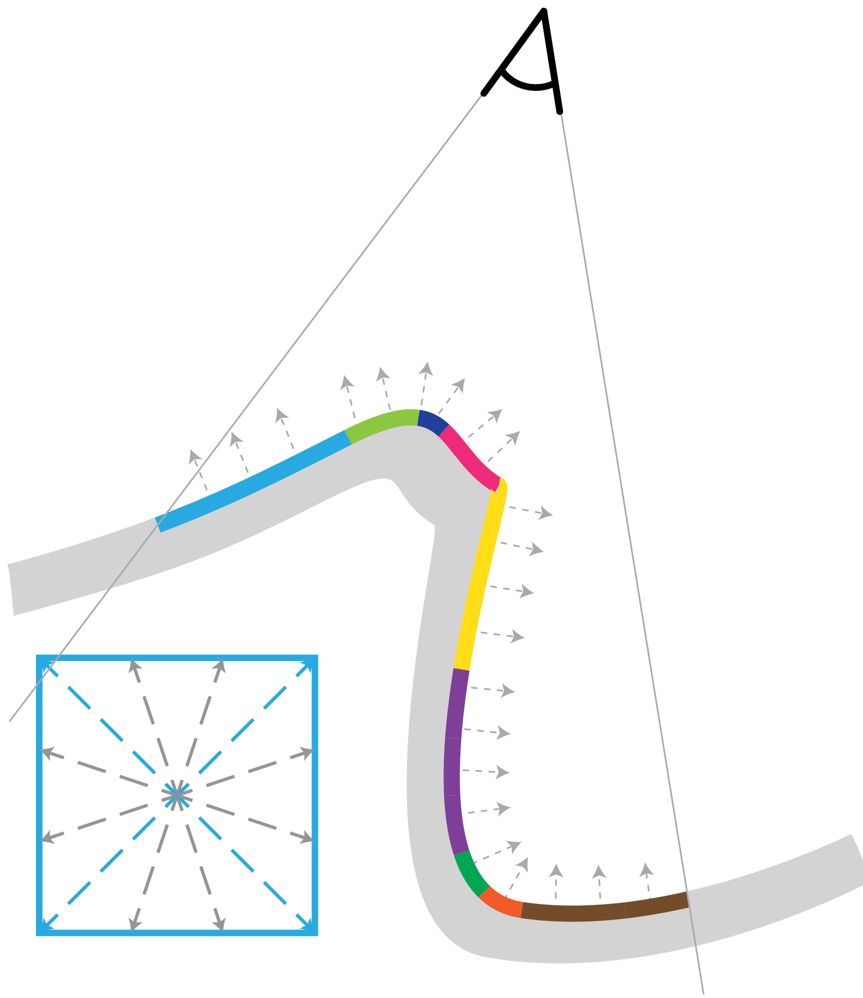
cluster by: tiles, depth

# Clustered Deferred Shading



cluster by: tiles, depth, normals

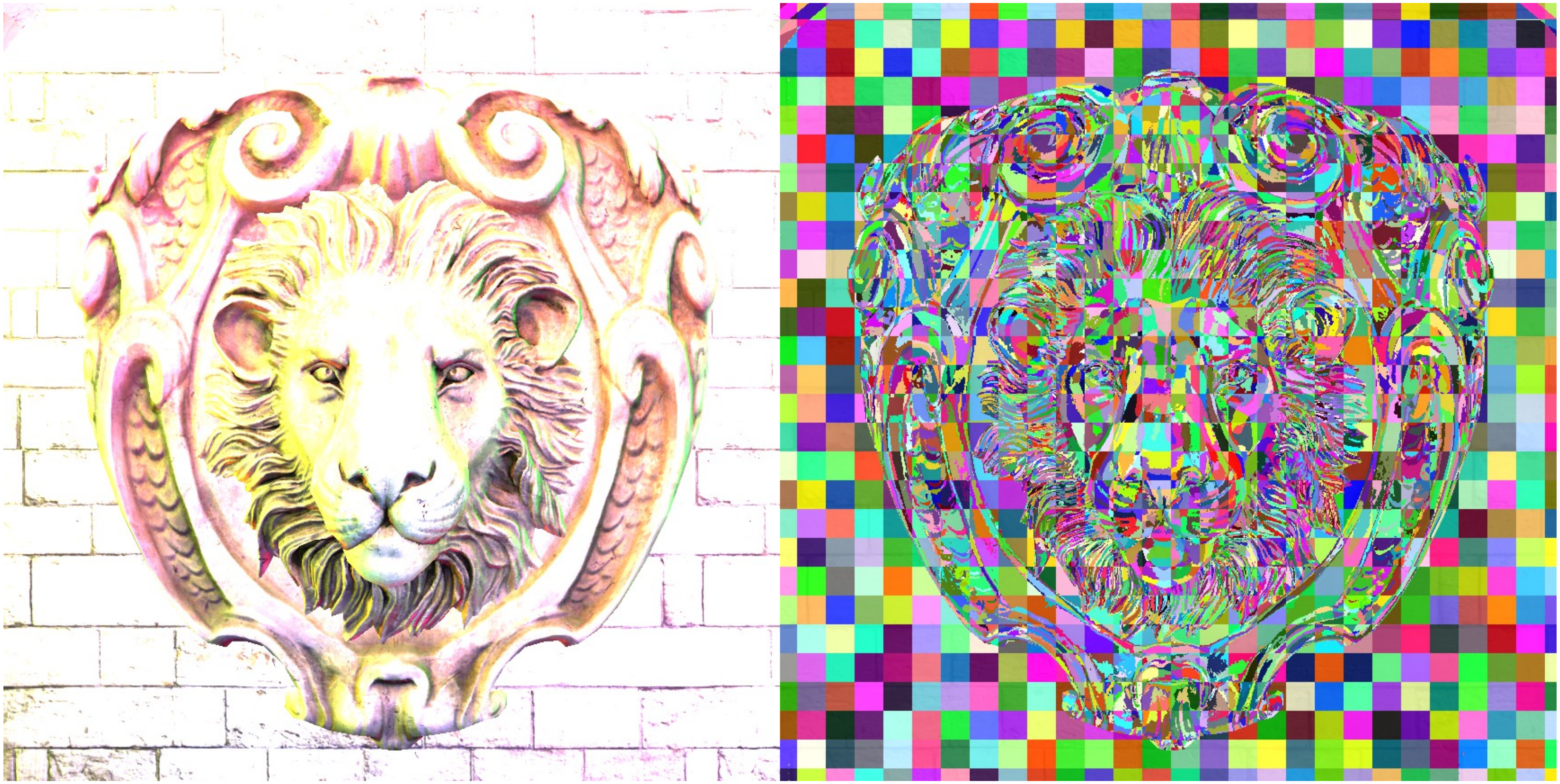
# Clustered Deferred Shading



cluster by: tiles, depth, normals

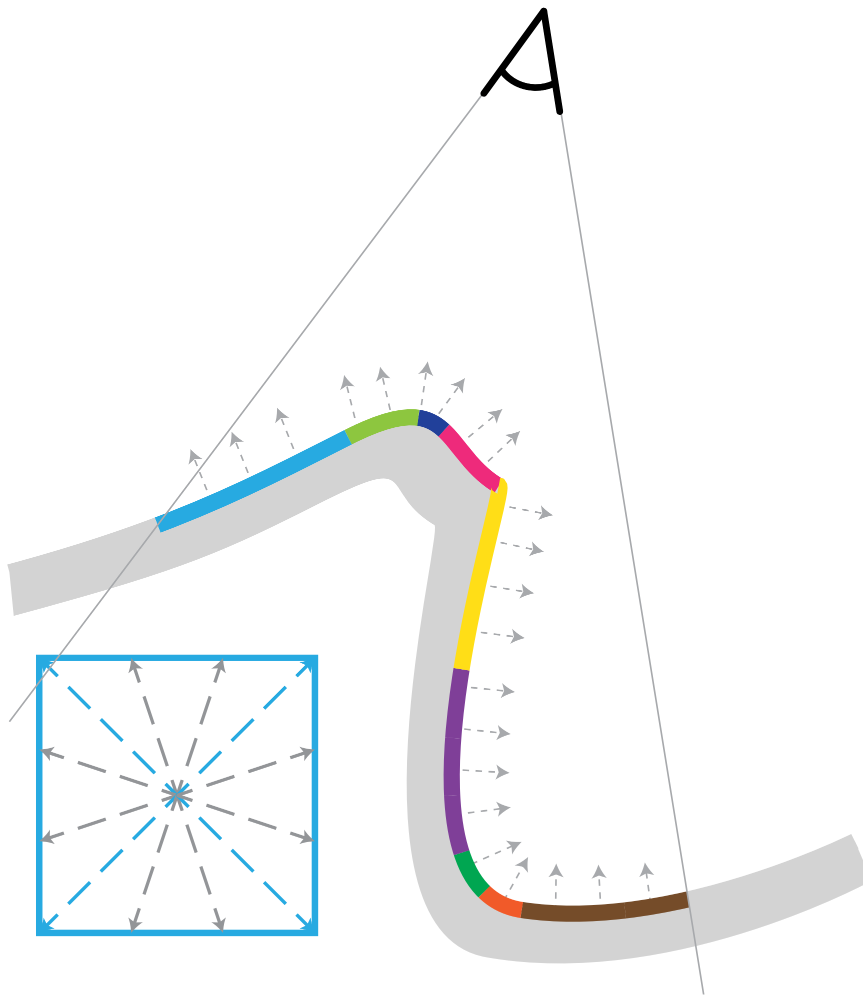
- global clustering of directions

# Clustered Deferred Shading



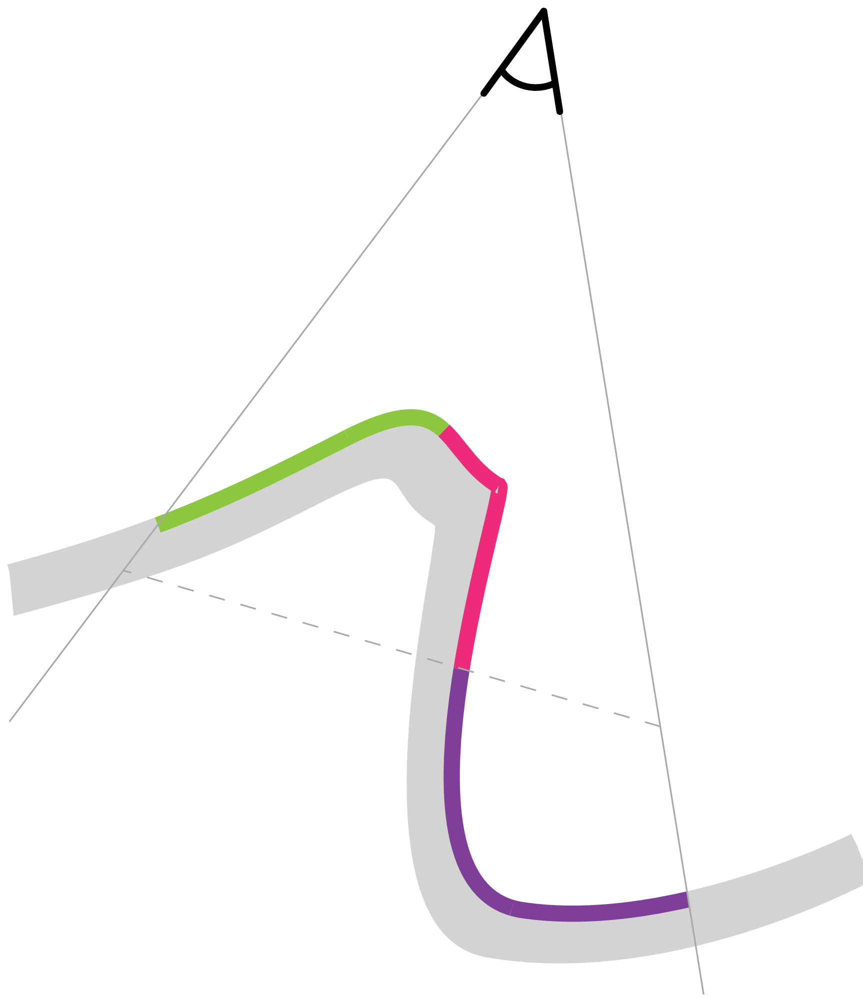
Example

# Radiance Cache Distribution



cluster by: tiles, depth, normals

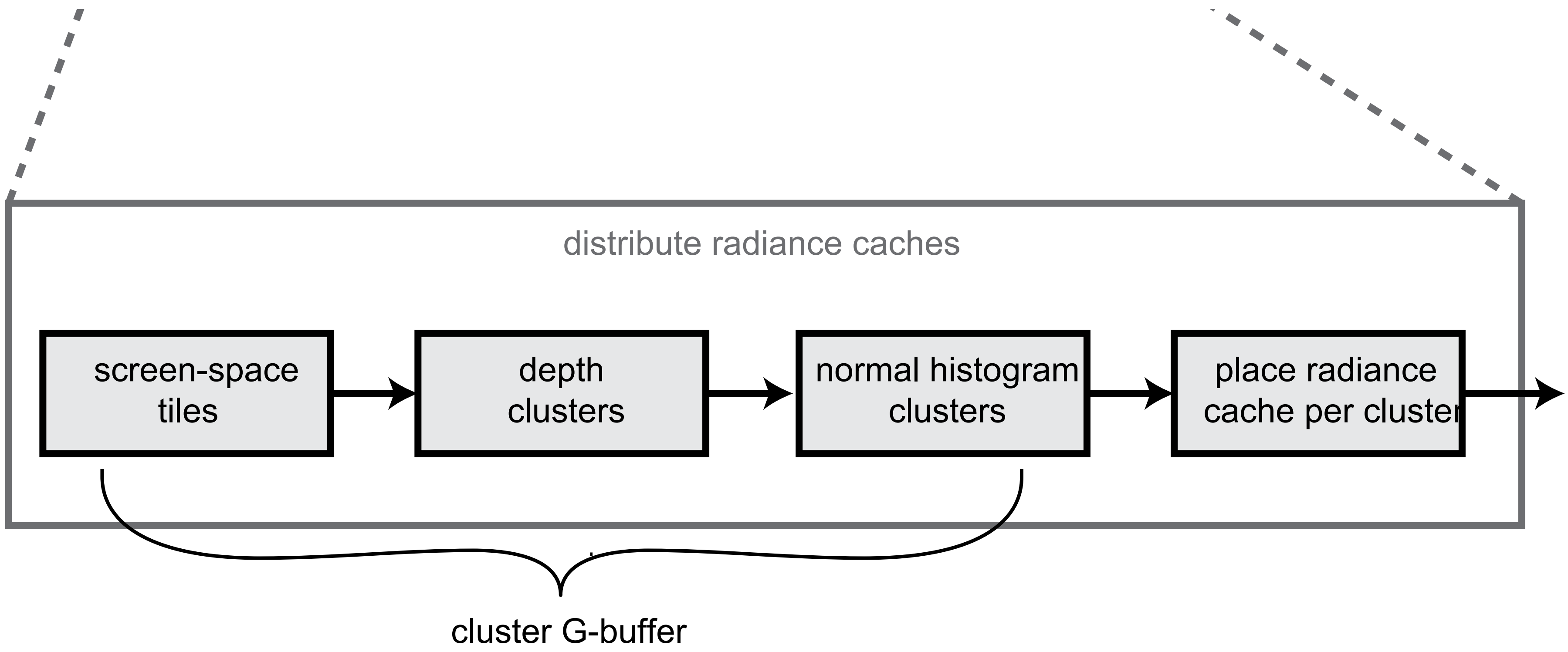
# Radiance Cache Distribution



cluster by: tiles, depth, ~~normals~~

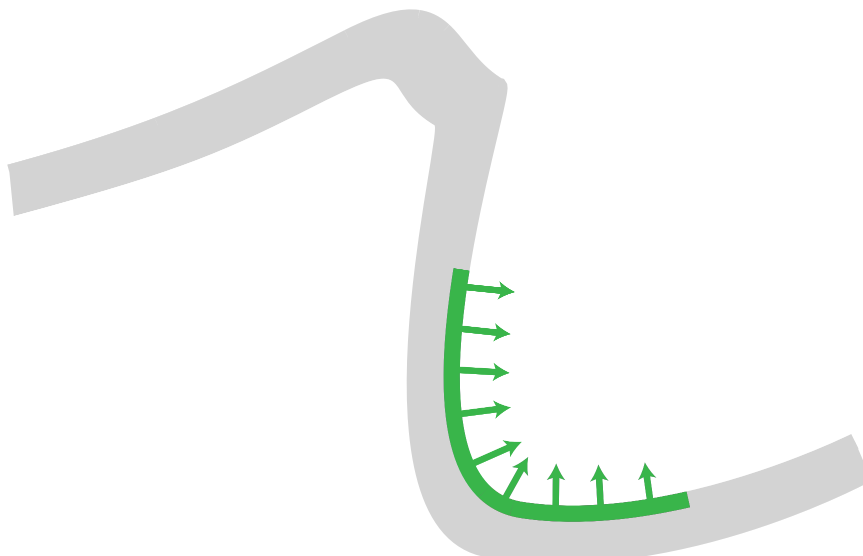
- reuse clustered deferred shading algorithm
- replace normal clustering

# Radiance Cache Distribution



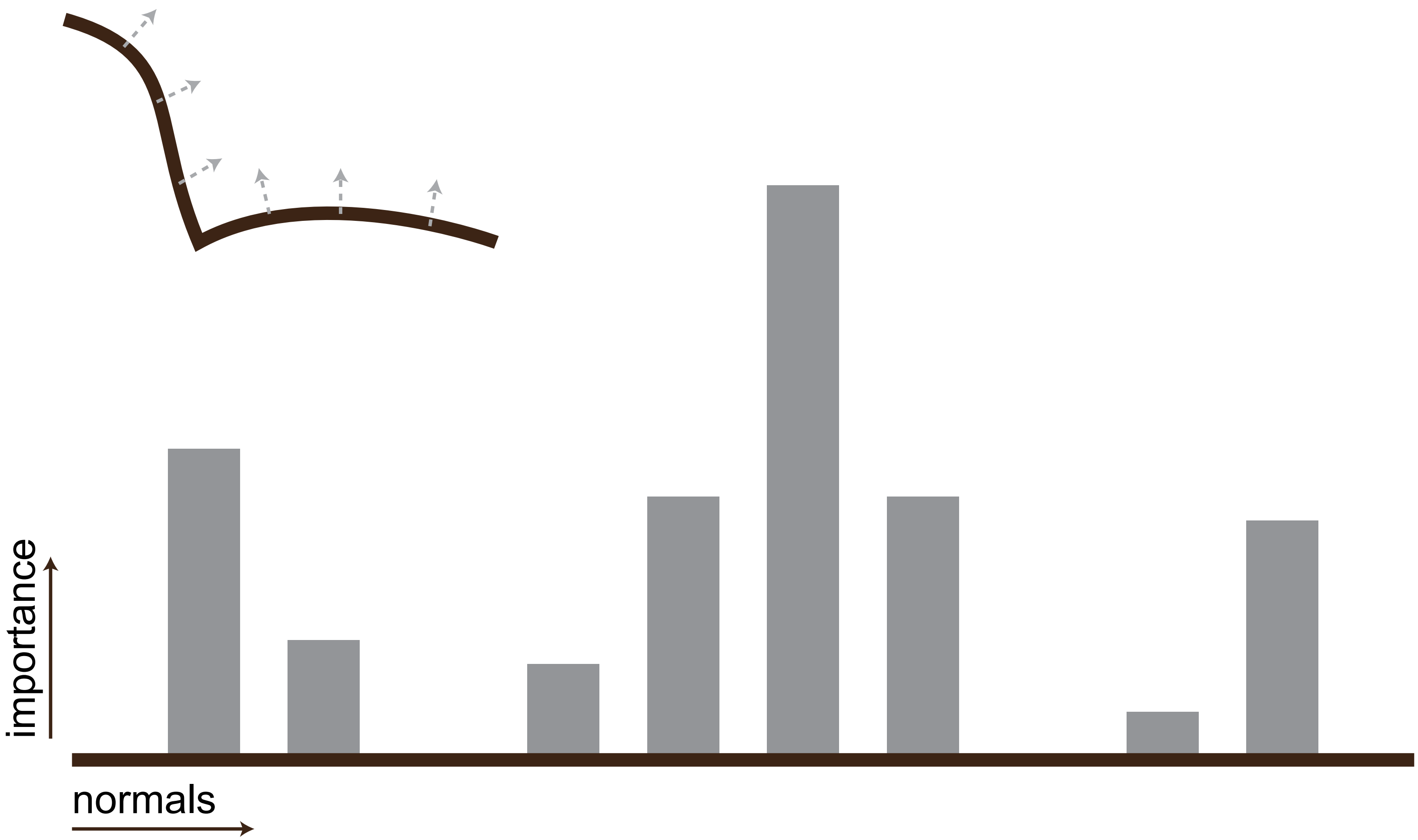
# Normals - 2D Histogram Clustering

A

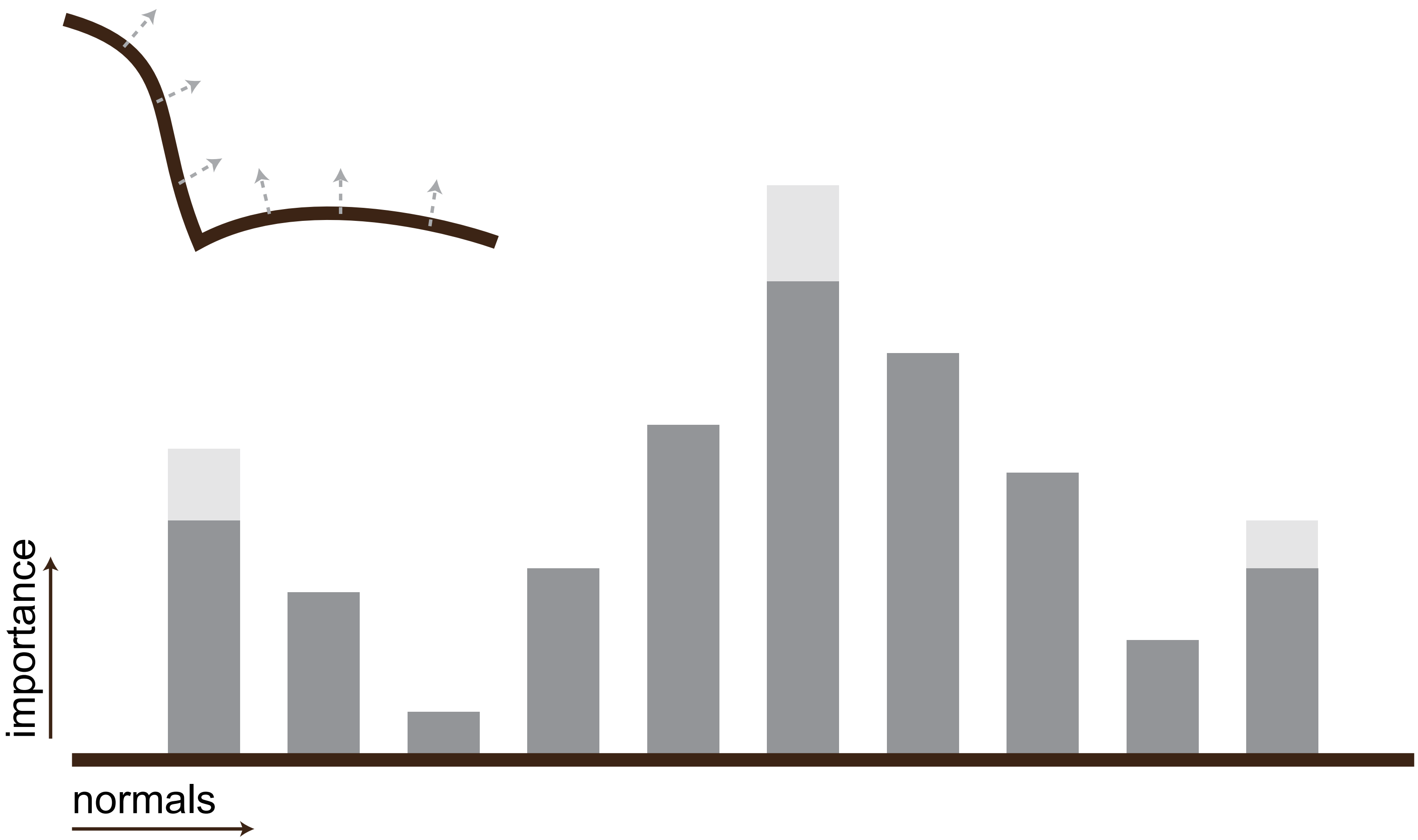


# Normals - 2D Histogram Clustering

# Normals - 2D Histogram Clustering

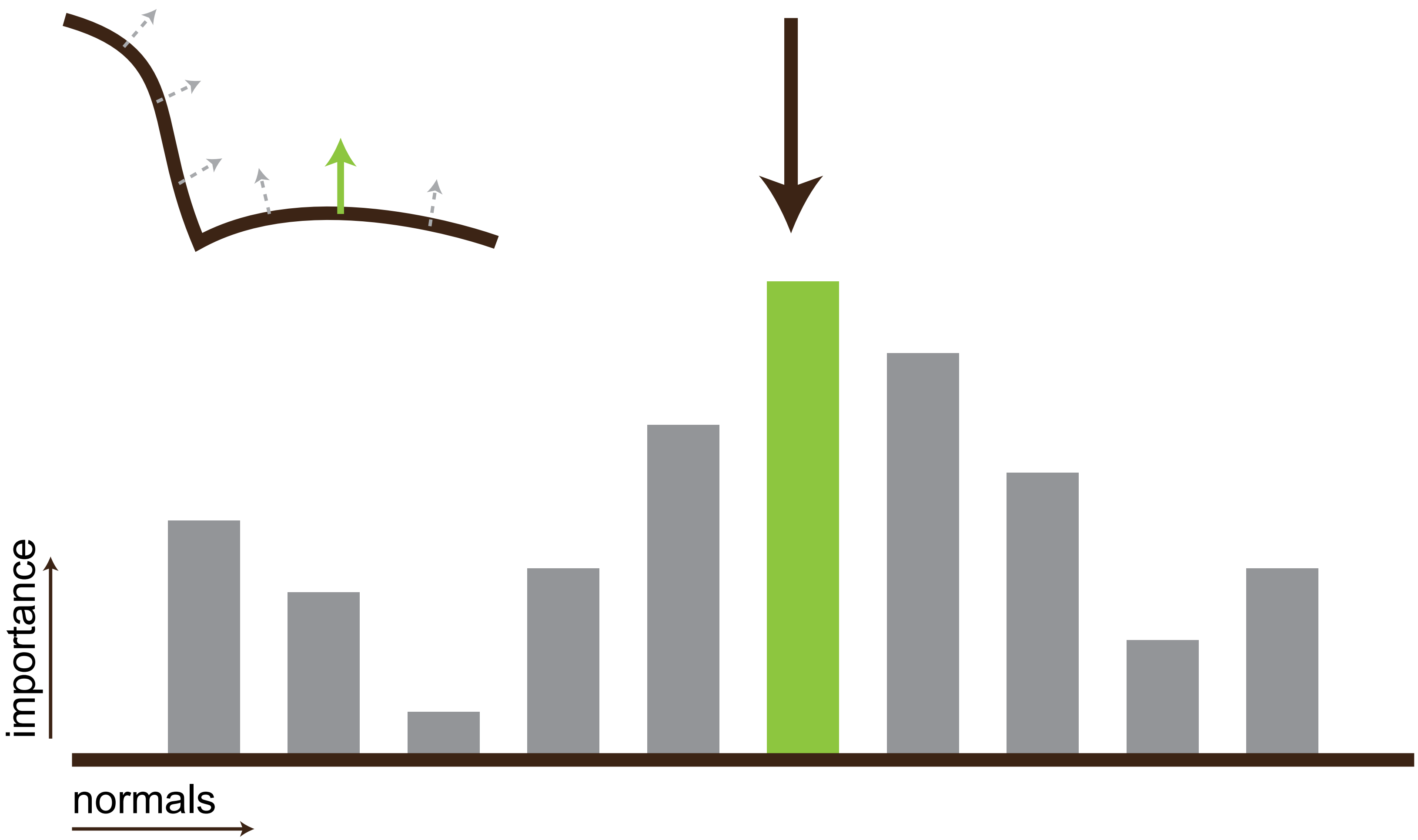


# Normals - 2D Histogram Clustering



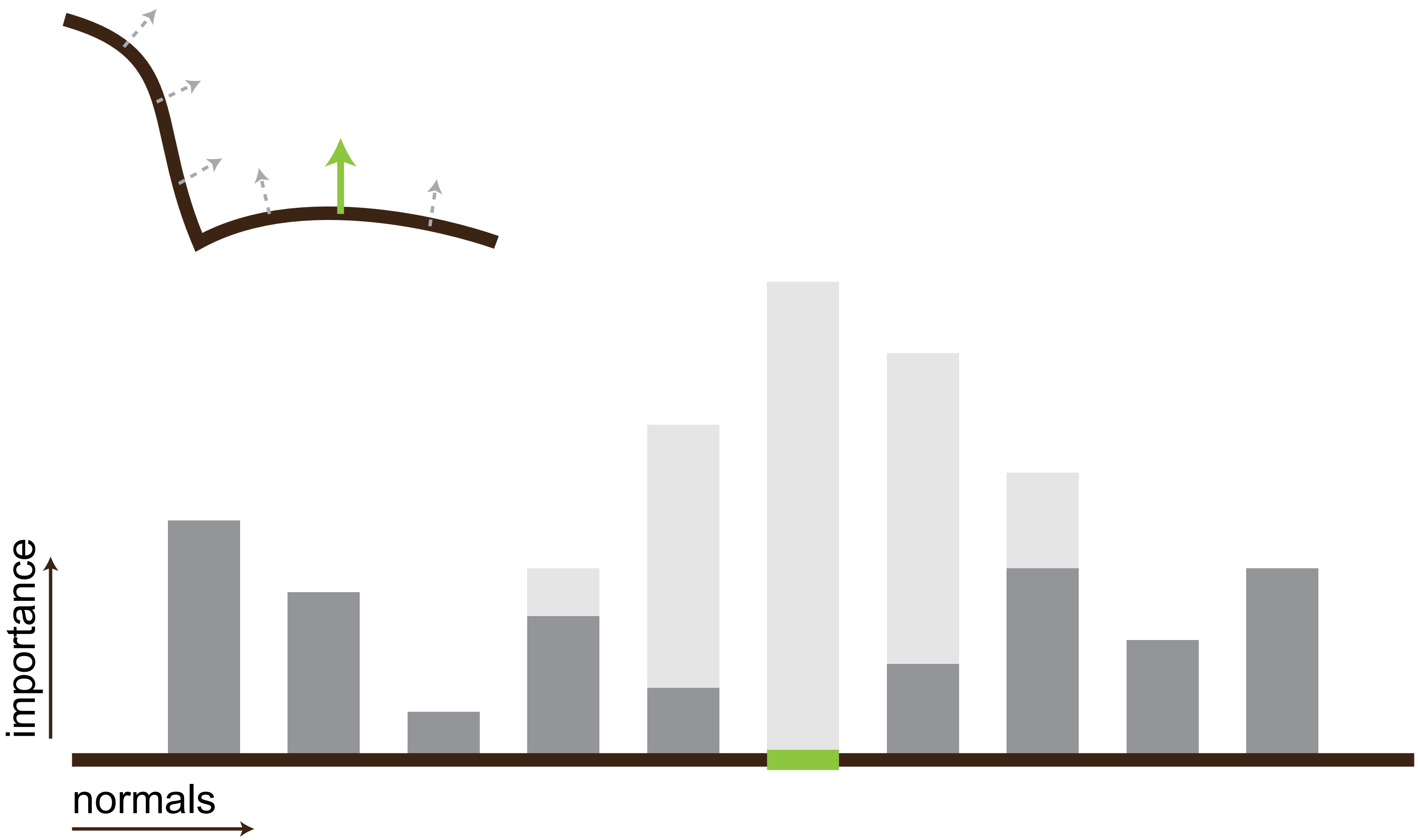

- smooth → peaks where normals cluster

# Normals - 2D Histogram Clustering



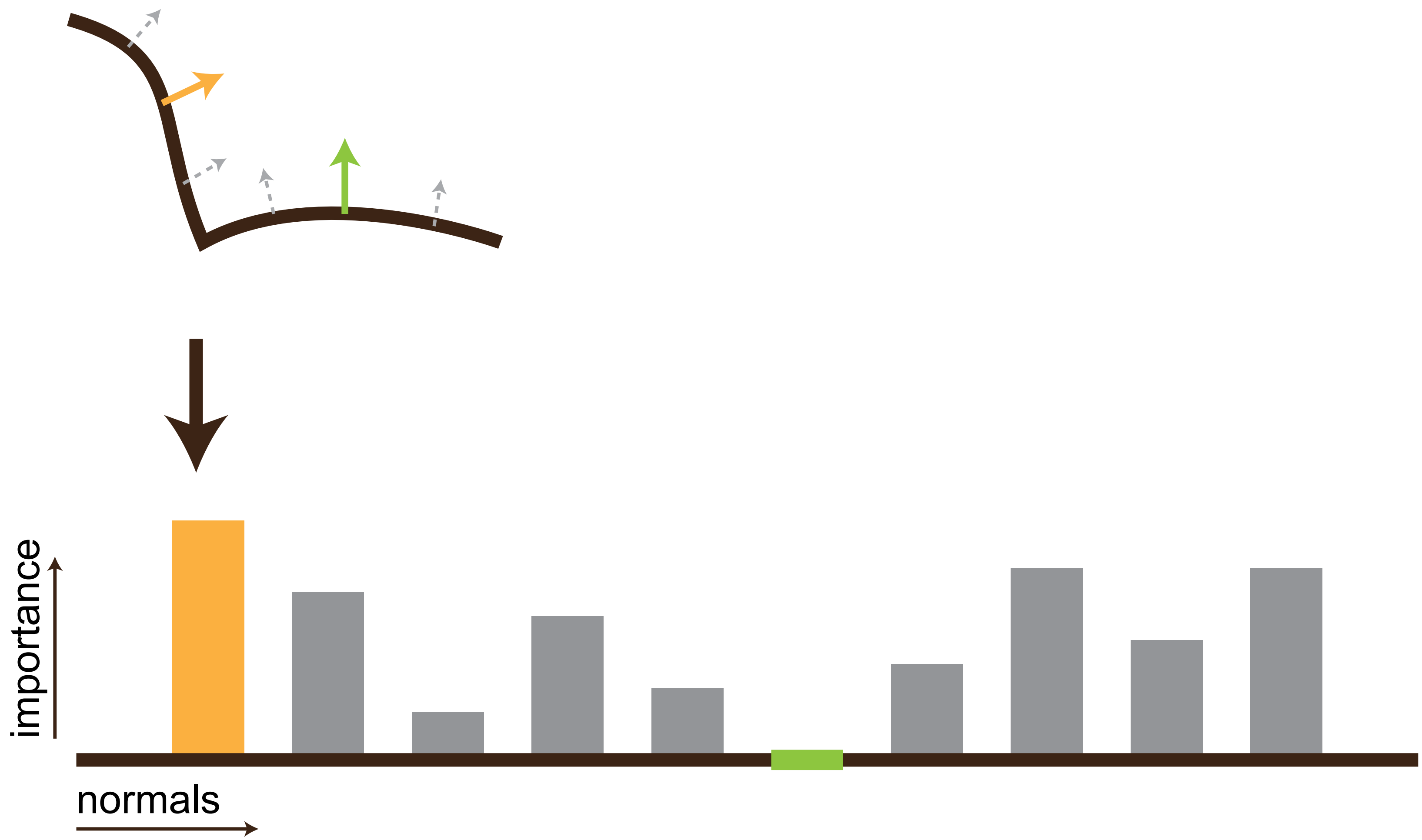

- smooth → peaks where normals cluster
- select direction

# Normals - 2D Histogram Clustering



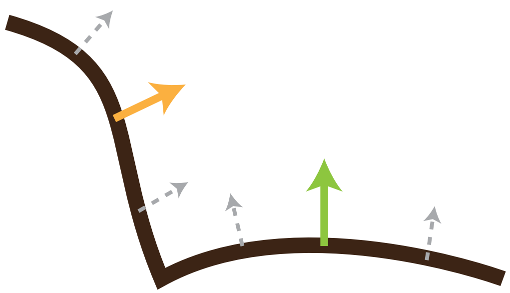

- smooth → peaks where normals cluster
- select direction
- reduce weights

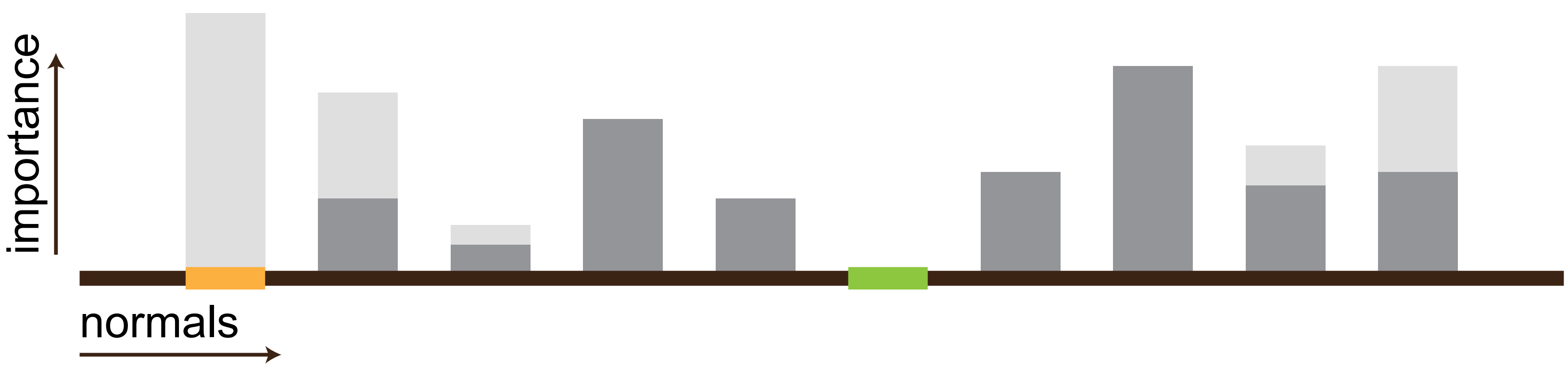
# Normals - 2D Histogram Clustering




- smooth → peaks where normals cluster
- select direction
- reduce weights

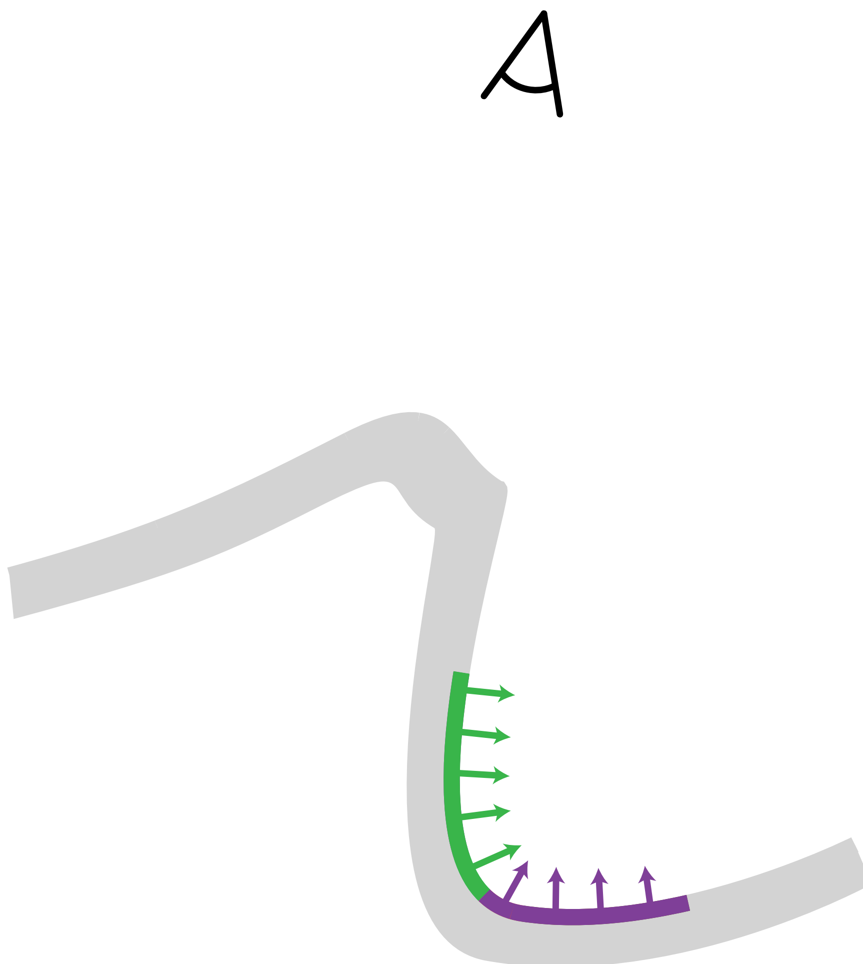
# Normals - 2D Histogram Clustering






- smooth → peaks where normals cluster
- select direction
- reduce weights

# Build Clusters




- assign pixels to clusters
  - based on normal alignment (+tile, +depth)

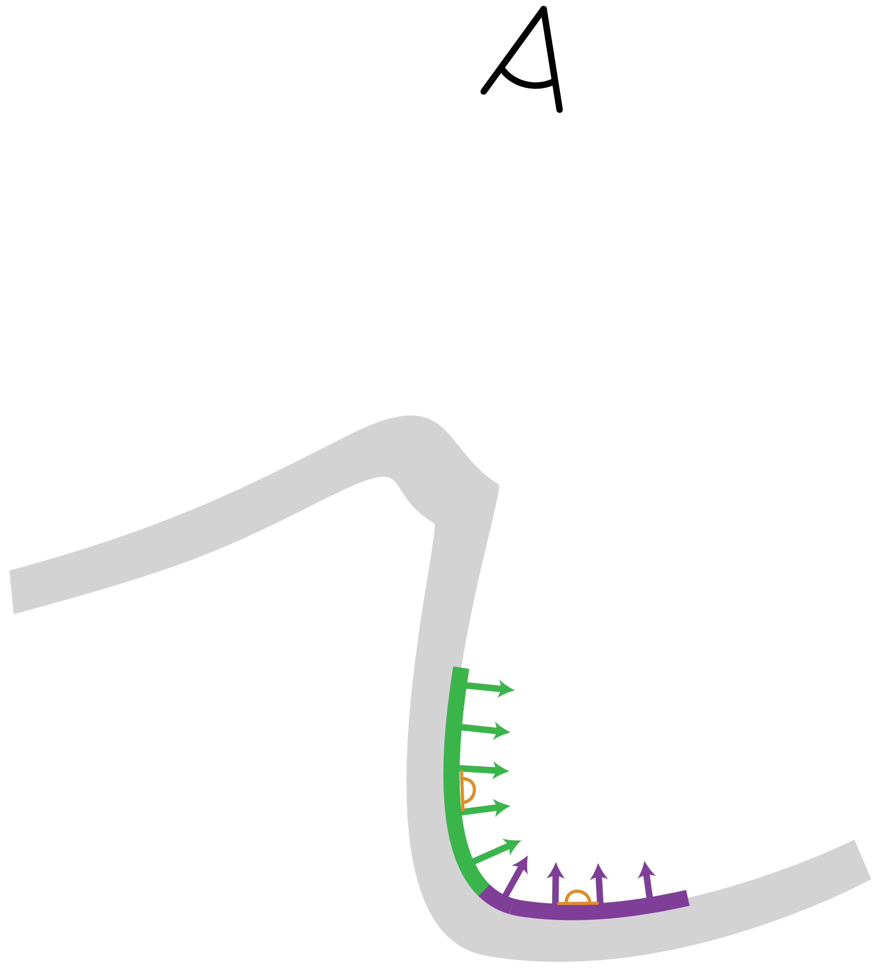
# Place Radiance Cache per Cluster

place radiance cache on pixel that maximizes

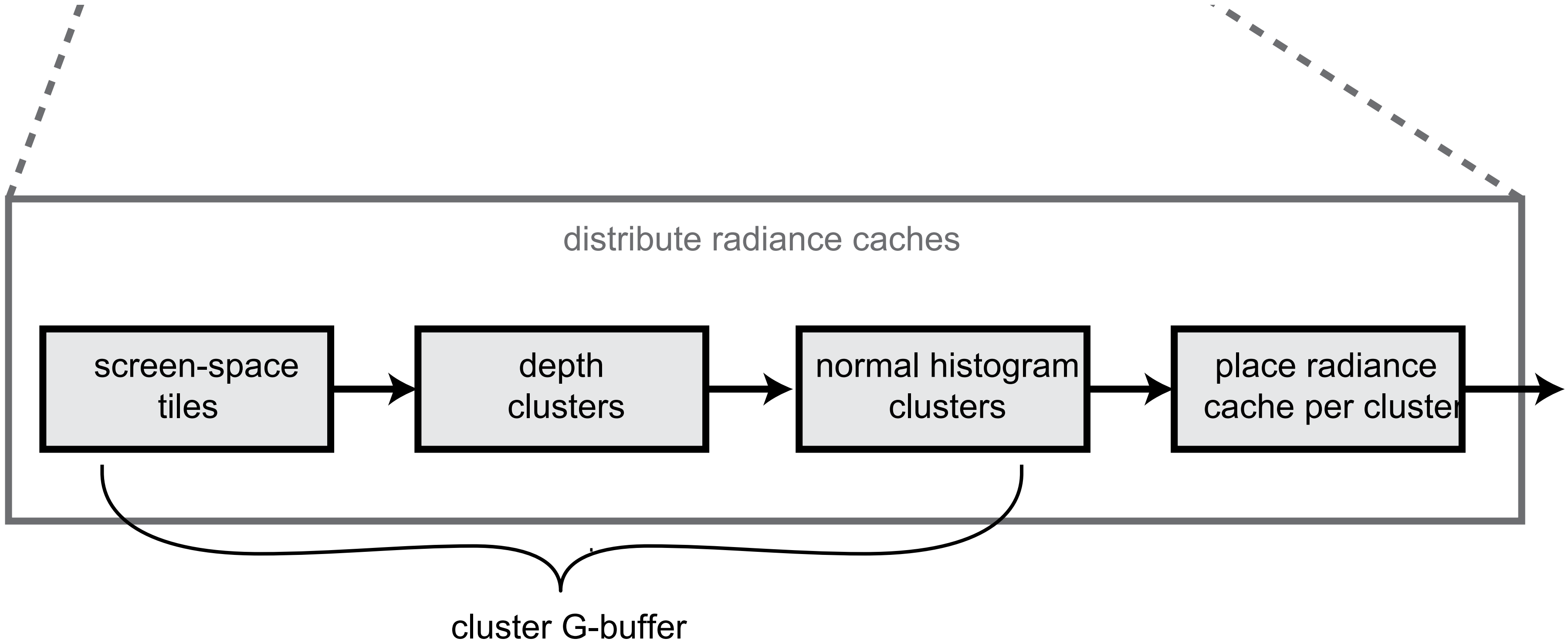
$$\frac{\text{normal alignment} + \text{surface offset}}{\text{distance to cluster center}}$$

$$\hat{=} \frac{\vec{n}_p \cdot \vec{n}_c + (\vec{p} - \vec{c}) \cdot \vec{n}_c}{1 + \vec{p}_{\text{proj } \vec{c}}}$$

# Place Radiance Cache per Cluster

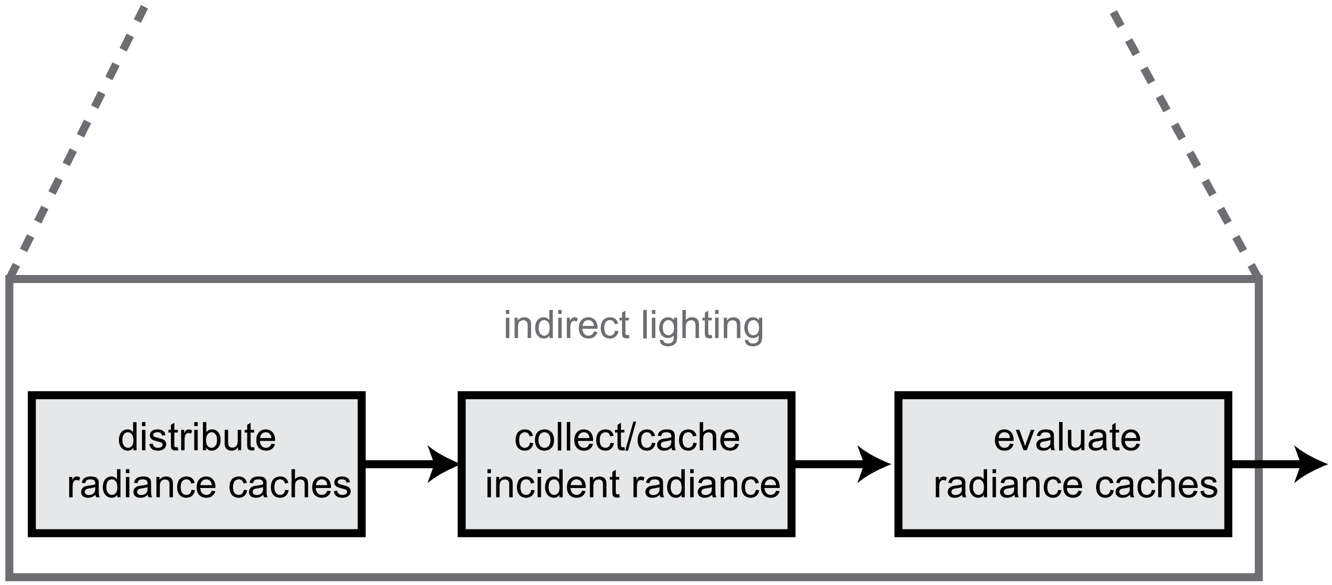


# Radiance Cache Distribution



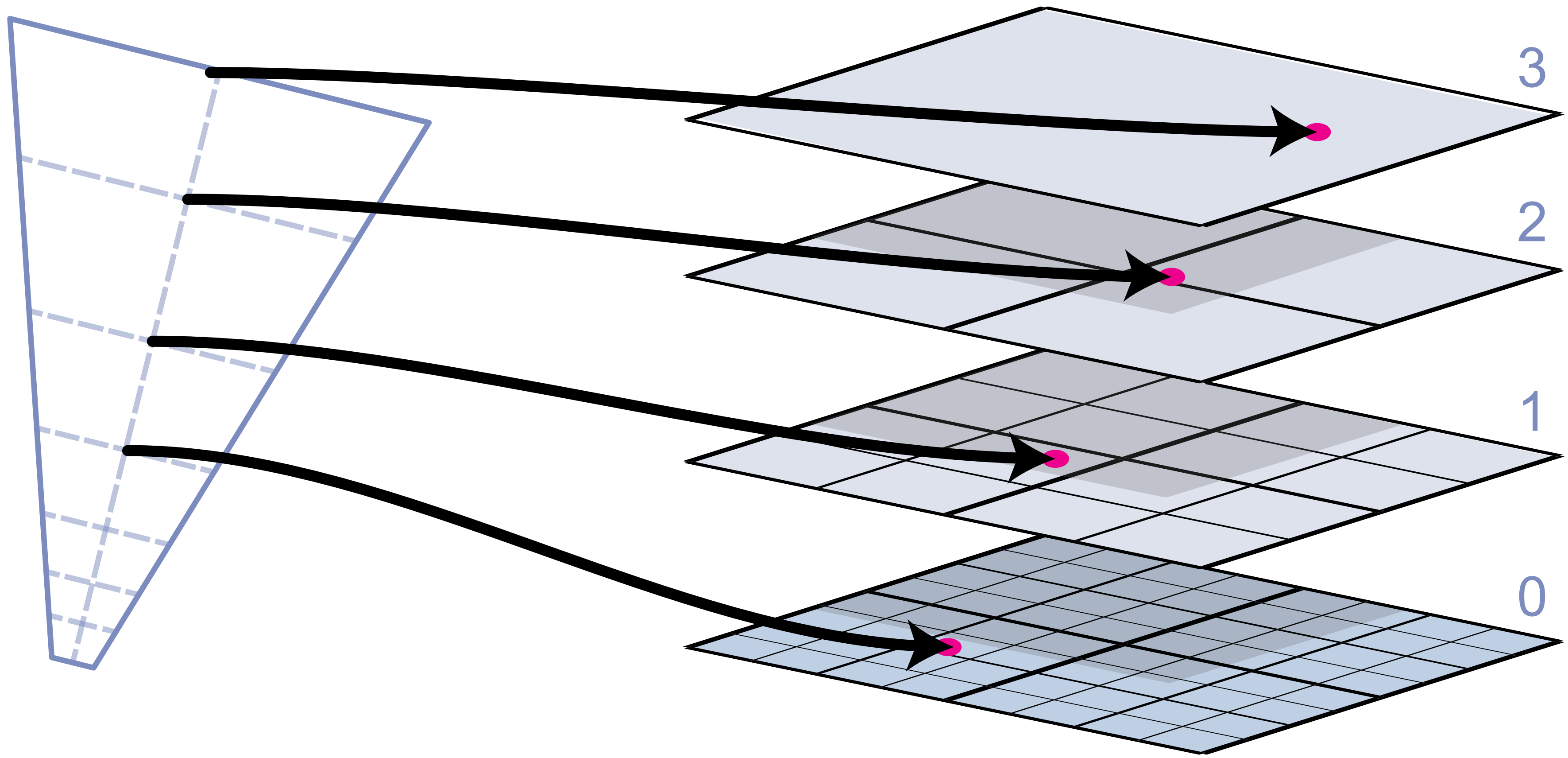

- cluster by tile and depth
- cluster from smoothed normal histogram
  - repeatedly choose direction and reduce weights
- place radiance cache on representative pixel

# Clustered Pre-convolved Radiance Caches



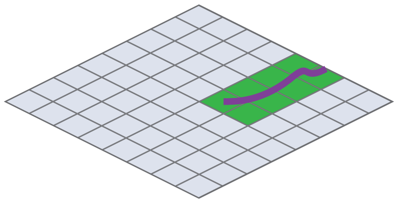

- distribute radiance caches
- collect incident radiance

# Collect Incident Radiance



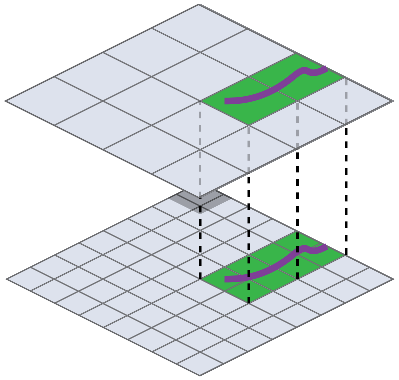

- one cone per pixel of radiance cache

# Collect Incident Radiance



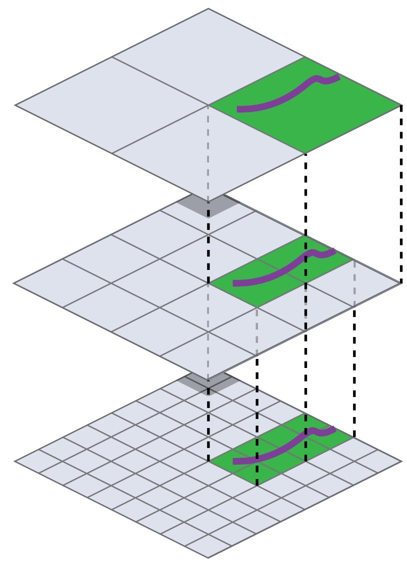
- one cone per pixel of radiance cache

# Collect Incident Radiance



- one cone per pixel of radiance cache
- mip-mapping approach

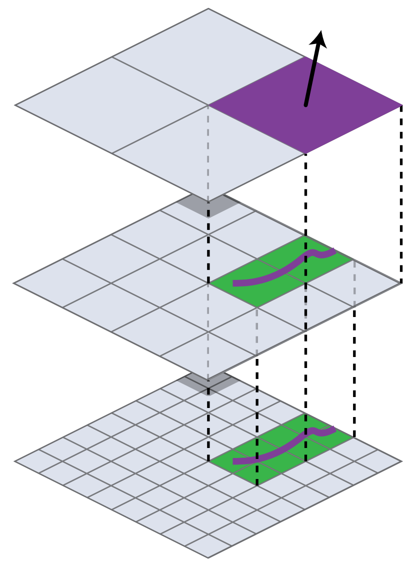
# Collect Incident Radiance



- one cone per pixel of radiance cache
- mip-mapping approach with *surface coverage*

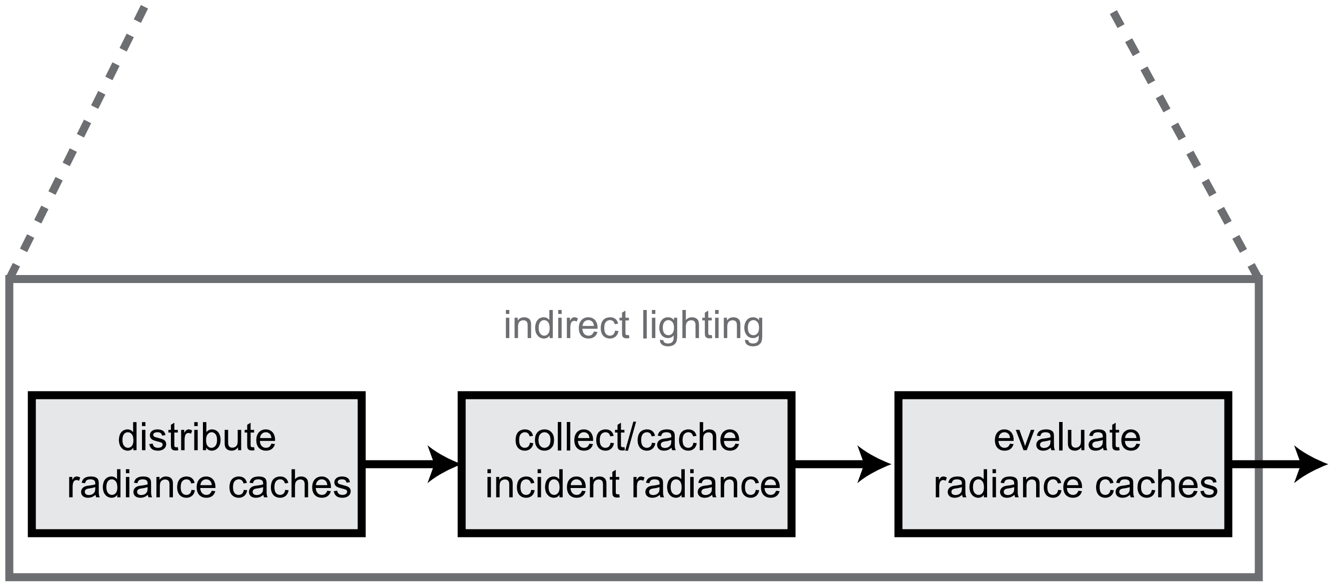
$$L_{i+1}(v) = \frac{\sum_k^8 L_i(v,k)}{4}$$

# Collect Incident Radiance



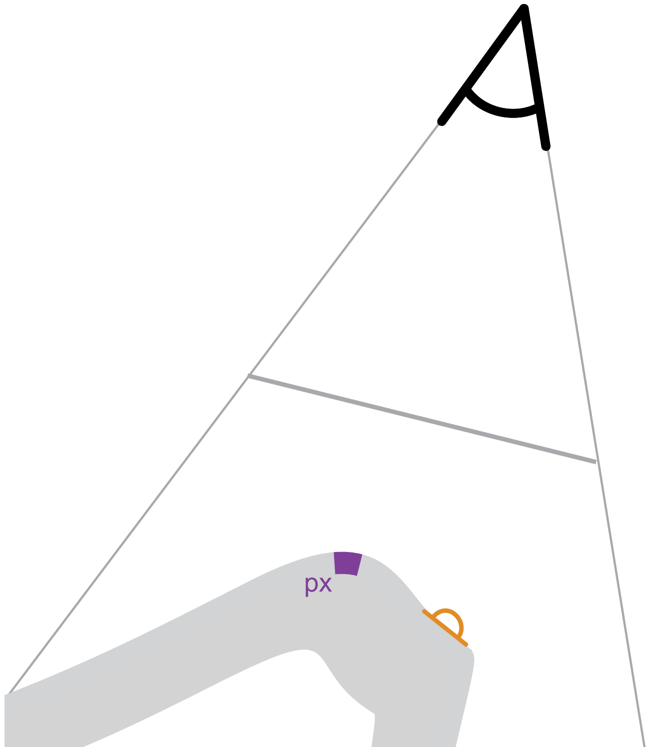
- one cone per pixel of radiance cache
- mip-mapping approach with *surface coverage*
- albedo & surface normal per voxel

# Clustered Pre-convolved Radiance Caches



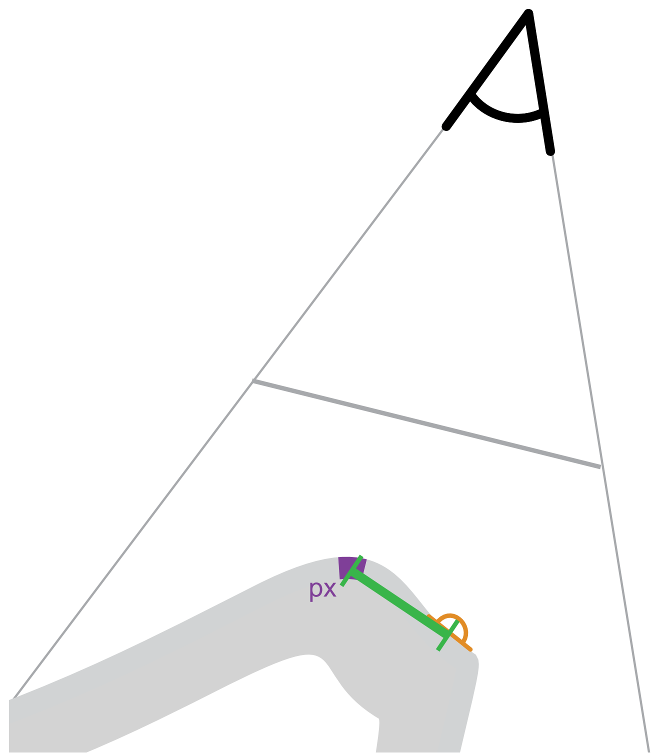

- distribute radiance caches
- collect incident radiance
- evaluate radiance caches

# Evaluate Pre-Convolved Radiance Caches



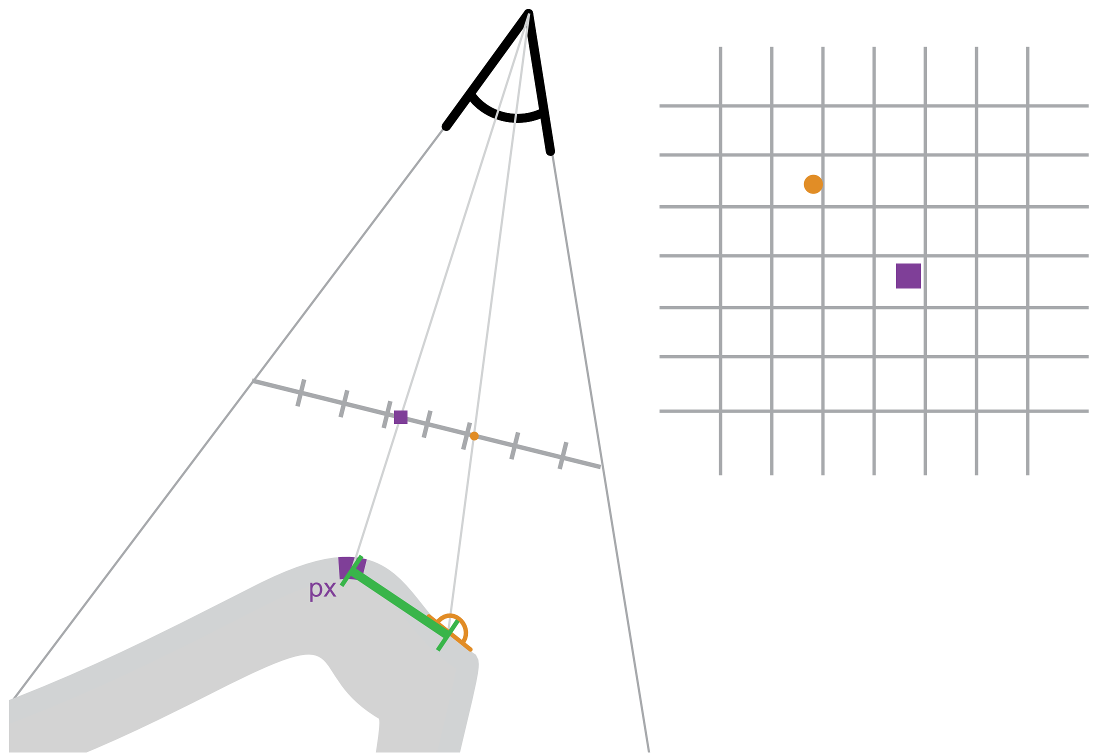

- evaluate radiance caches per pixel

# Evaluate Pre-Convolved Radiance Caches

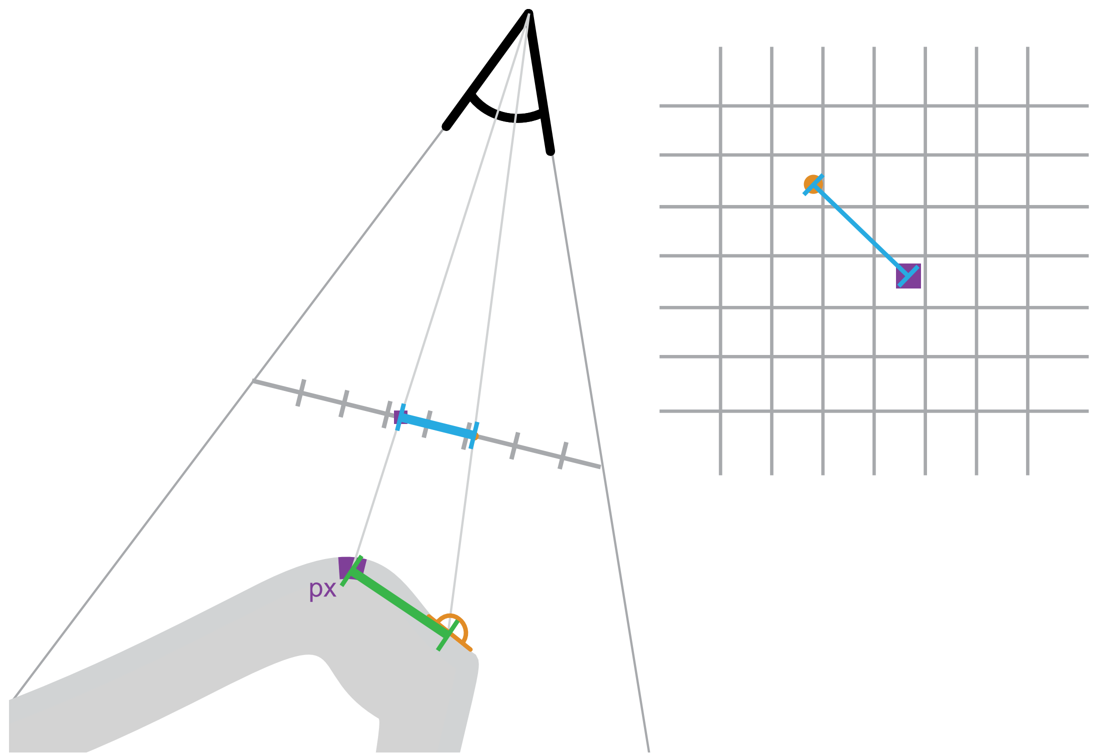



- splatting:
  - requires large splats
  - hard to implement efficiently in CUDA

# Evaluate Pre-Convolved Radiance Caches

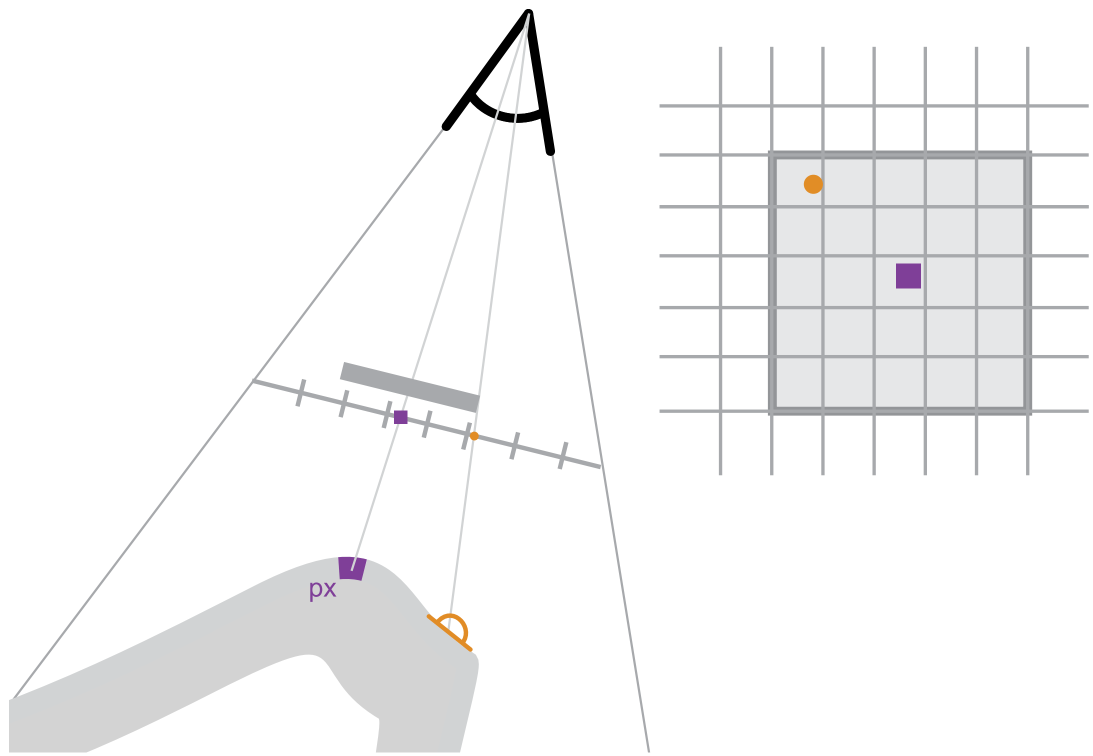


# Evaluate Pre-Convolved Radiance Caches



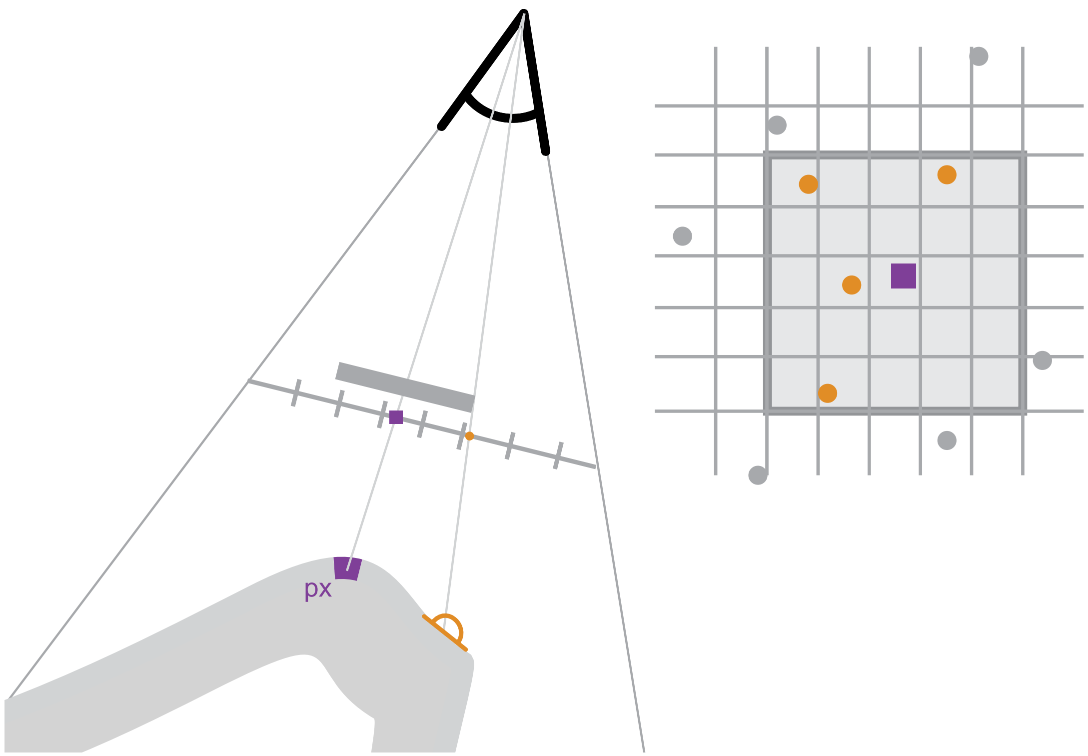

- introduce screen-space weight

# Evaluate Pre-Convolved Radiance Caches




- switch to a gathering approach over screen-space tiles

# Evaluate Pre-Convolved Radiance Caches

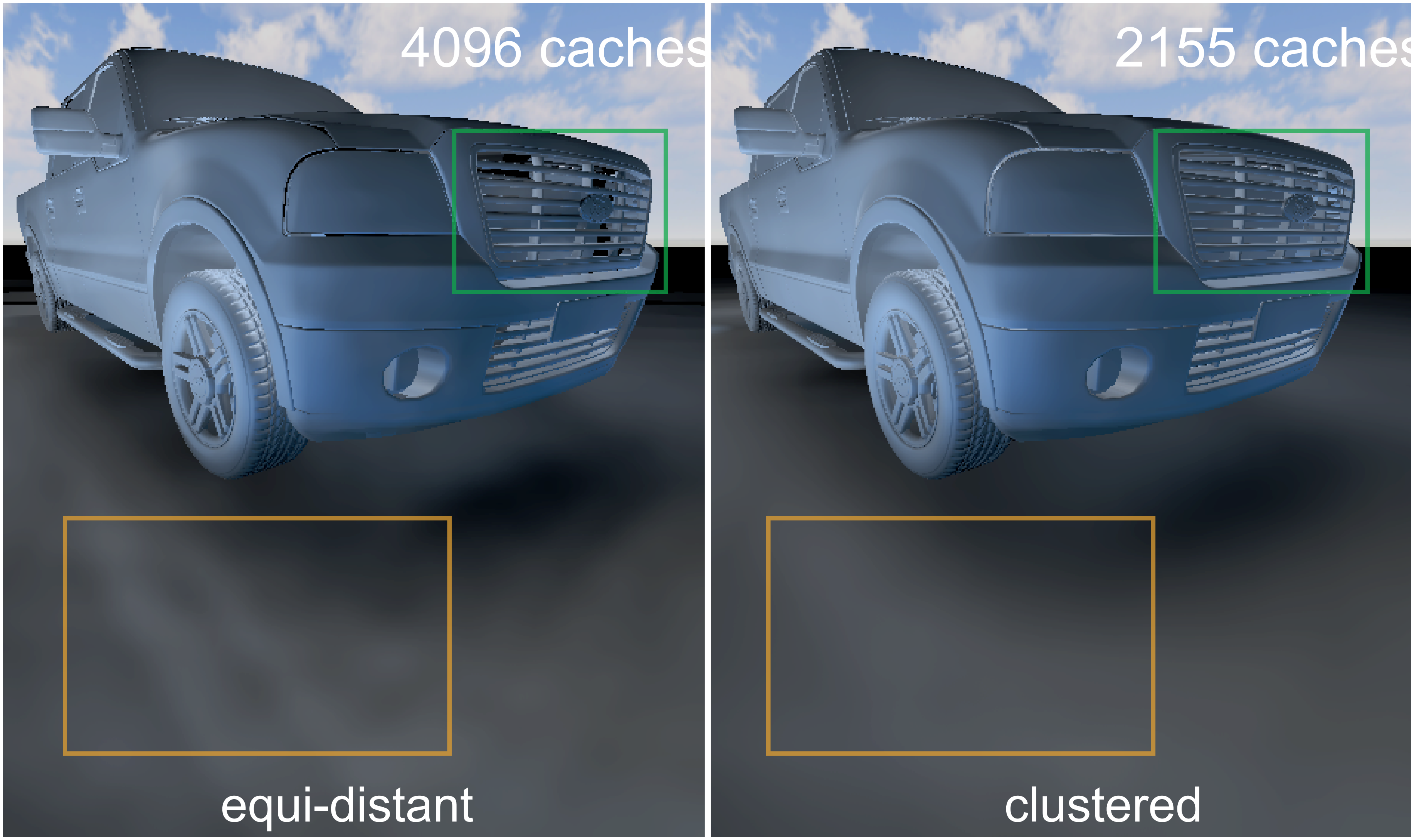



- use world-space weight
- add screen-space weight
- switch to a gathering approach over screen-space tiles

# Results

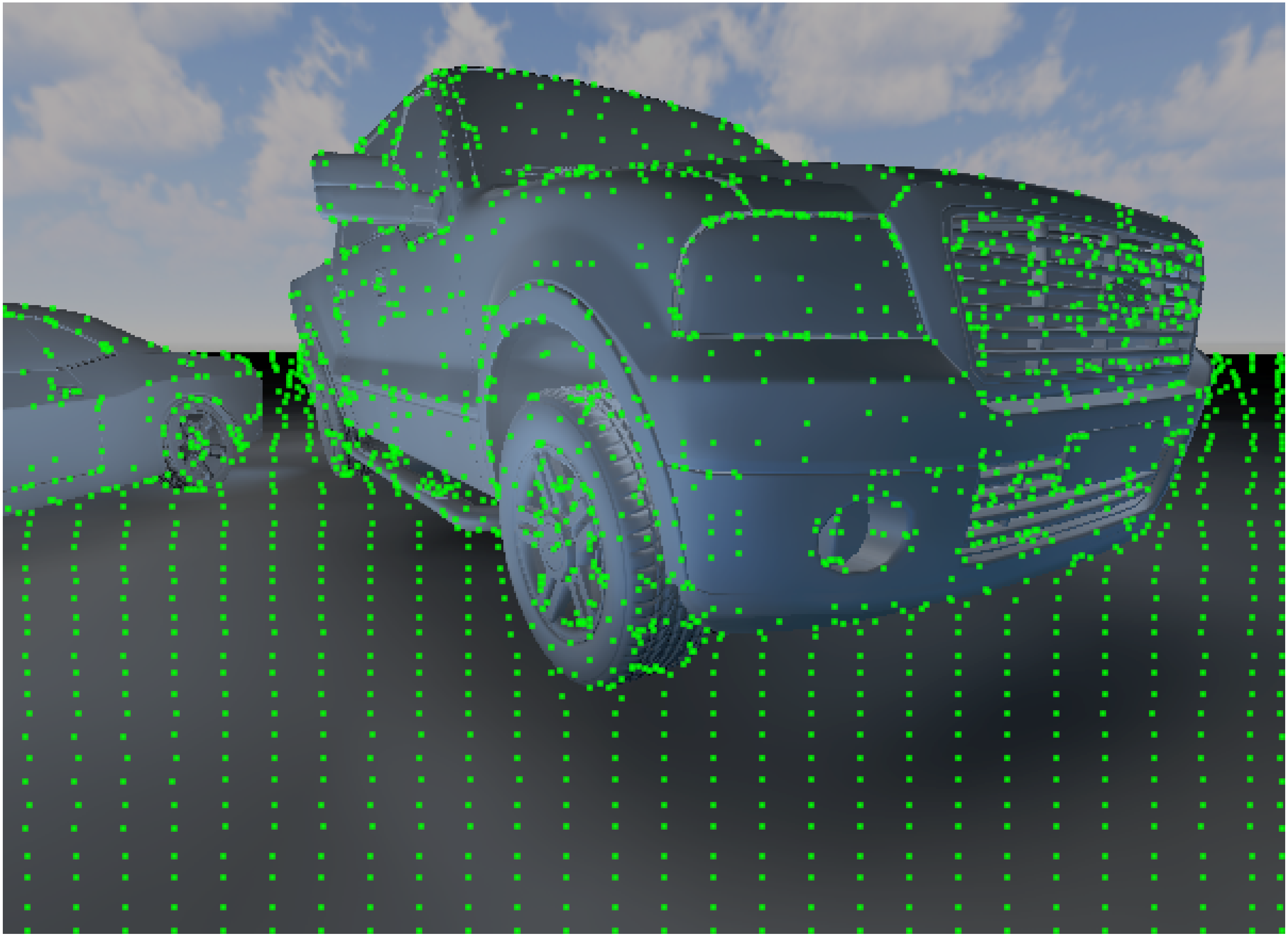
- indirect illumination only!

# Equi-distant vs. Clustered Distribution



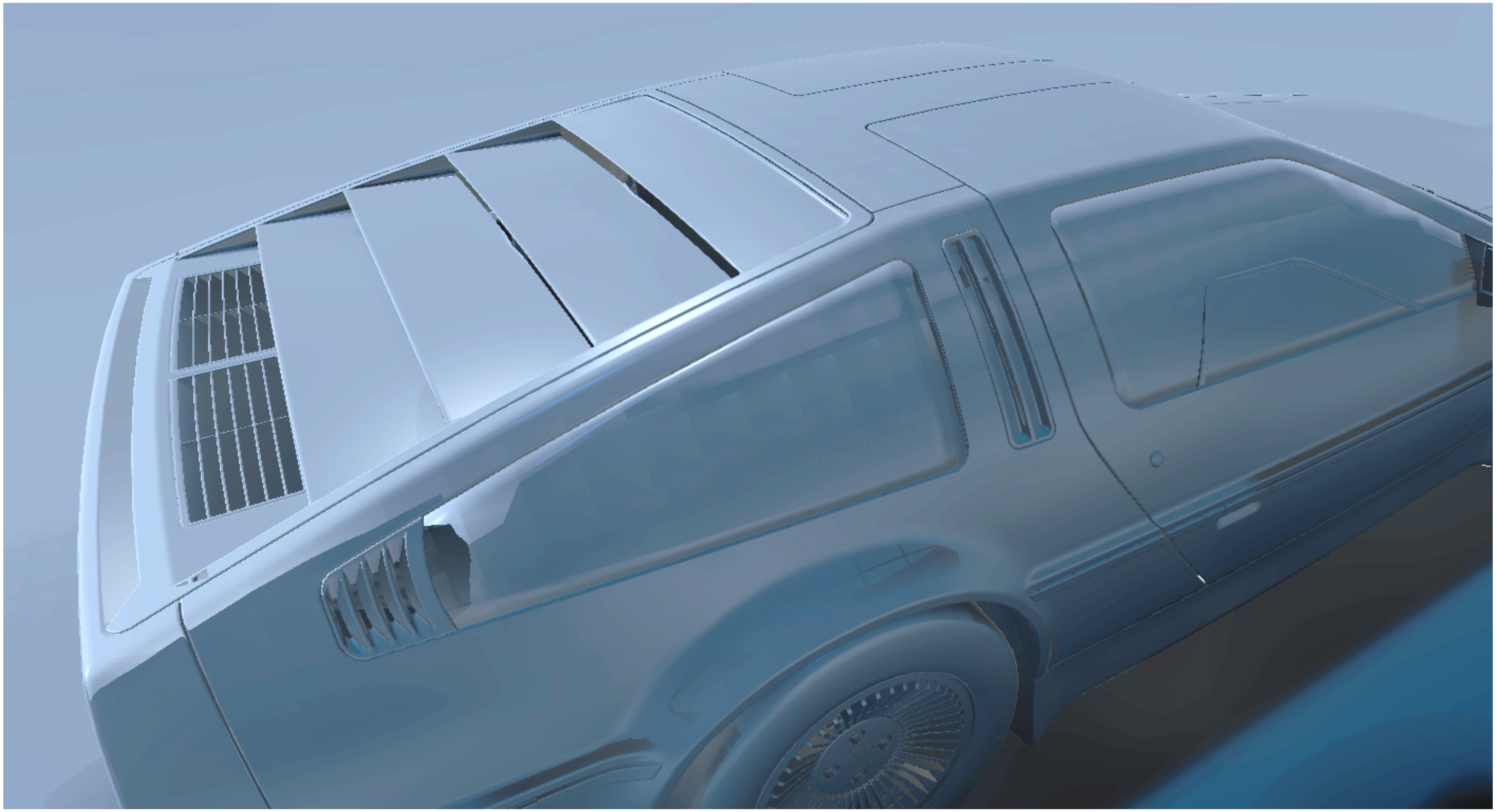

- high-frequency geometry represented much better
- large improvement even on flat surfaces

# Clustered Distribution



# Quality Comparison

# Final



# Conclusion

- interactive, high-quality radiance cache distribution
- interactive, fully dynamic and fully parallel indirect illumination
- glossy scenes are handled well
- not temporaly stable (future work)
- some artifacts remain (voxelization)

# Questions?

**Clustered Pre-convolved Radiance Caching**

Hauke Rehfeld (*hauke.rehfeld@kit.edu*), Tobias Zirr, Carsten Dachsbacher