View-Region Optimized Image-Based Scene Simplification

Puneet Lall, Silviu Borac, Dave Richardson,
Matt Pharr, Manfred Ernst

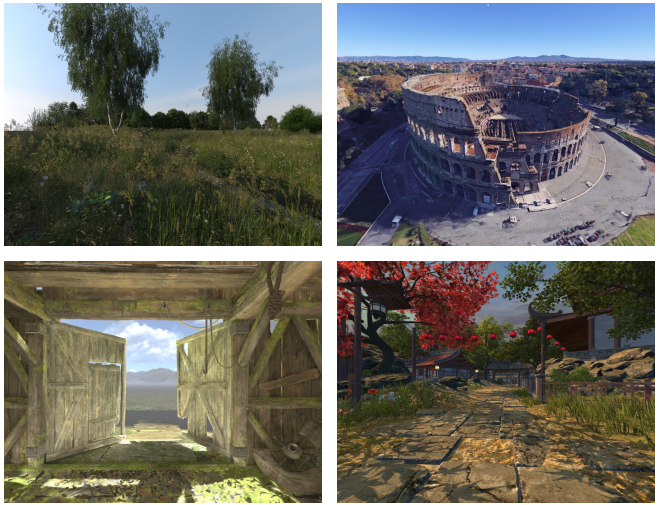
Google

Motivation

Creating Good VR Content is Challenging

Motivation

Goal



Offline Processing

Environments created from…

- Path tracers
- Game engines
- Photogrammetry

Real-time rendering on a mobile-class HMD

Requirements

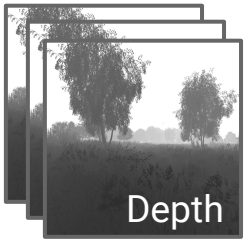
- 6 DoF rendering
- Flexible ingest
- Low, predictable rendering cost
- Edge anti-aliasing
- Low memory footprint

Limitations

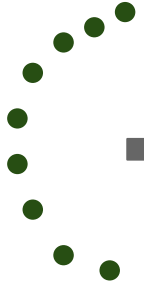
- Limited view region
- Pre-processing
- Static geometry
- Prebaked shading

Our Method

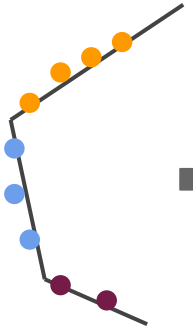
Our Method



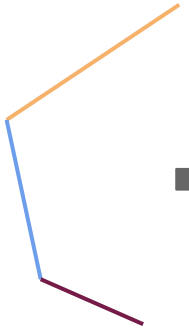
Scene Capture



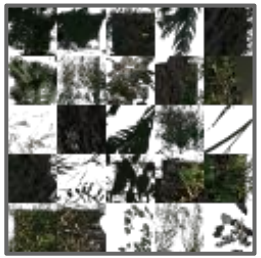
Point Merging



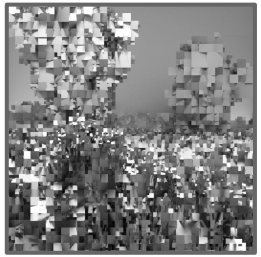
Tiling



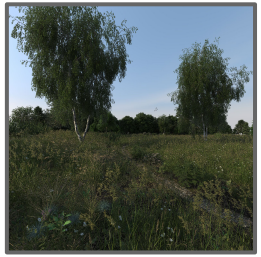
Texturing and Compression



Texture Atlas

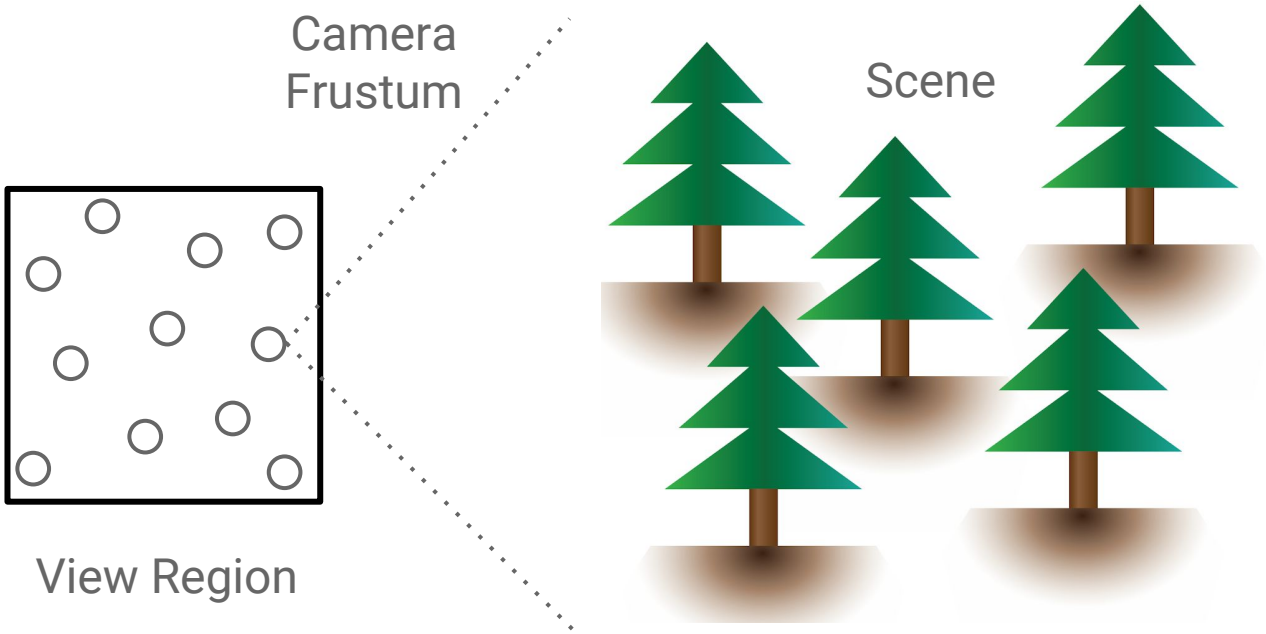


Mesh

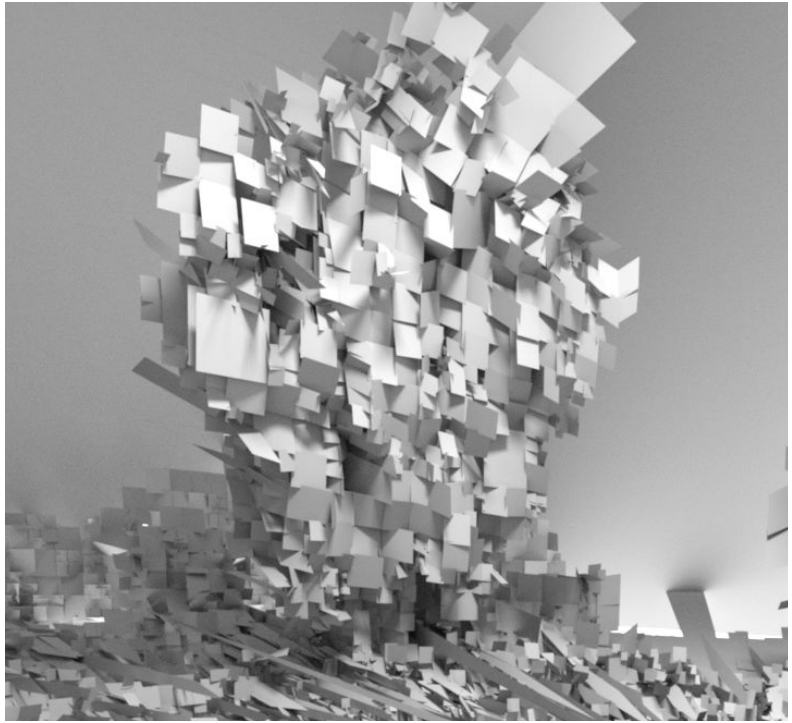


Real-Time Rendering

Scene Capture

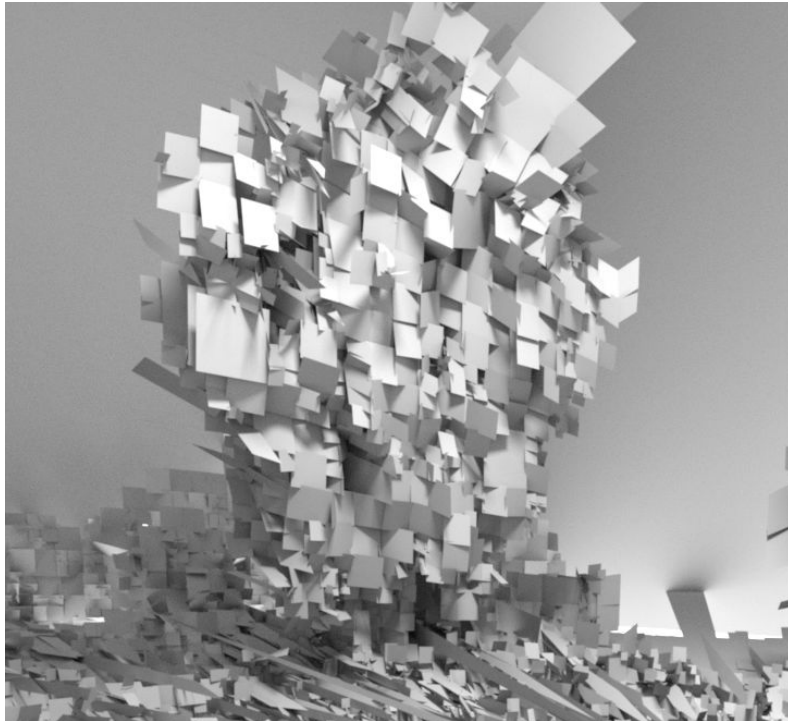


Tiling



Tiling - Goals

- Geometric fit
- Constrained runtime cost
  - Primitive count
  - Area
  - Fillrate

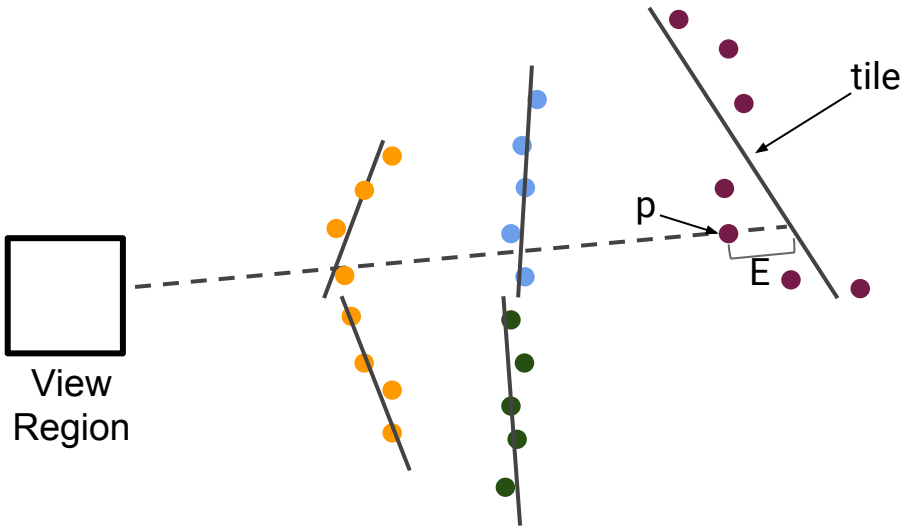


Tiling – Model

- Generate quad tiles, ***T***
- ***E(p, t)***: Cost of modeling ***p*** with ***t***
- ***Weight(t)***: Resources required by ***t***
  - Primitive count
  - Texture area
  - Overdraw
- Objective

$$\underset{T}{\text{minimize}} \quad \sum_{p \in P} \min_{t \in T} E(p, t)$$

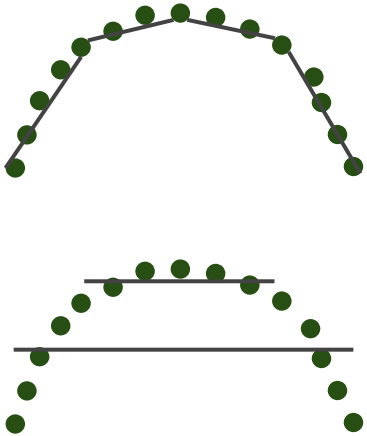
$$\text{subject to} \quad \sum_{t \in T} \text{Weight}(t) \leq \text{Capacity},$$



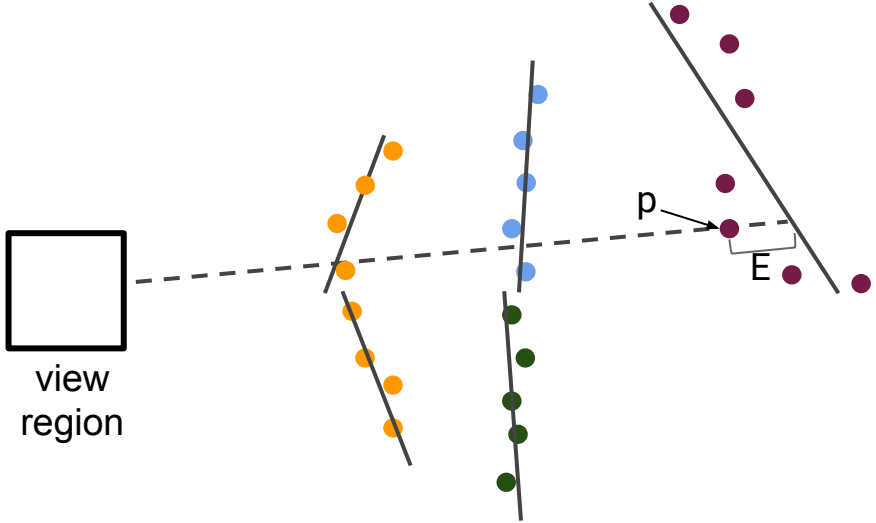
Tiling - Method

Two Step Algorithm:

1. Generate diverse, overcomplete set of candidate tiles
2. Select tiles subject to capacity constraint

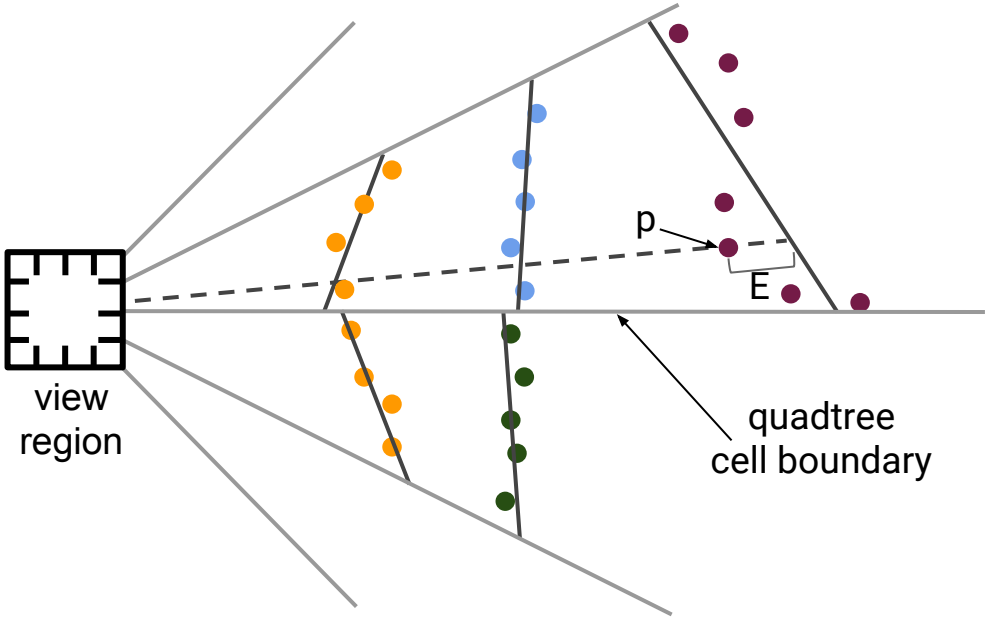


Tiling – Candidate Generation



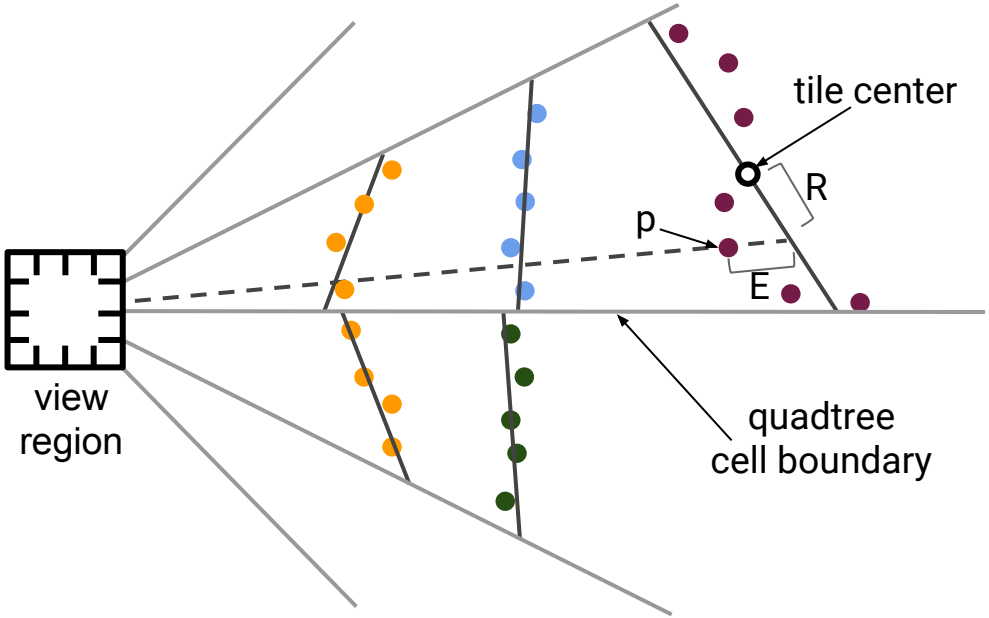
Tiling – Candidate Generation

- Perspective-space quadtree



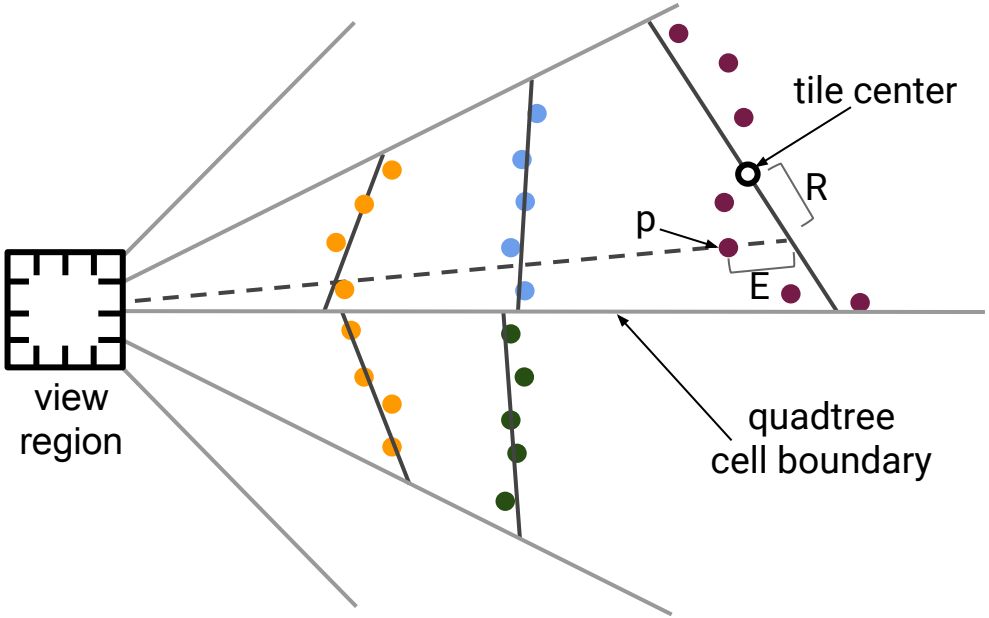
Tiling – Candidate Generation

- Perspective-space quadtree
- Model tiles as disks
  - Minimize ***E(p, t) + R(p, t)***



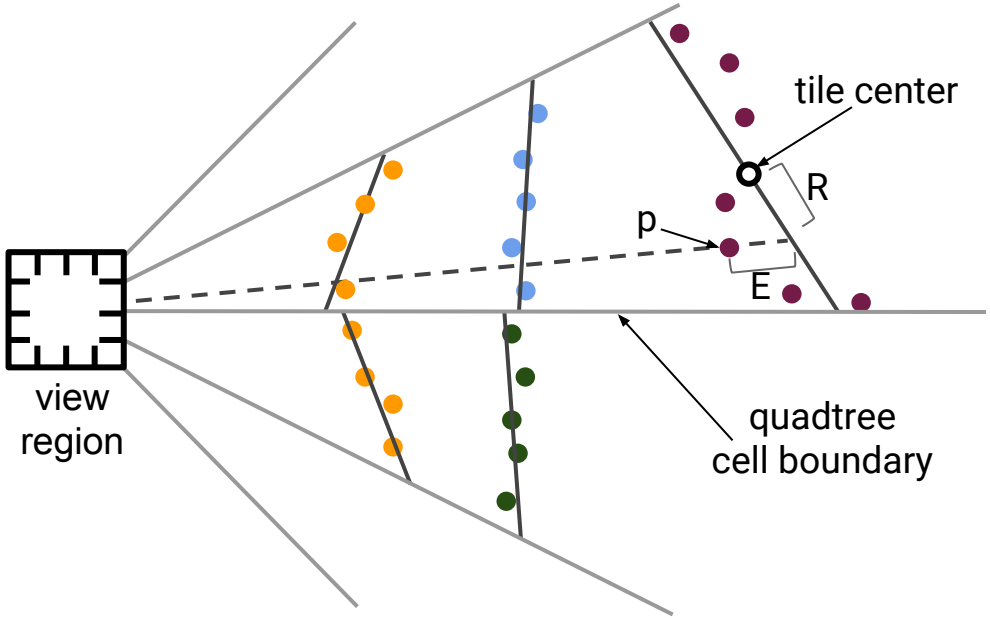
Tiling – Candidate Generation

- Perspective-space quadtree
- Model tiles as disks
  - Minimize ***E(p, t) + R(p, t)***
- Iterative clustering
  - Assignment, Update, …



Tiling – Candidate Generation

- Perspective-space quadtree
- Model tiles as disks
  - Minimize ***E(p, t) + R(p, t)***
- Iterative clustering
  - Assignment, Update, …
- Repeat with 1…K tiles per cell



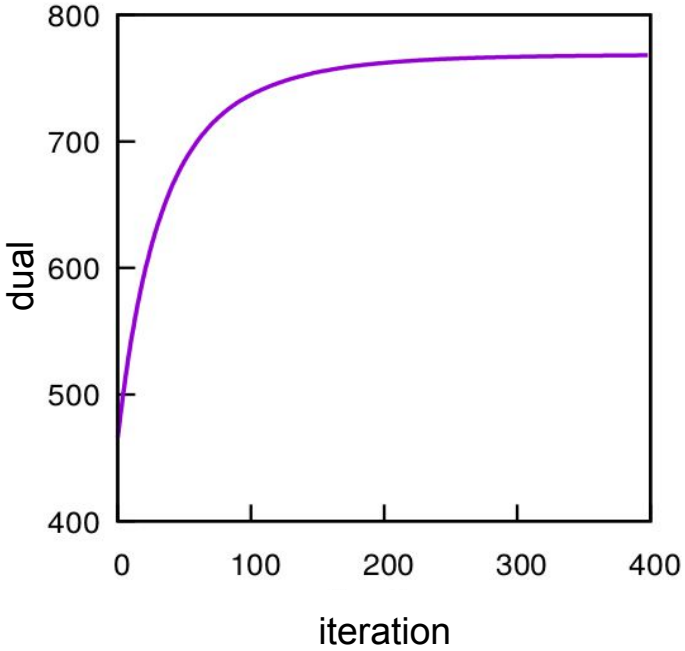
Tiling – Candidate Selection

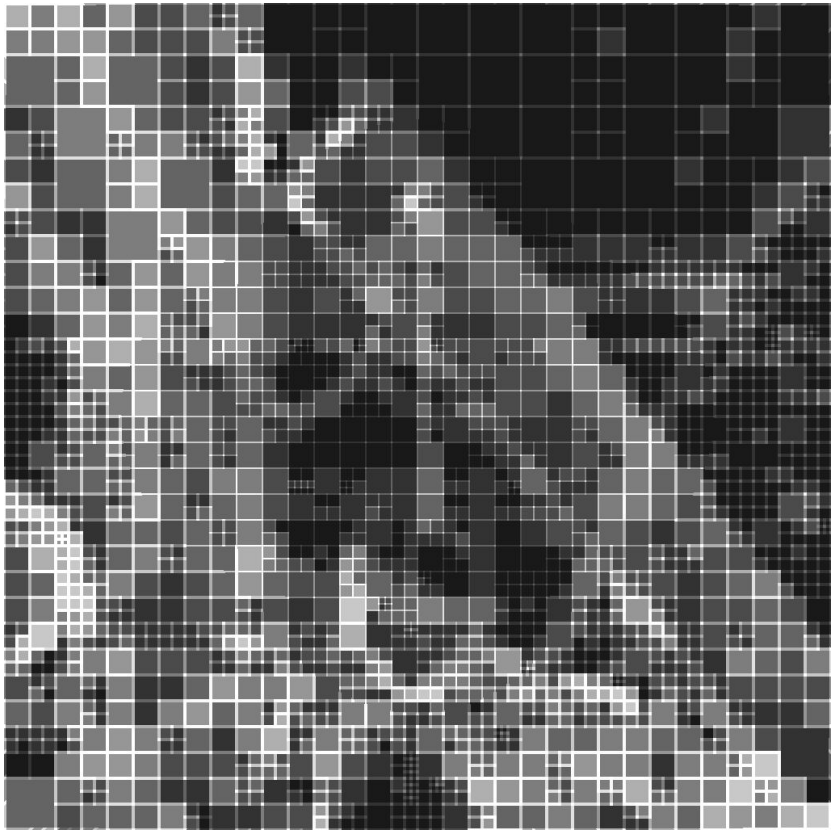
- >1M candidates to select from, ***s***
- Linear integer program:

$$\underset{s}{\text{minimize}} \quad d^T s$$

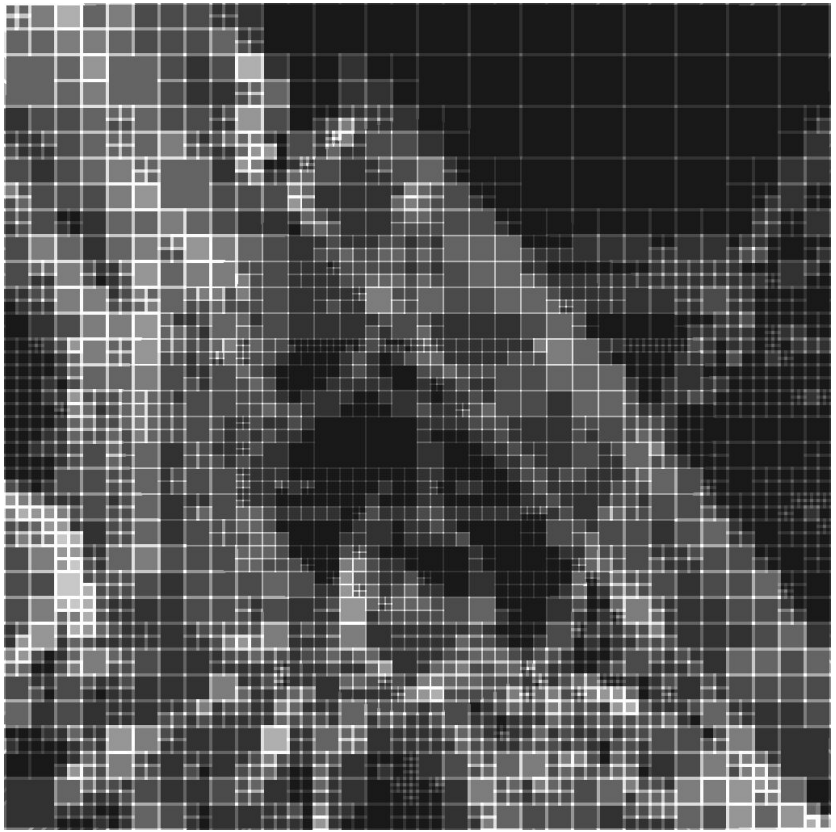
$$\text{subject to} \quad \begin{cases} \text{ValidTiling}(s) \\ w \cdot s \leq \text{Capacity} \end{cases}$$

- Solve with a dual decomposition and subgradient ascent

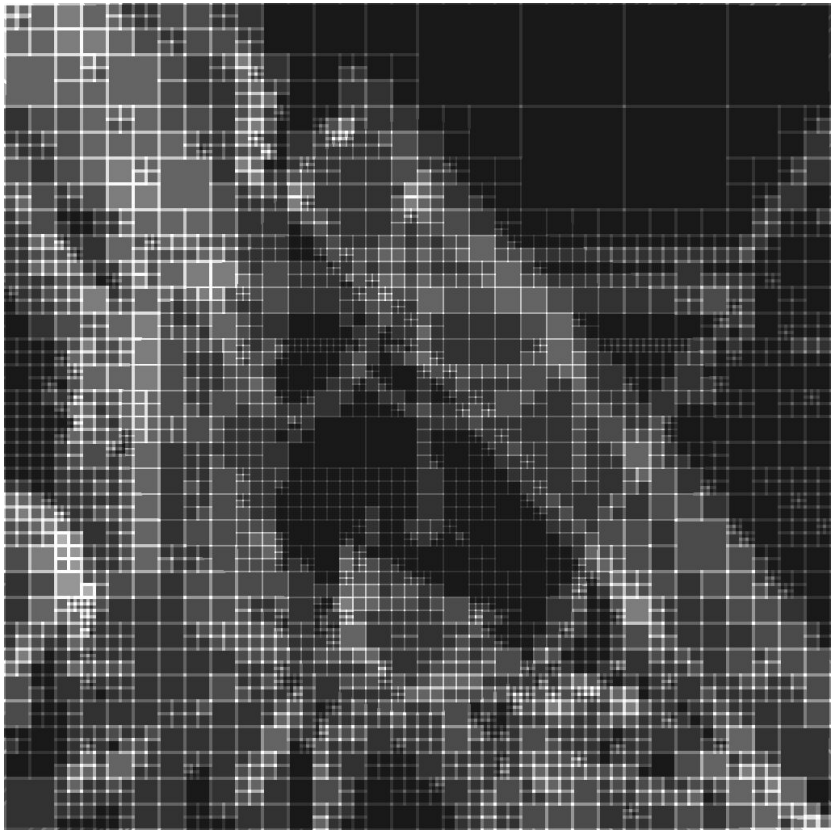




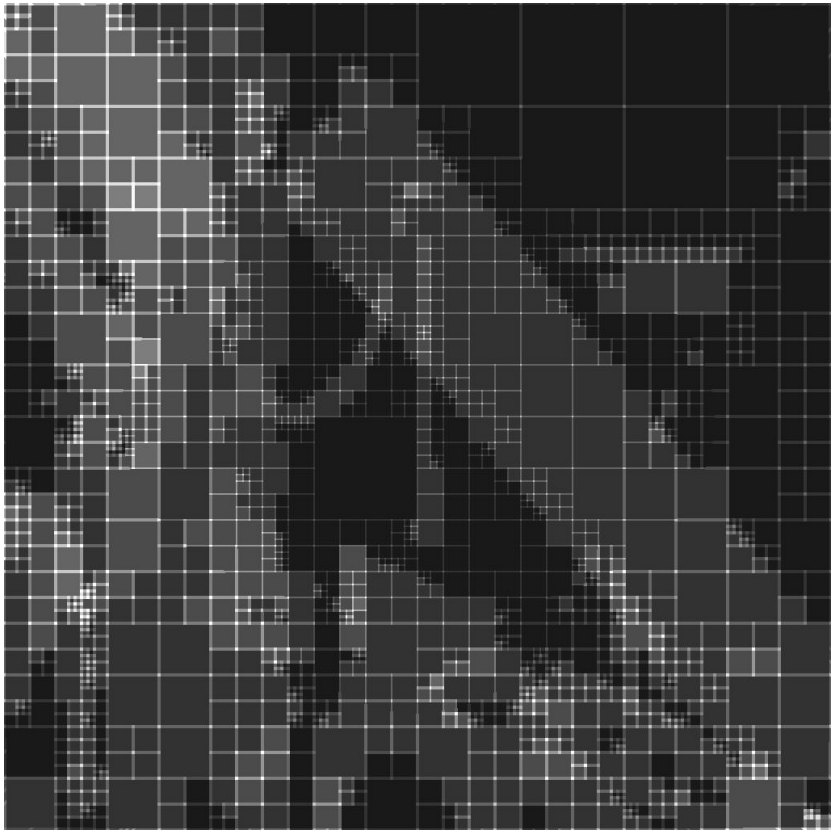
Results - 3.5x overdraw



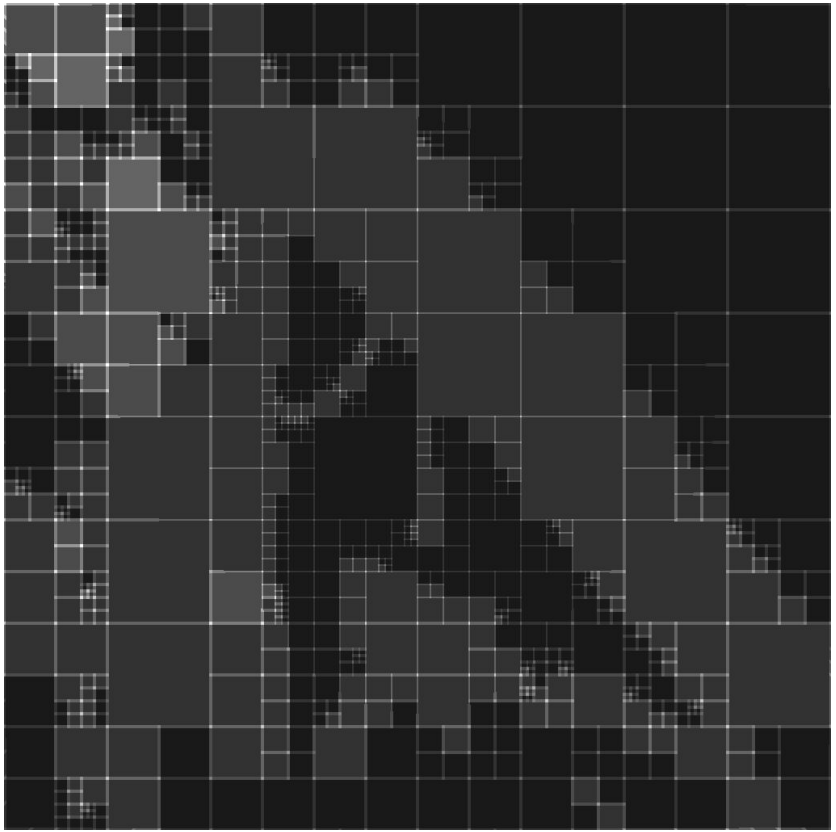
Results - 3.0x overdraw



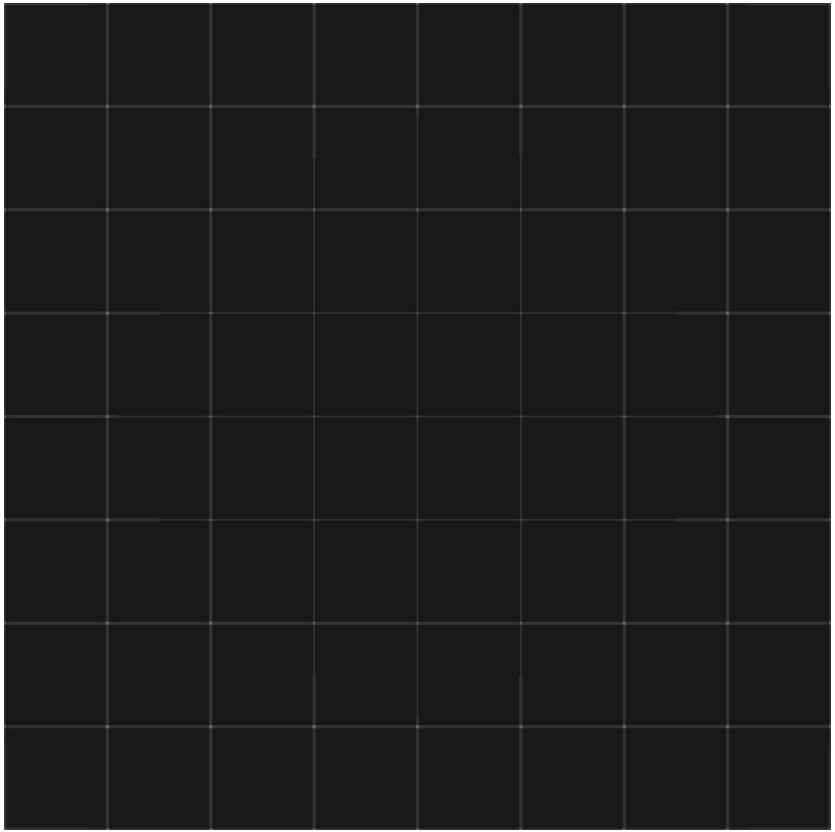
Results - 2.5x overdraw



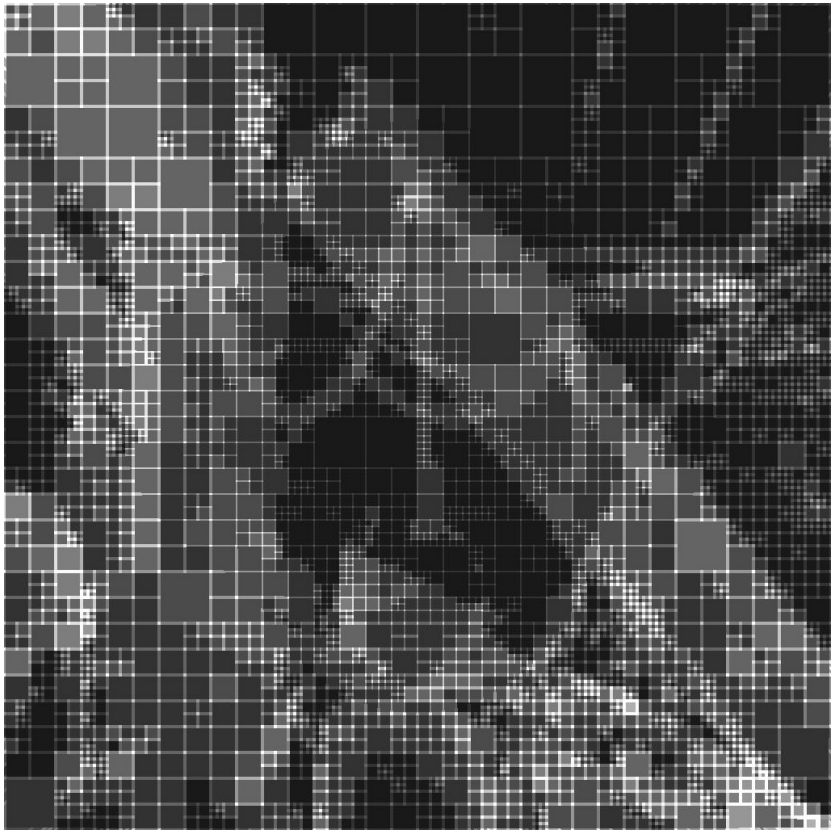
Results - 2.0x overdraw



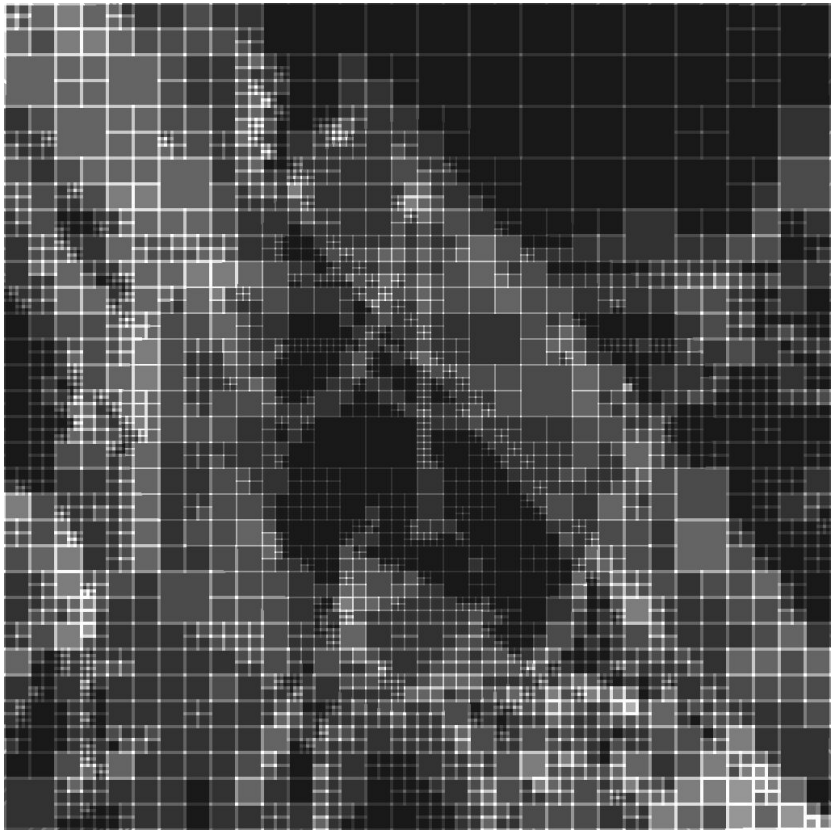
Results – 1.5x overdraw



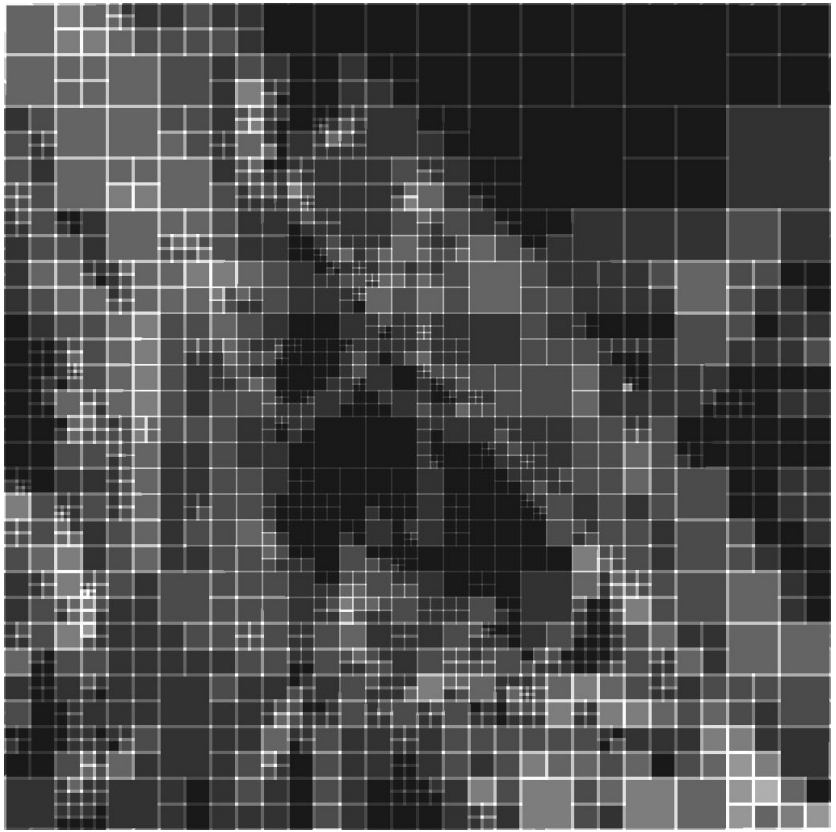
Results – 1.0x overdraw



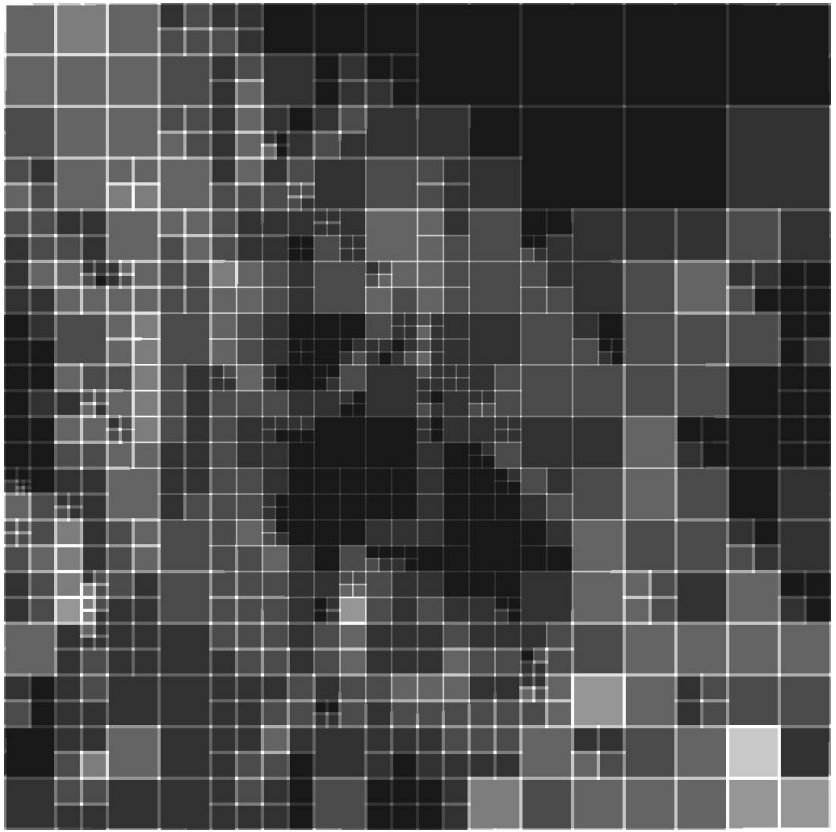
Results - 128k triangles



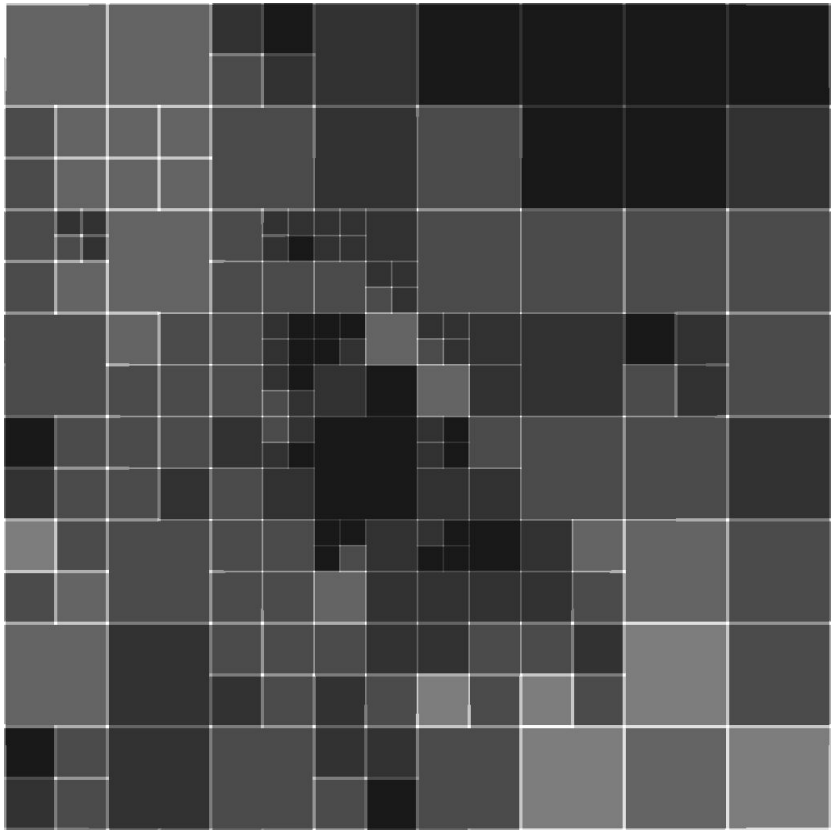
Results - 64k triangles



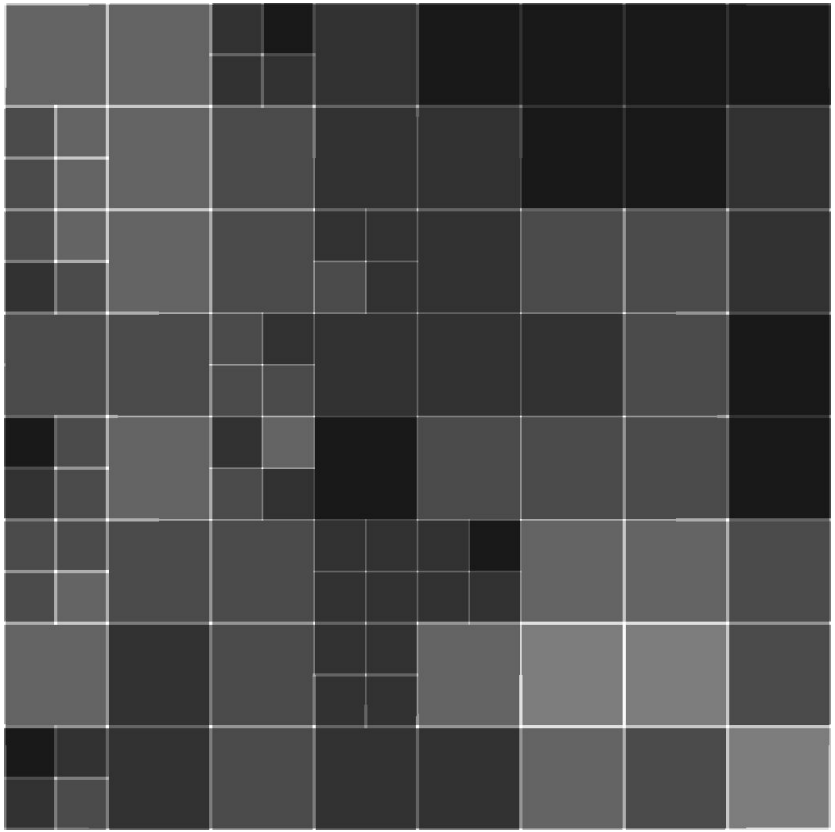
Results - 32k triangles



Results - 16k triangles



Results - 8k triangles

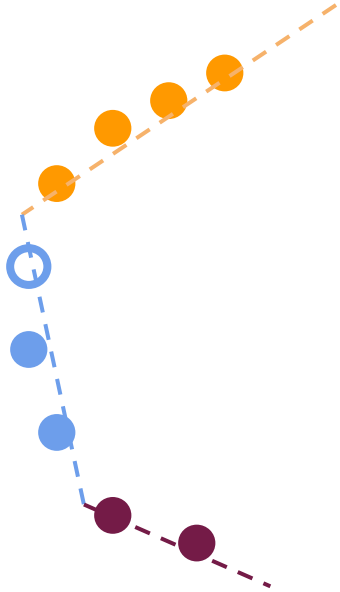


Results - 4k triangles

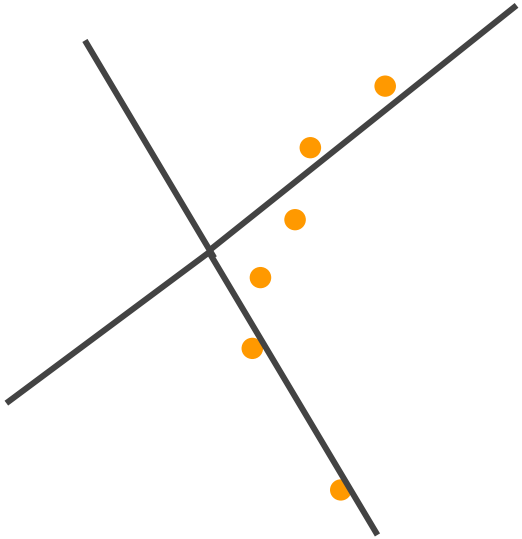
Texturing

Texturing - Point Splatting

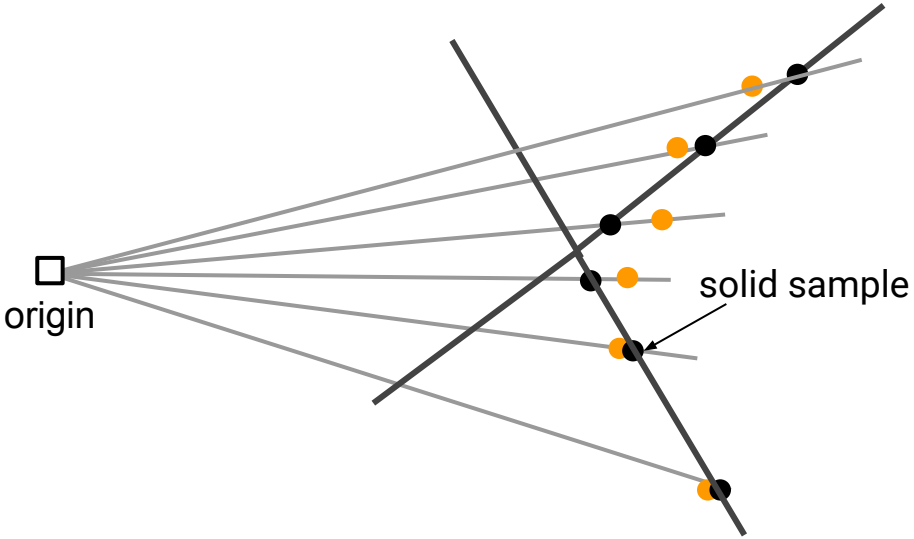
- Works for surface radiance
- Problematic for silhouettes/alpha
  - Kernel shape?
  - Cracks vs. edge-thickening
  - Unknown sampling rate
  - Subpixel geometry



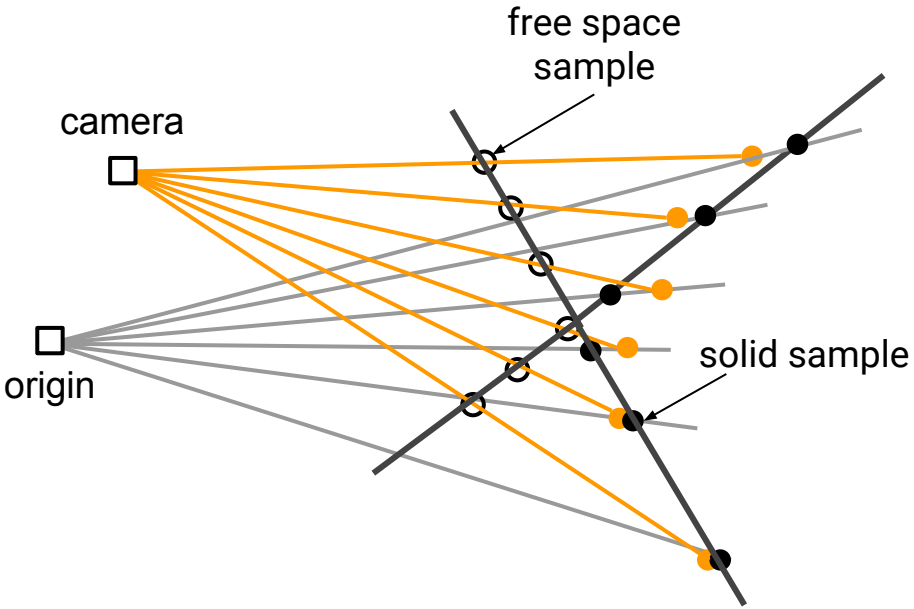
Texturing – Space Carving



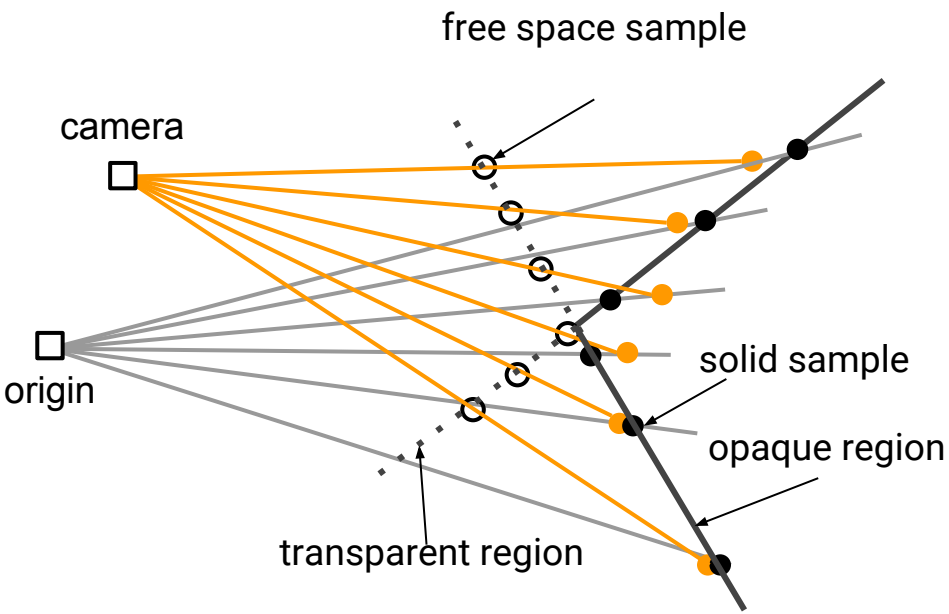
Texturing – Space Carving



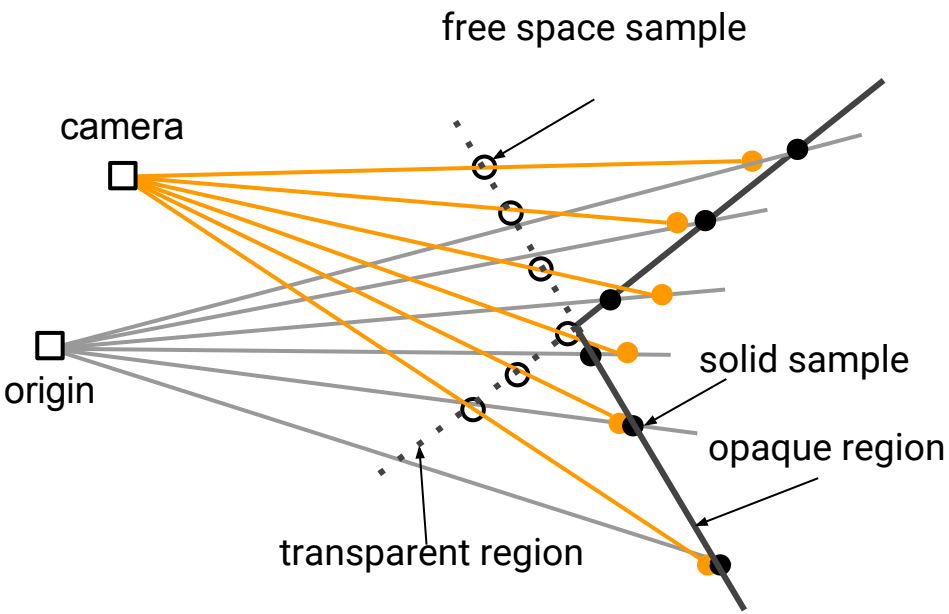
Texturing – Space Carving

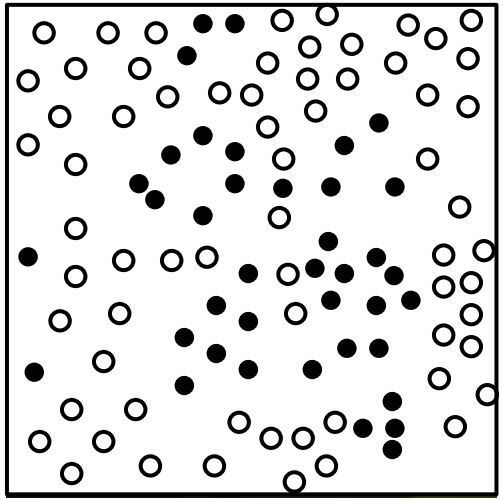


Texturing – Space Carving



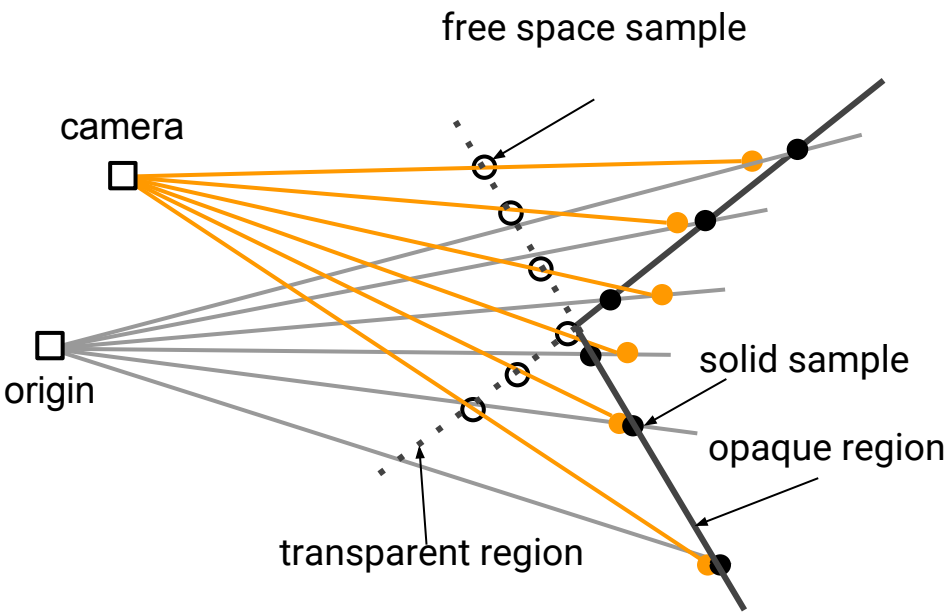
Texturing – Space Carving

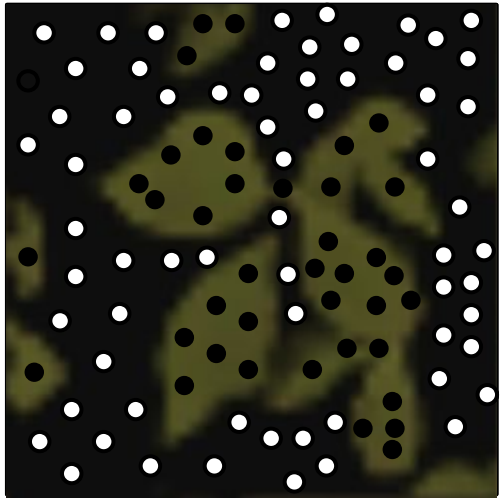




silhouette accumulator

Texturing – Space Carving





silhouette accumulator

Texturing

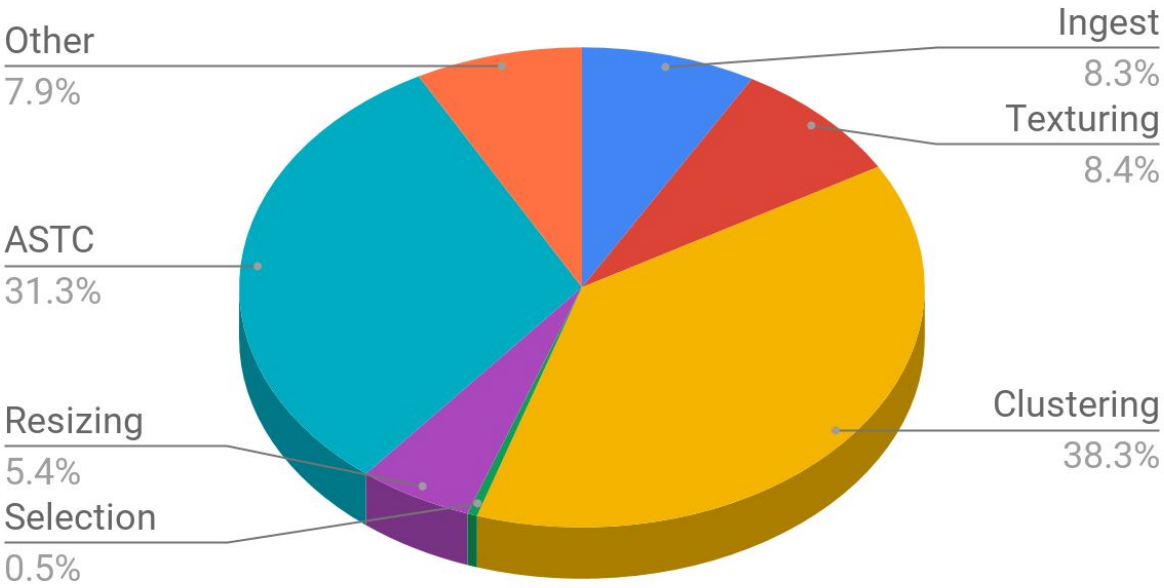
Results

Results




- Input scenes
  - 600k - 3B input triangles
- Output
  - <72k triangles
  - 16.8M texels
  - 4.0 - 5.6x peak overdraw
  - 13 - 17 MB disk size
- 5.1 - 6.7 ms rendering time

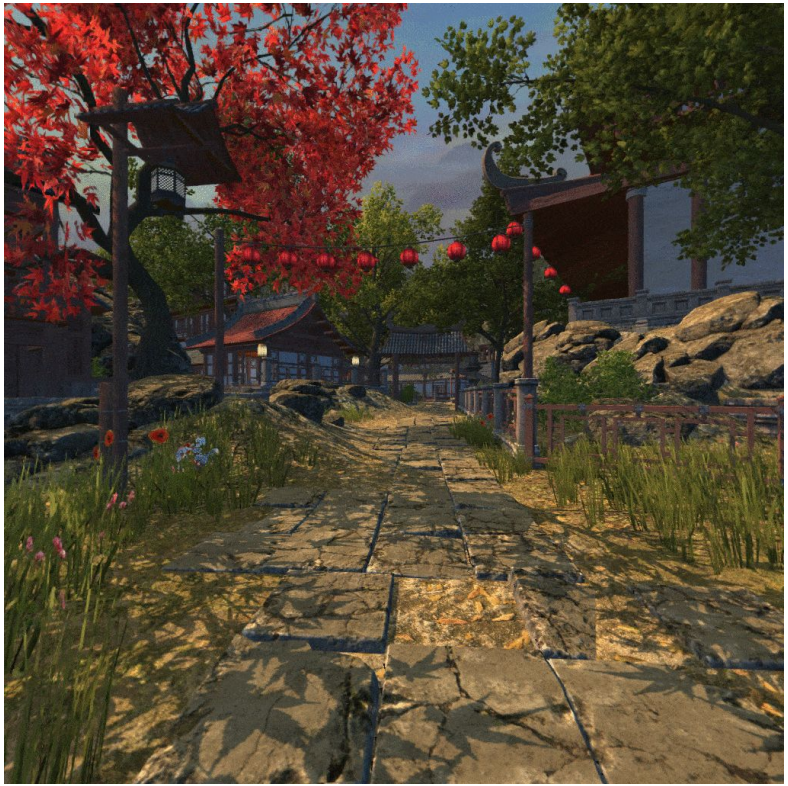
Processing Time



Results

Results

Results



Conclusion

- Practical system for pre-rendered 6DoF VR content
- Layered tile representation
- Constrained optimization method for rendering in a fixed budget
- Space carving for texturing from RGBD samples

Source Code Available:

https://github.com/googlevr/seurat

Questions?