Adaptive Ray-bundle Tracing with Memory Usage Prediction:
Efficient Global Illumination in Large Scenes

Yusuke Tokuyoshi (Square Enix Co., Ltd.)
Takashi Sekine (Square Enix Co., Ltd.)
Tiago da Silva (Square Enix Co., Ltd., Univ. of Tokyo)
Takashi Kanai (Univ. of Tokyo)

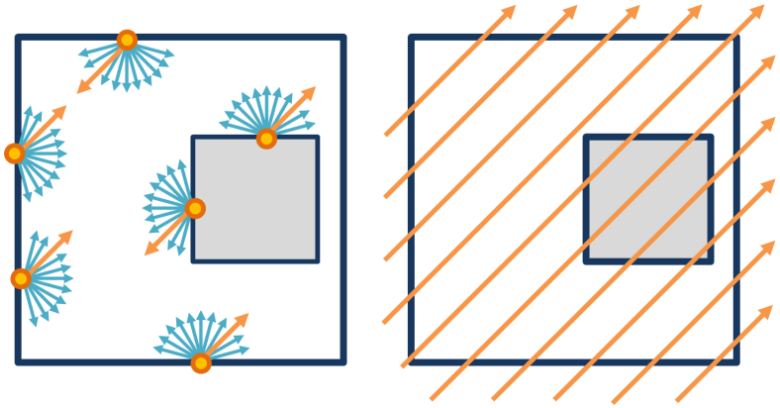
Light Maps for Large Scenes

45.3 M texel light maps
Scene with 4.9 km in diameter (3.7 M triangles)
Computation time: 1396 secs (2000 sample directions)
(GPU: NVIDIA GeForce GTX 580 1.5GB memory)

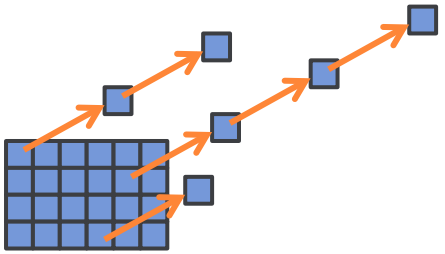
Introduction

1. Introduction
2. Adaptive Tiling for Ray-bundles
3. Experimental Results & Future Work

Ray-bundle Tracing

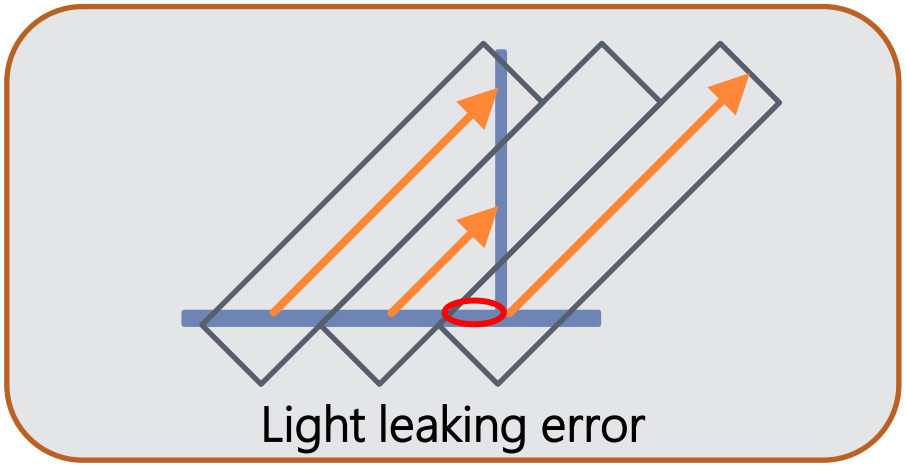


- Set of parallel rays for a sample direction [Sbert96]
- Implemented with GPU rasterization [Szirmay-Kalos98, Hachisuka05]
  - Benefits: HW acceleration, tessellation etc.
- Multi-fragment problem is identical to OIT
  - Per-pixel linked-list [Yang10]



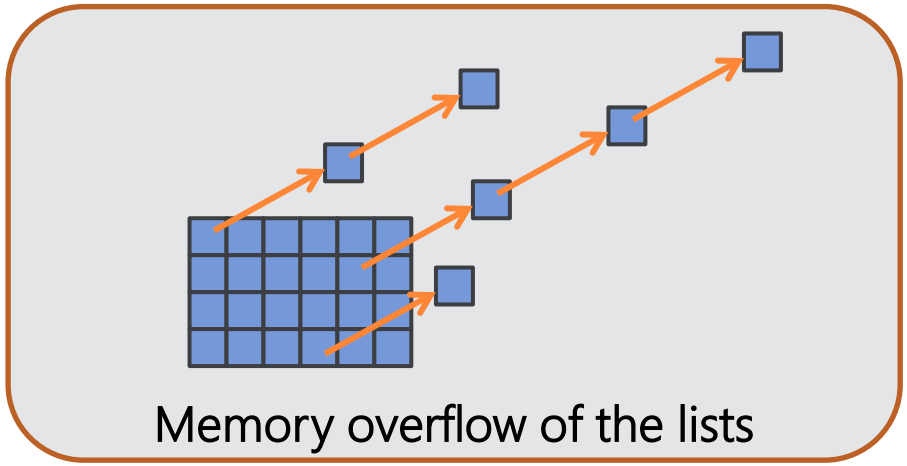
Limited Memory Capacity of GPUs

Ray-bundle tracing is weak in large scenes



Light leaking error

- Uniformly distributed rays
- Inhomogeneous light map density
- High-resolution ray-bundle buffer is required

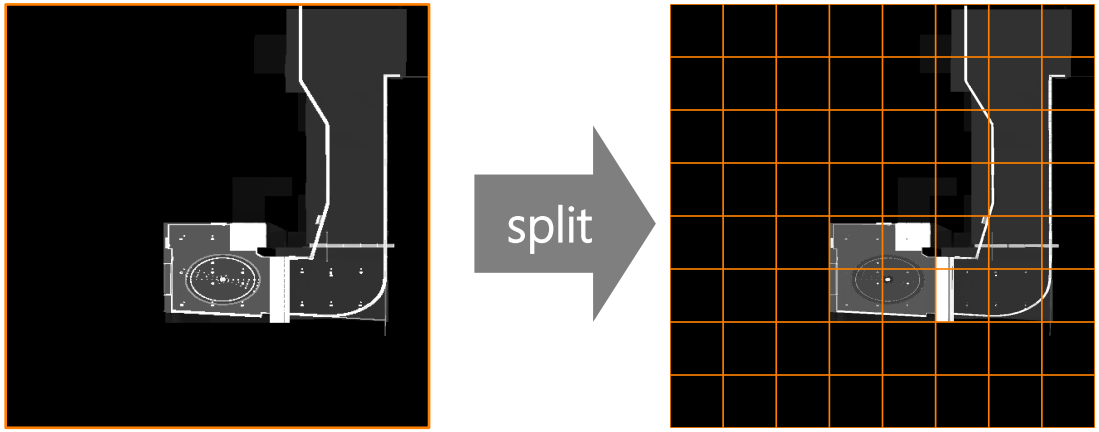


Memory overflow of the lists

- Memory usage is unknown before rendering
- Excessive memory has to be allocated

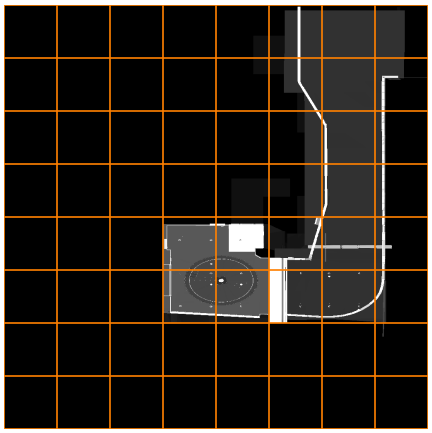
Uniform Tiling [Thibieroz11]

- Proposed for real-time linked-list OIT
  - Split a render target into smaller tiled regions
  - Each tile is rendered separately
- Unsuitable for off-line rendering
  - Overflow is still unpredictable
  - Scene-dependent parameter tuning

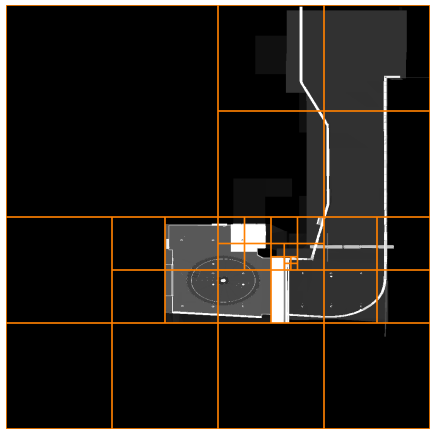


Our Contributions

- Memory usage prediction for linked-list ray-bundles
- Adaptive tile subdivision using the above prediction
  - Reduce the risk of memory overflow & light leaking error
  - Avoid over-splitting
  - Less parameter tuning



Uniform tiling



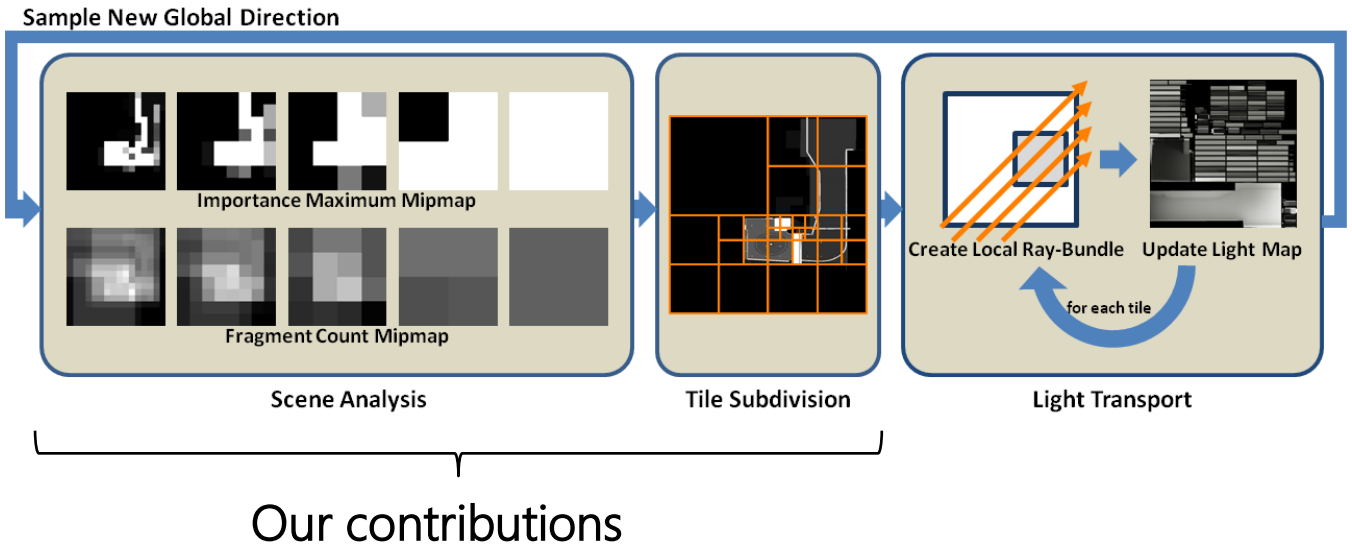
Our adaptive tiling

Adaptive Tiling for Ray-bundles

1. Introduction
2. Adaptive Tiling for Ray-bundles
3. Experimental Results & Future Work

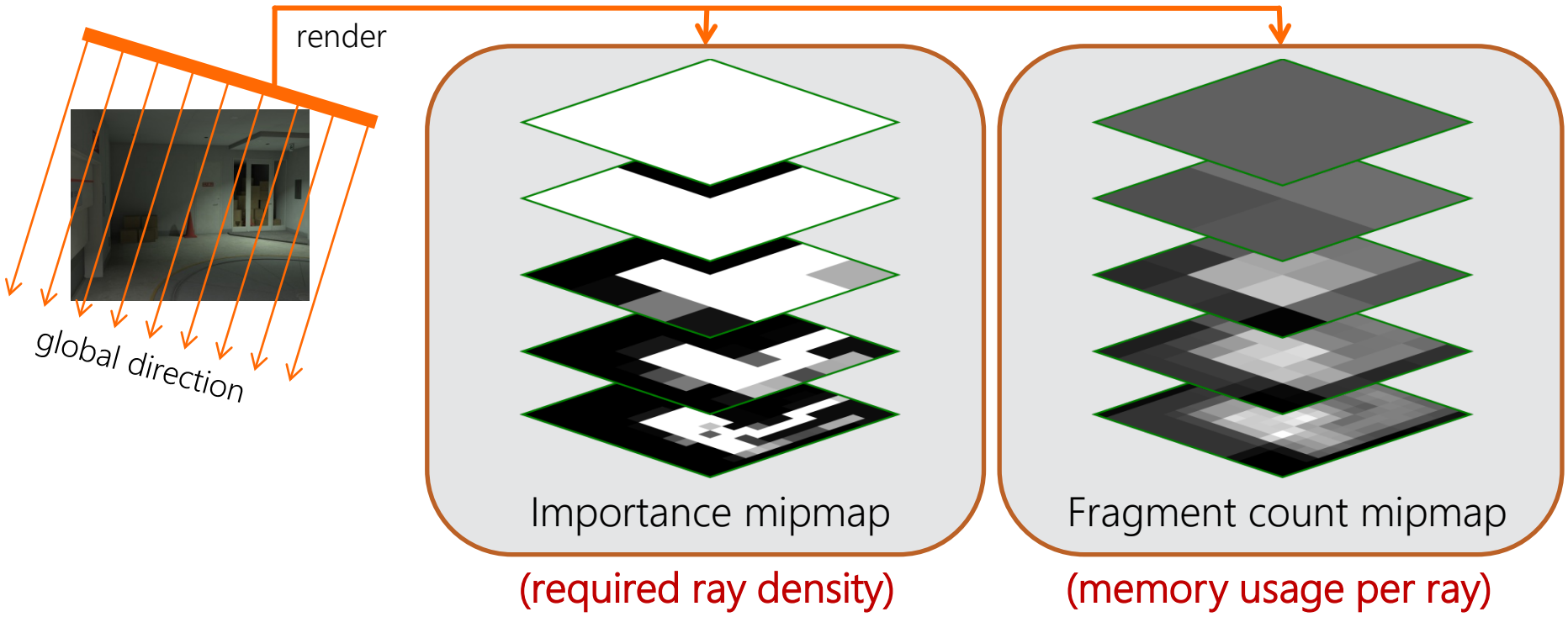
Adaptive Tiling

- Based on adaptive shadow mapping [Fernando01]
  - Quadtree-based tile subdivision
  - According to a low-resolution scene analysis
- Analysis for memory usage prediction is also added
  - The overflow risk is reduced dramatically
    - It is not completely eliminated, however



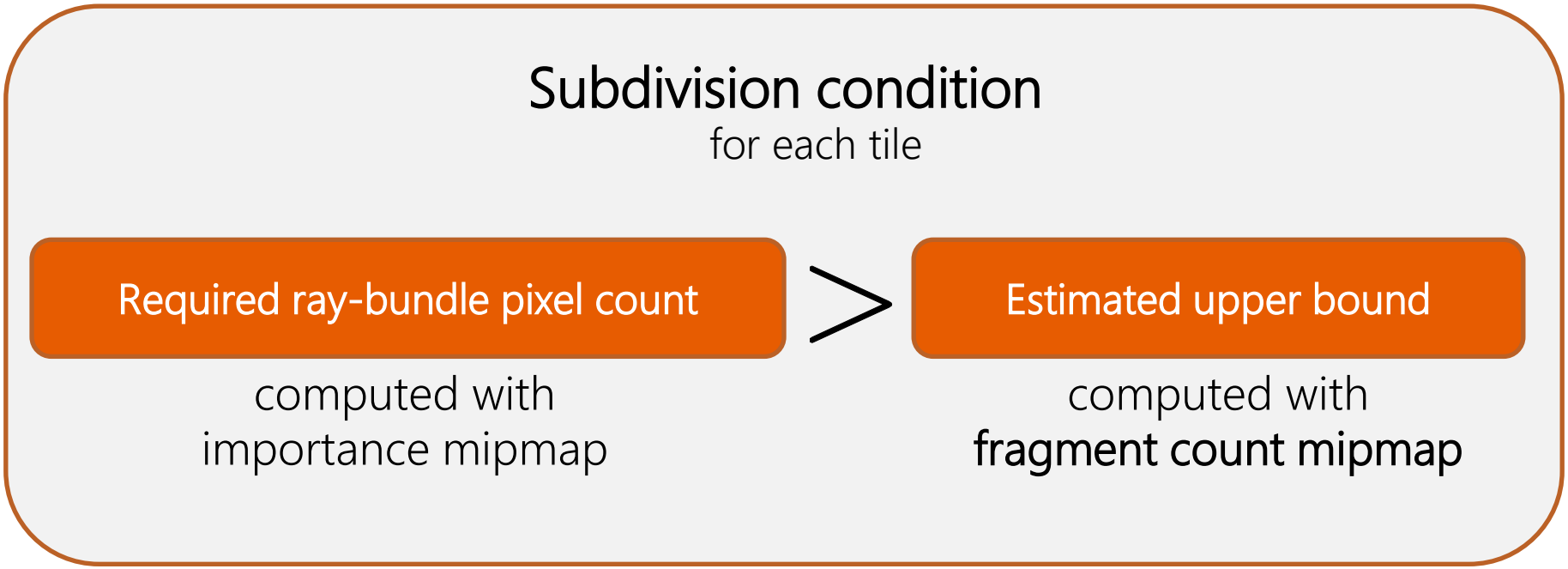
Importance & Fragment Count Analysis

- Render two mipmaps from the ray-bundle direction
- Pixels as quadtree nodes (resolution: 2^n)



Recursive Tile Subdivision

- Start from the top mip level (root of the quadtree)
- A tile is subdivided when overflow is predicted

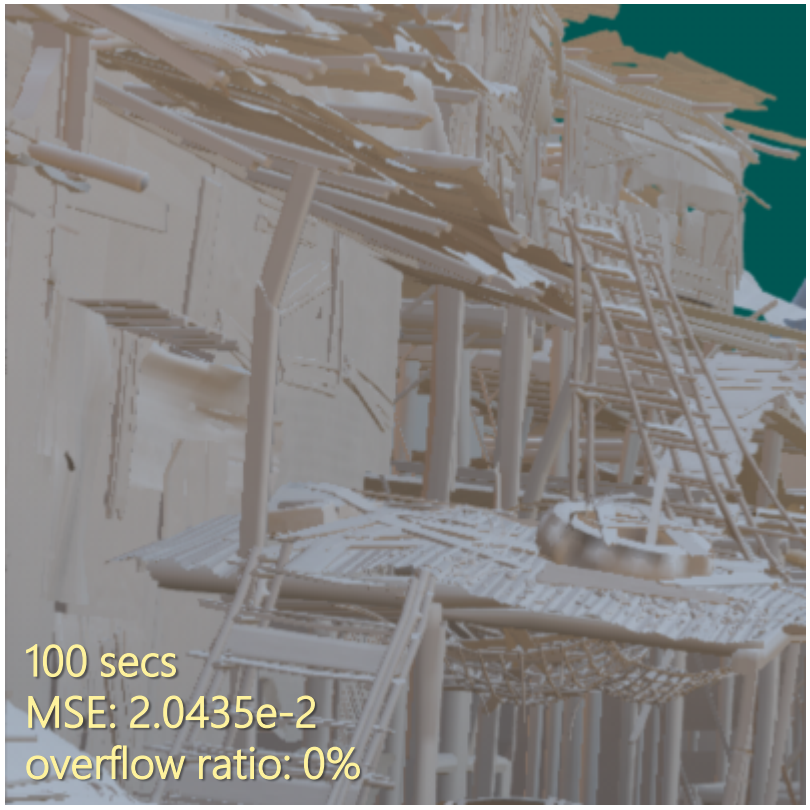


Experimental Results & Conclusions

1. Introduction
2. Adaptive Tiling for Ray-bundles
3. Experimental Results & Future Work

No Tiling

2000 sample directions
Ray-bundle resolution: 1024^2
Node buffer size: 5M nodes
Analysis resolution: 1024^2



GPU: NVIDIA GeForce GTX 580 with 1.5GB memory

35x35 Uniform Tiling

2000 sample directions
Ray-bundle resolution: 1024^2
Node buffer size: 5M nodes
Analysis resolution: 1024^2

GPU: NVIDIA GeForce GTX 580 with 1.5GB memory

Our Adaptive Tiling

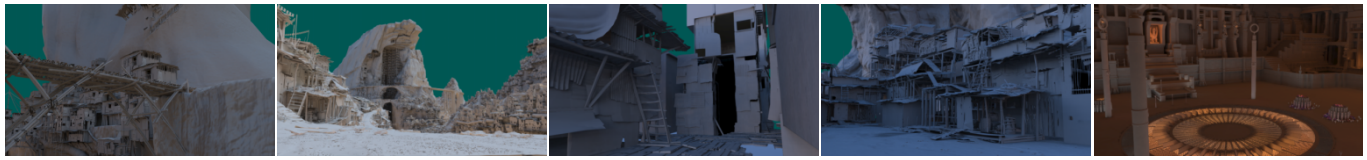
(172.7 tiles / direction)

2000 sample directions
Ray-bundle resolution: 1024^2
Node buffer size: 5M nodes
Analysis resolution: 1024^2

GPU: NVIDIA GeForce GTX 580 with 1.5GB memory

Computation Times per Sample Direction

2% overhead



Analysis Rendering	7.9	13.3	2.4	7.5	12.5
Mipmapping	0.3	0.5	0.3	0.2	0.3
Tile Subdivision	0.3	0.2	0.3	0.4	0.4
GPU-CPU Data Copy	0.7	0.8	0.6	0.8	0.7
Ray-bundle Creation	291.6	405.7	69.9	269.8	418.9
Light Map Update	180.3	274.7	63.8	217.4	286.9

(ms)

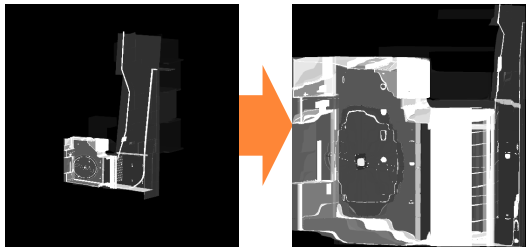
GPU: NVIDIA GeForce GTX 580 with 1.5GB memory

Conclusions

- Adaptive tiling for linked-list ray-bundles
  - A tiles is subdivided when overflow is predicted
  - The risk of memory overflow is reduced dramatically
  - Less parameter tuning
- Memory usage prediction
  - Using the fragment count mipmap
- Demonstrated baking light maps of large scenes
  - With a limited memory capacity

Future Work

- Improving the analysis accuracy
  - Supersampling
  - Conservative rasterization [Hasselgren05]
- Ray-bundle warping
  - Rectilinear texture warping [Rosen12]
- Real-time linked-list OIT
  - For an arbitrary node buffer size



Warping

Thank You