PERFORMANCE COMPARISON OF BOUNDING VOLUME HIERARCHIES FOR GPU RAY TRACING

Daniel Meister[1] and Jiří Bittner[2]

Advanced Micro Devices, Inc.[1]

Czech Technical University in Prague[2]

MOTIVATION AND CONTRIBUTION

Motivation

- A lot of research on ray tracing done in the last decades
- Typically comparing against one or two reference methods
  - *“The methods are orthogonal …”*
- Experts in the field cannot say which method is the best

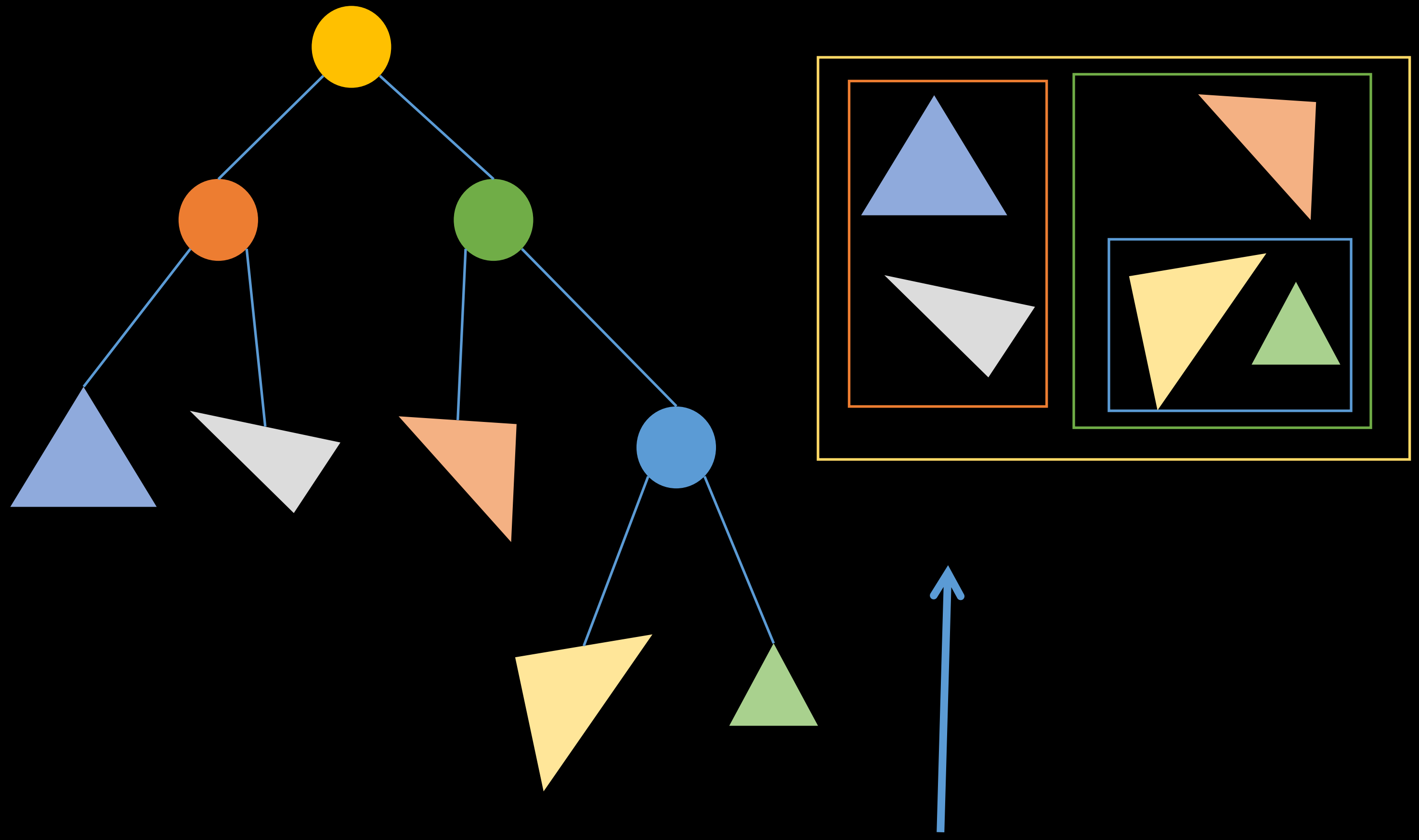
Contribution

- Comparison most popular methods in a unified framework
- Simulated annealing for insertion-based BVH optimization

BOUNDING VOLUME HIERARCHY (BVH)

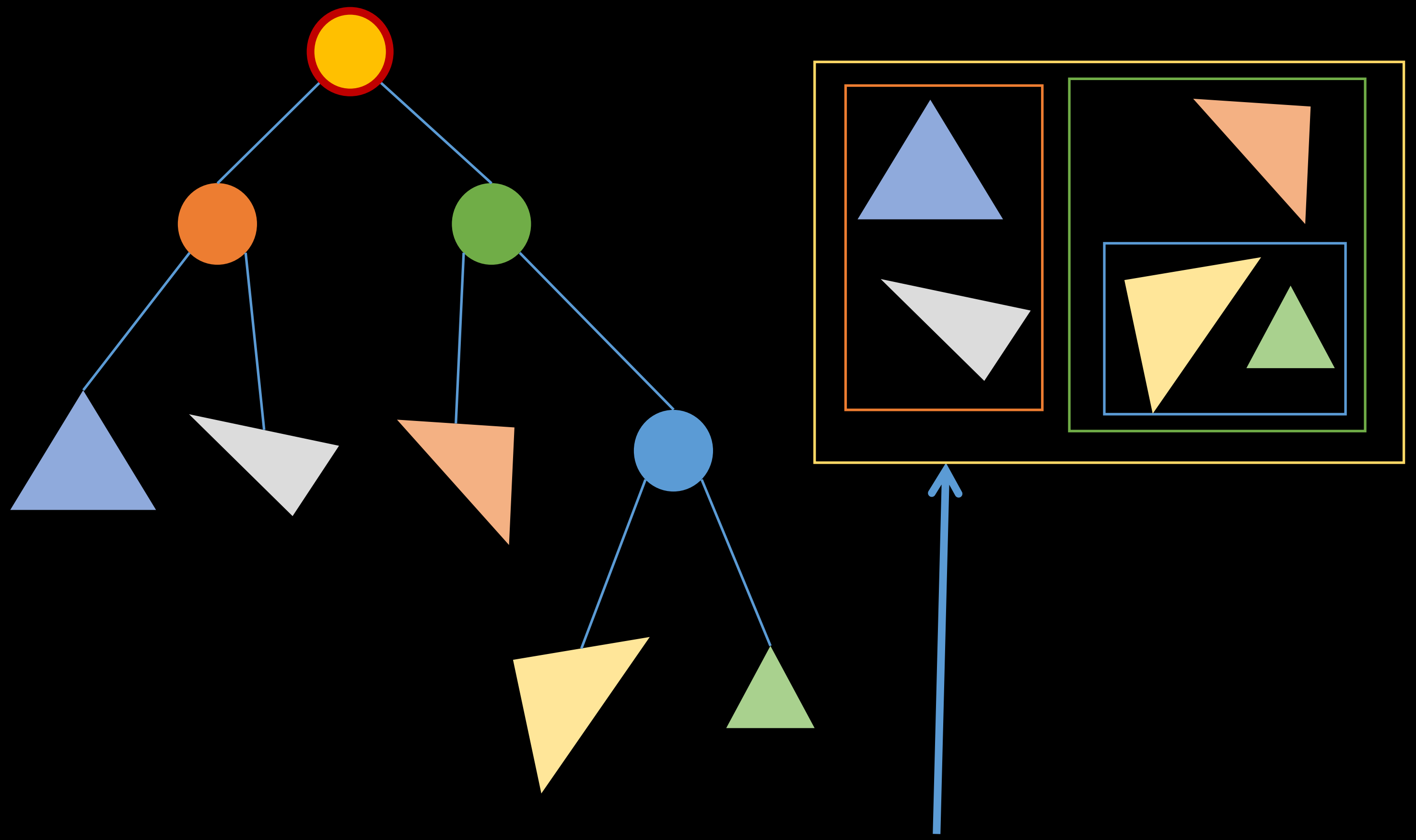
- Hierarchical data structure of bounding boxes
  - Geometric primitives in leaves
  - Bounding boxes in interior nodes

- Ray Traversal (finding the intersection using BVH)
  - If ray hits the box, go one level bellow and test child boxes
  - Otherwise, skip the whole subtree



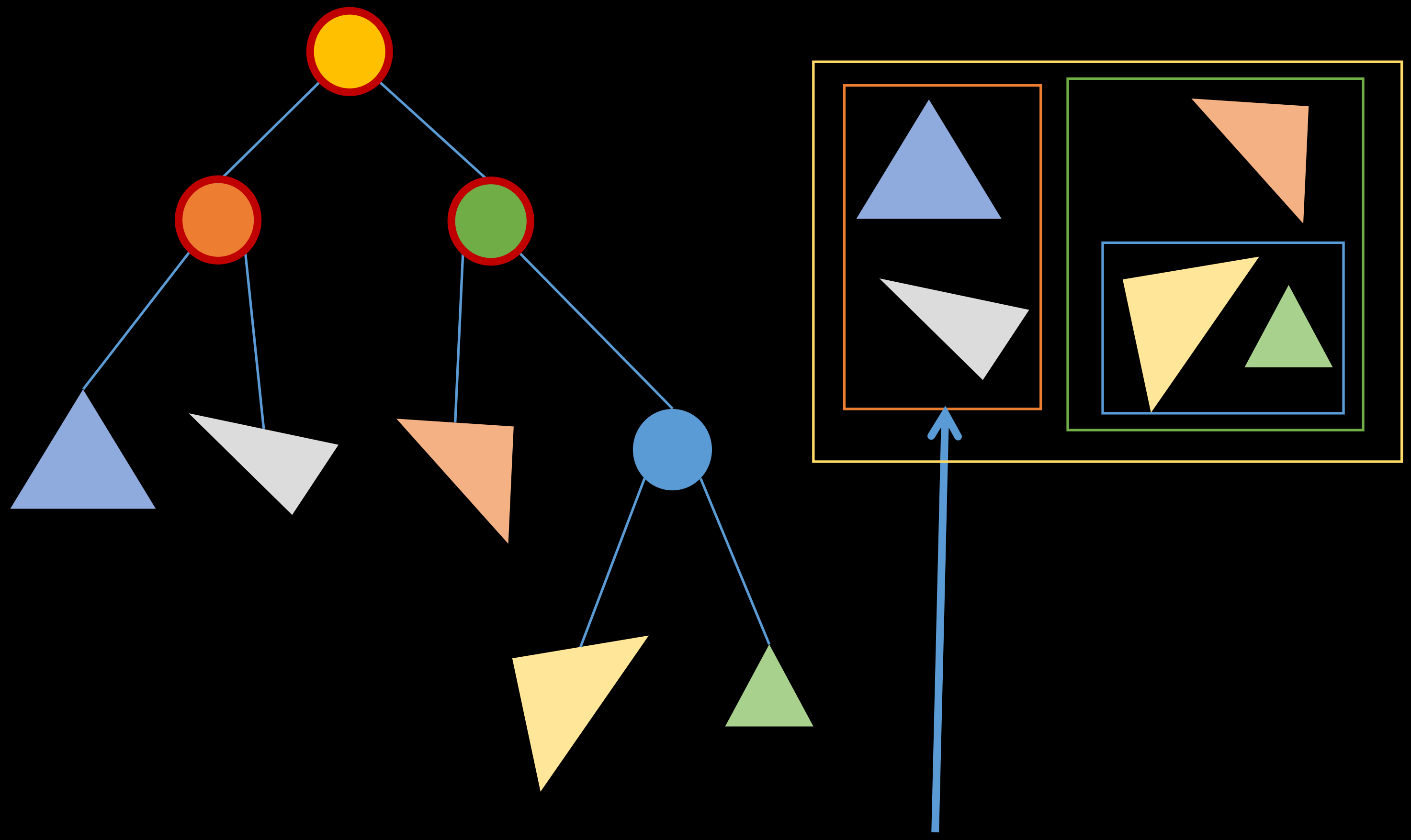
BOUNDING VOLUME HIERARCHY (BVH)

- Hierarchical data structure of bounding boxes
  - Geometric primitives in leaves
  - Bounding boxes in interior nodes
- Ray Traversal (finding the intersection using BVH)
  - If ray hits the box, go one level bellow and test child boxes
  - Otherwise, skip the whole subtree



BOUNDING VOLUME HIERARCHY (BVH)

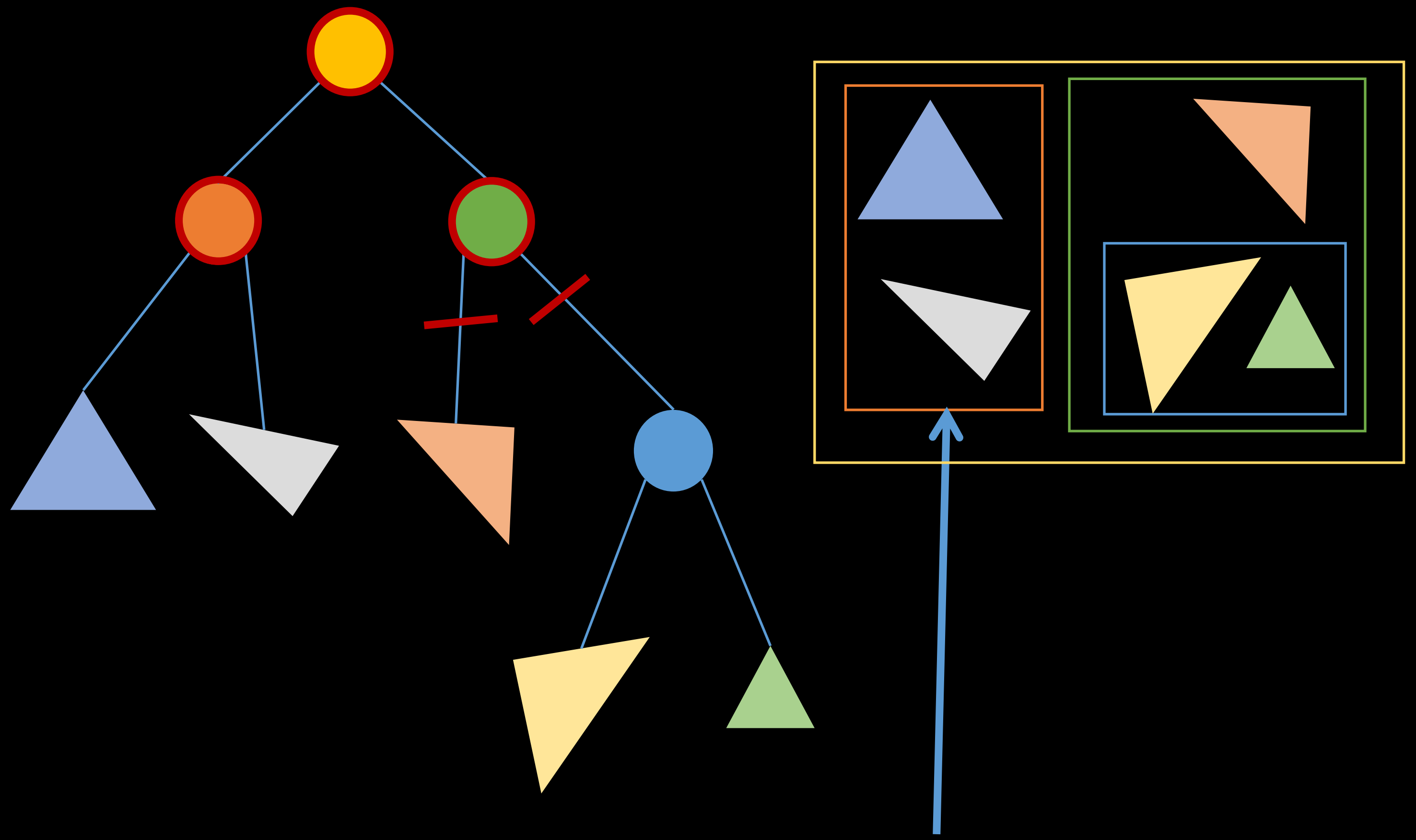
- Hierarchical data structure of bounding boxes
  - Geometric primitives in leaves
  - Bounding boxes in interior nodes
- Ray Traversal (finding the intersection using BVH)
  - If ray hits the box, go one level bellow and test child boxes
  - Otherwise, skip the whole subtree



BOUNDING VOLUME HIERARCHY (BVH)

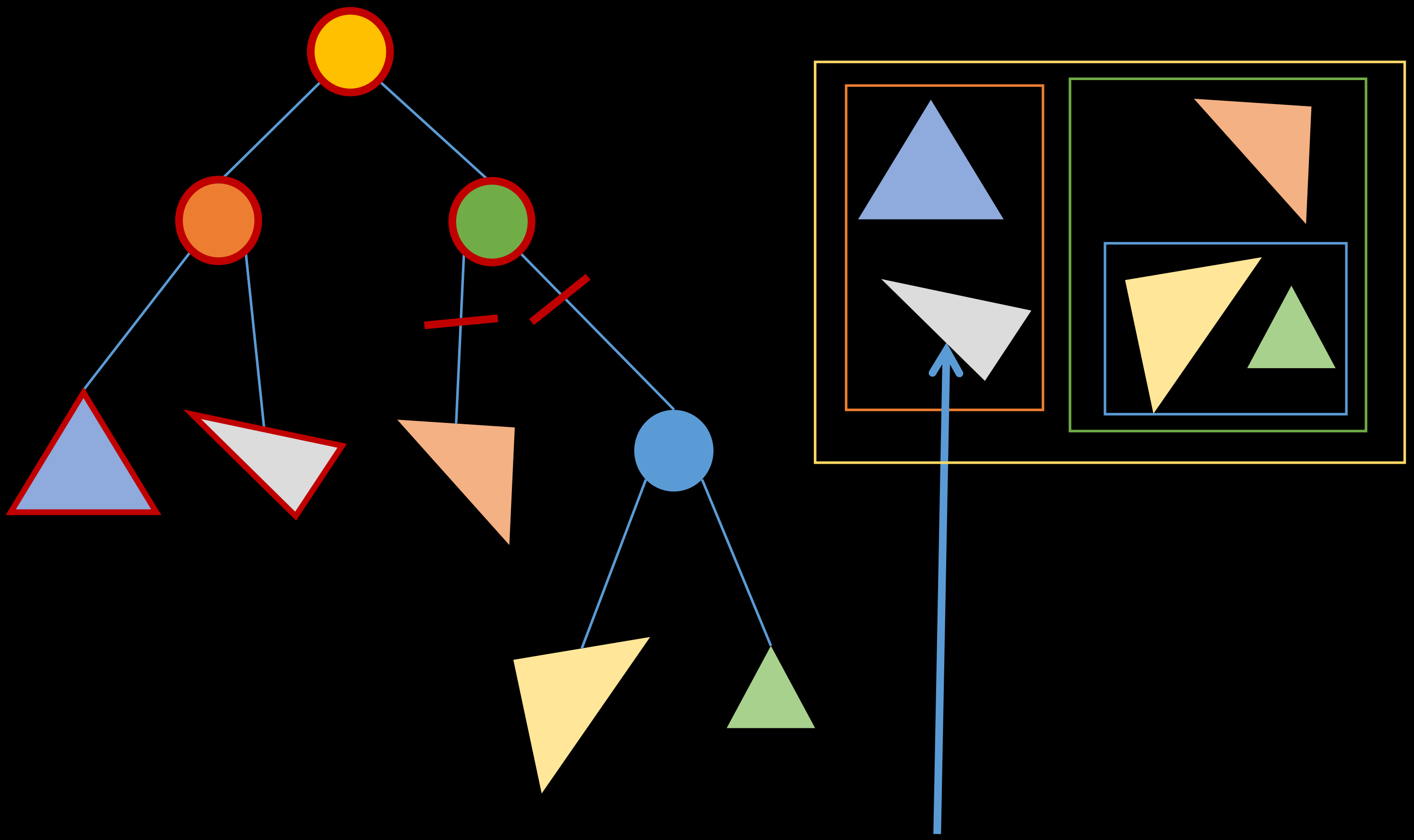
- Hierarchical data structure of bounding boxes
  - Geometric primitives in leaves
  - Bounding boxes in interior nodes

- Ray Traversal (finding the intersection using BVH)
  - If ray hits the box, go one level bellow and test child boxes
  - Otherwise, skip the whole subtree



BOUNDING VOLUME HIERARCHY (BVH)

- Hierarchical data structure of bounding boxes
  - Geometric primitives in leaves
  - Bounding boxes in interior nodes
- Ray Traversal (finding the intersection using BVH)
  - If ray hits the box, go one level bellow and test child boxes
  - Otherwise, skip the whole subtree



SURFACE AREA HEURISTIC (SAH)

BVH construction is a difficult problem

- Many possible BVHs for a given input geometry
- We can express the quality of a BVH via a cost model
- Trade-off between the construction time and the BVH quality

Cost of a child

Traversal cost constant

$$c(N) = \begin{cases} c_T + \sum_{N_c} \frac{SA(N_c)}{SA(N)} c(N_c) & \text{if } N \text{ is interior node} \\ c_I |N| & \text{otherwise} \end{cases}$$

Geometric prim. count

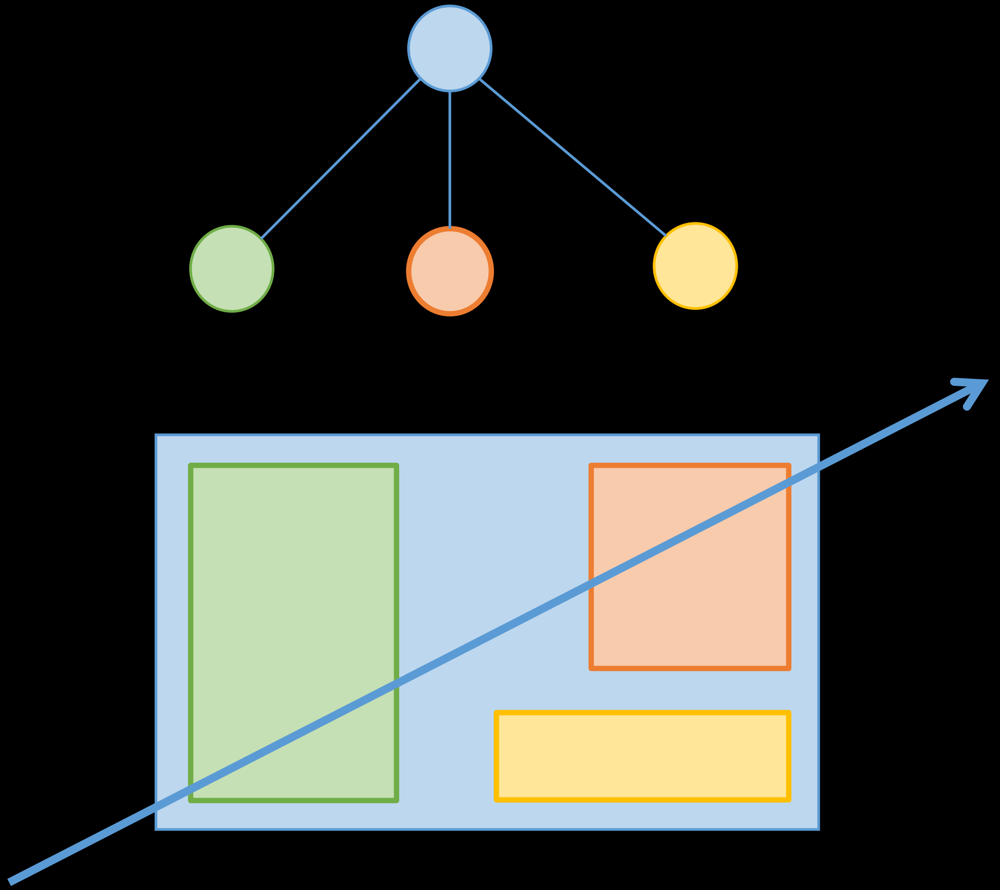
Intersection cost constant

$$c(N) = \frac{1}{SA(N)} \left[c_T \sum_{N_i} SA(N_i) + c_I \sum_{N_l} SA(N_l)|N_l| \right]$$

Surface area of the root

Cost of interior nodes

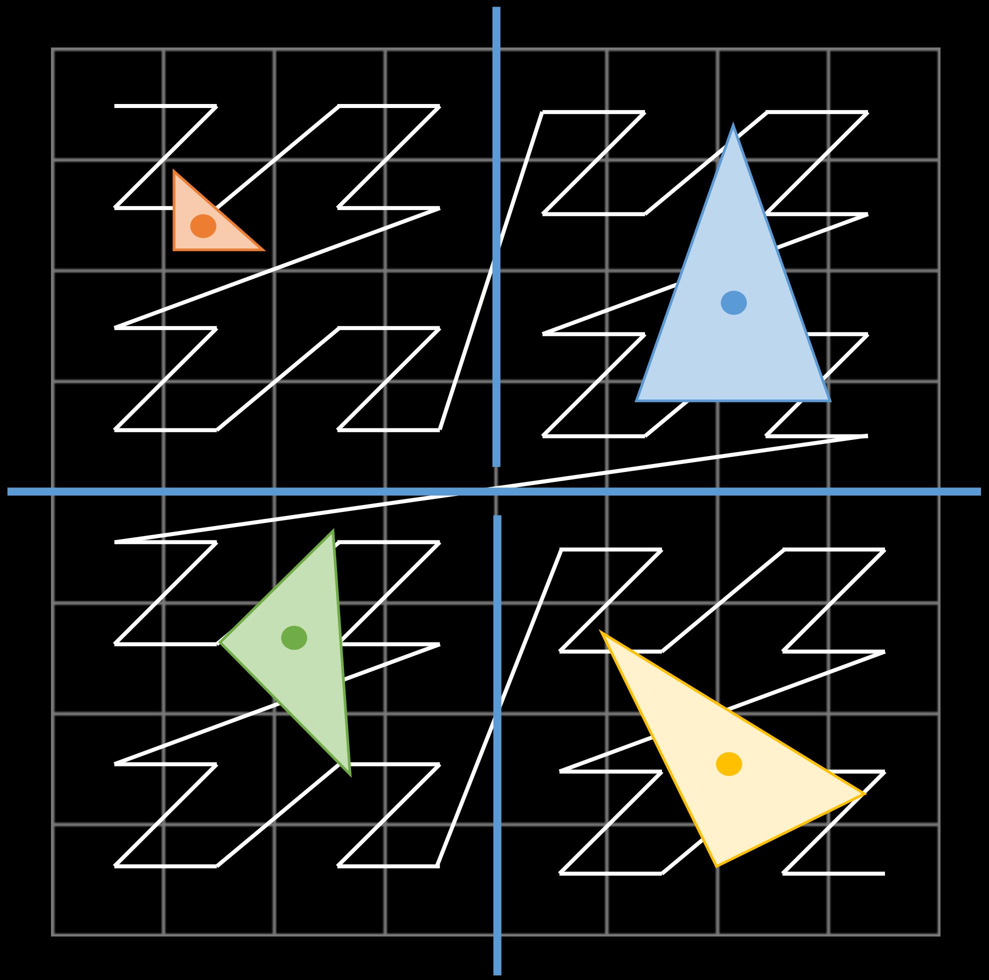
Cost of leaf nodes



[MacDonald and Booth 1990]

FAST BVH CONSTRUCTION

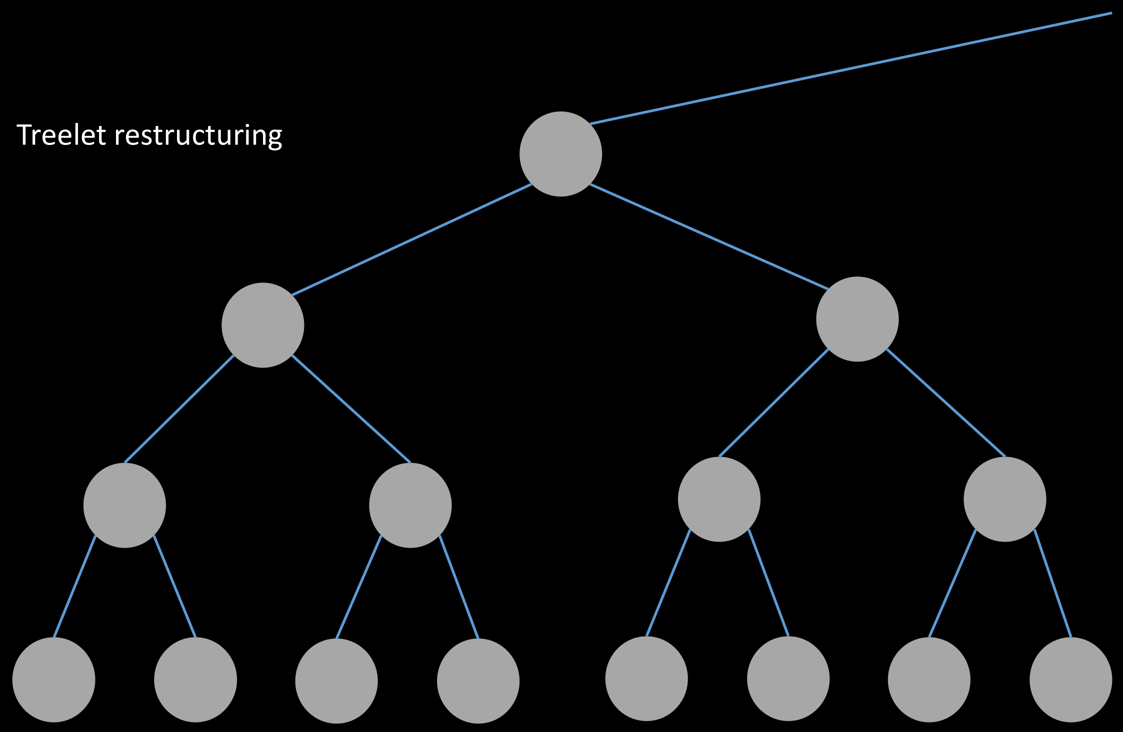
- LBVH [Karras 2012]
  - Very fast algorithm but lower BVH quality
  - Sorting geometric primitives along a space-filling curve such as Morton curve
  - Morton curve encodes an implicit BVH constructed by spatial median splits
- HLBVH [Garanzha et. al 2011]
  - Using more significant bits of Morton codes as bin indices for SAH splits in top levels to improve the quality
  - Bottom levels constructed in the same manner as LBVH



BALANCED BVH CONSTRUCTION

- PLOC [Meister and Bittner 2018a]
  - Parallel locally-ordered clustering
  - Using Morton curve to find nearest neighbors
  - Good quality and very fast
  - Optimized version PLOC++ [Benthin et al. 2022]

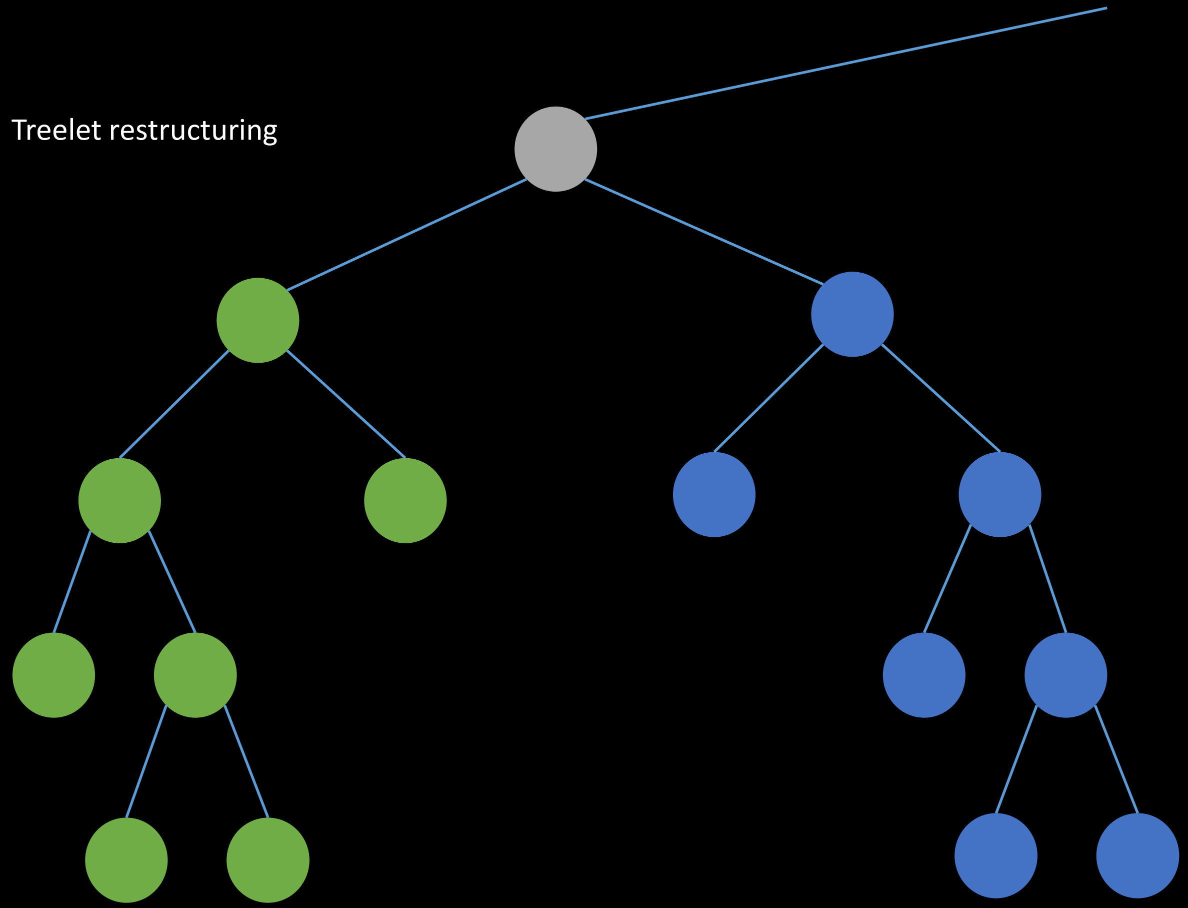
- ATRBVH [Domingues and Pedrini 2015]
  - Treelet restructuring via agglomerative clustering
  - Optimizes an existing BVH (typically LBVH)
  - Processing treelets in a bottom-up fashion
  - Very good quality but slower than PLOC



BALANCED BVH CONSTRUCTION

- PLOC [Meister and Bittner 2018a]
  - Parallel locally-ordered clustering
  - Using Morton curve to find nearest neighbors
  - Good quality and very fast
  - Optimized version PLOC++ [Benthin et al. 2022]

- ATRBVH [Domingues and Pedrini 2015]
  - Treelet restructuring via agglomerative clustering
  - Optimizes an existing BVH (typically LBVH)
  - Processing treelets in a bottom-up fashion
  - Very good quality but slower than PLOC

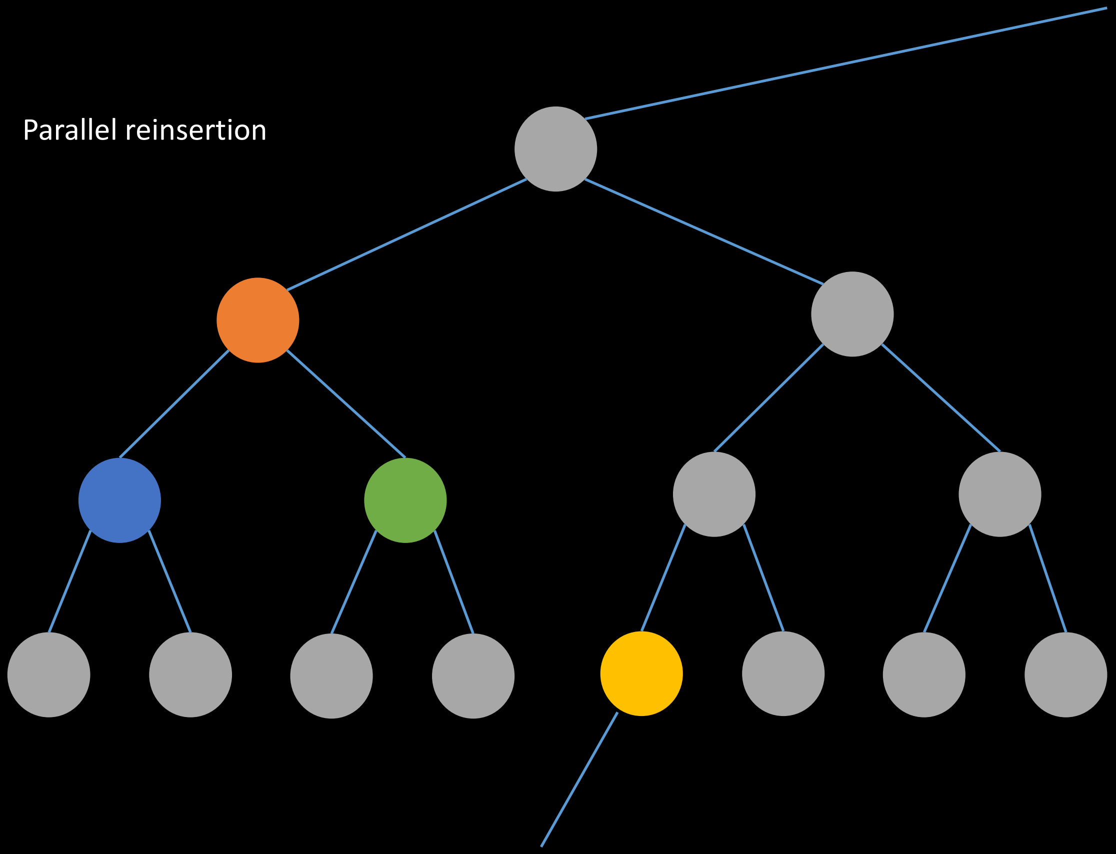


HIGH-QUALITY BVH CONSTRUCTION

- PRBVH [Meister and Bittner 2018b]
  - Parallel insertion-based optimization
  - Removing and inserting subtrees to new positions
  - Systematically minimizes the BVH cost
  - Very high-quality BVHs

- SBVH [Stich et. al 2009]
  - Top-down construction using spatial splitting
  - Robust to diagonal and oblong primitives
  - Slow (no efficient GPU implementation)
  - Very high-quality

- Collapse BVH2 to BVH{4|8} [Ylitie et al. 2017]
  - Optimal algorithm minimizing the BVH cost
  - Dynamic programming



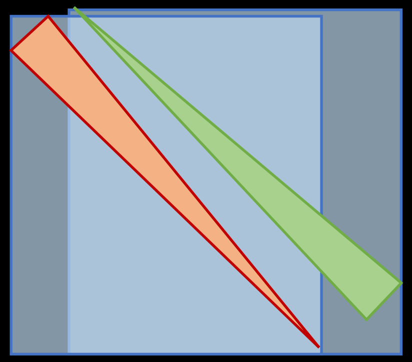
HIGH-QUALITY BVH CONSTRUCTION

- PRBVH [Meister and Bittner 2018b]
  - Parallel insertion-based optimization
  - Removing and inserting subtrees to new positions
  - Systematically minimizes the BVH cost
  - Very high-quality BVHs

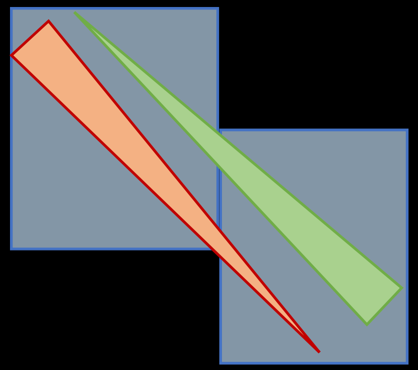
- SBVH [Stich et. al 2009]
  - Top-down construction using spatial splitting
  - Robust to diagonal and oblong primitives
  - Slow (no efficient GPU implementation)
  - Very high-quality

- Collapse BVH2 to BVH{4|8} [Ylitie et al. 2017]
  - Optimal algorithm minimizing the BVH cost
  - Dynamic programming

Standard BVH



BVH with spatial splits



SIMULATED ANNEALING FOR INSERTION-BASED BVH OPTIMIZATION

- Idea: Accept also positions that may increase the cost to avoid getting stuck in local minima
- Acceptance probability given by Boltzmann factor

$$P(\Delta d, T) = \begin{cases} \min(e^{-\Delta d/T}, 1) & T > 0 \\ 0 & T = 0 \end{cases}$$

Surface area difference

- Temperature

$$T(i) = \max\left(0, -\sin\left(\frac{2\pi i}{f}\right)\right) T_{max} \frac{I-i}{I}$$

Max. temperature

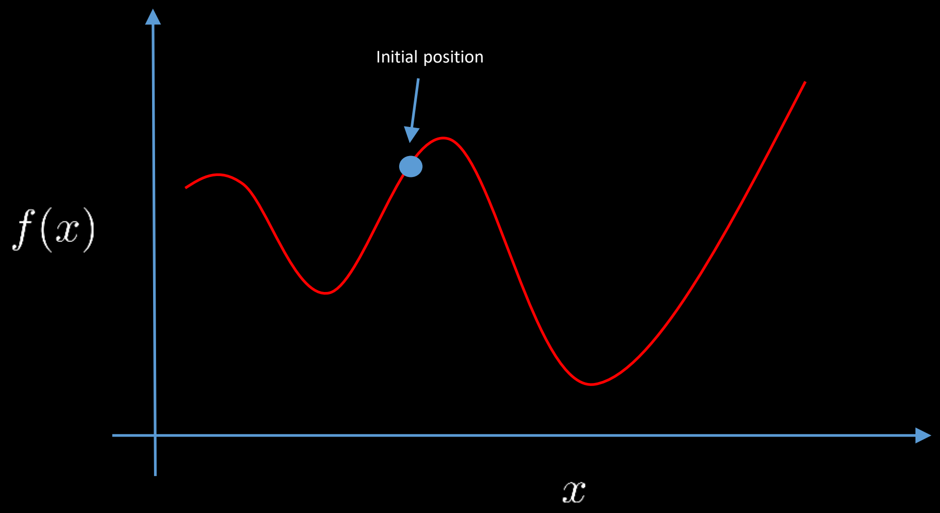
Max. iterations

Current iteration

Frequency

- Two issues specific to insertion-based optimization:
  - Search space is huge → stochastic pruning
  - Parallel processing → conflict resolution

Hill climbing



SIMULATED ANNEALING FOR INSERTION-BASED BVH OPTIMIZATION

- Idea: Accept also positions that may increase the cost to avoid getting stuck in local minima
- Acceptance probability given by Boltzmann factor

$$P(\Delta d, T) = \begin{cases} \min(e^{-\Delta d/T}, 1) & T > 0 \\ 0 & T = 0 \end{cases}$$

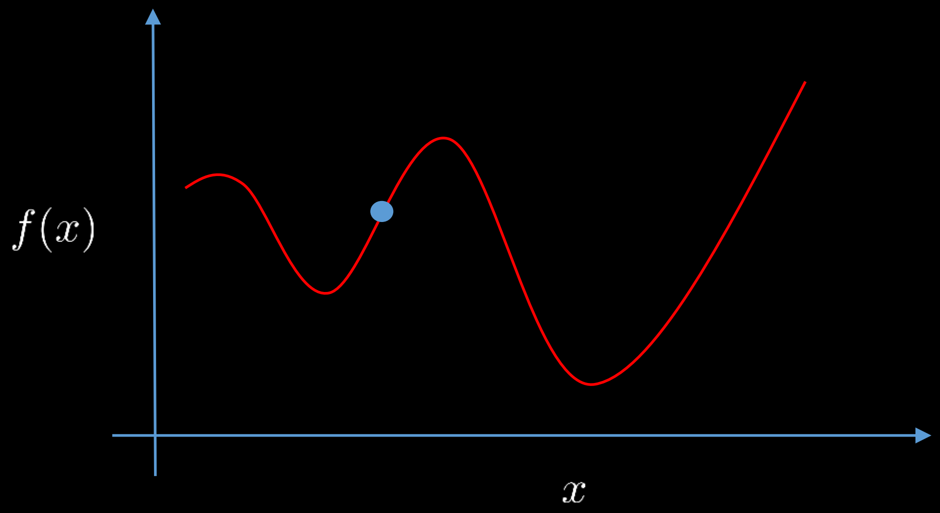
Surface area difference

- Temperature

$$T(i) = \max\left(0, -\sin\left(\frac{2\pi i}{f}\right)\right) T_{max} \frac{I-i}{I}$$

Max. temperature
Max. iterations
Current iteration
Frequency

Hill climbing



- Two issues specific to insertion-based optimization:
  - Search space is huge → stochastic pruning
  - Parallel processing → conflict resolution

SIMULATED ANNEALING FOR INSERTION-BASED BVH OPTIMIZATION

- Idea: Accept also positions that may increase the cost to avoid getting stuck in local minima
- Acceptance probability given by Boltzmann factor

$$P(\Delta d, T) = \begin{cases} \min(e^{-\Delta d/T}, 1) & T > 0 \\ 0 & T = 0 \end{cases}$$

Surface area difference

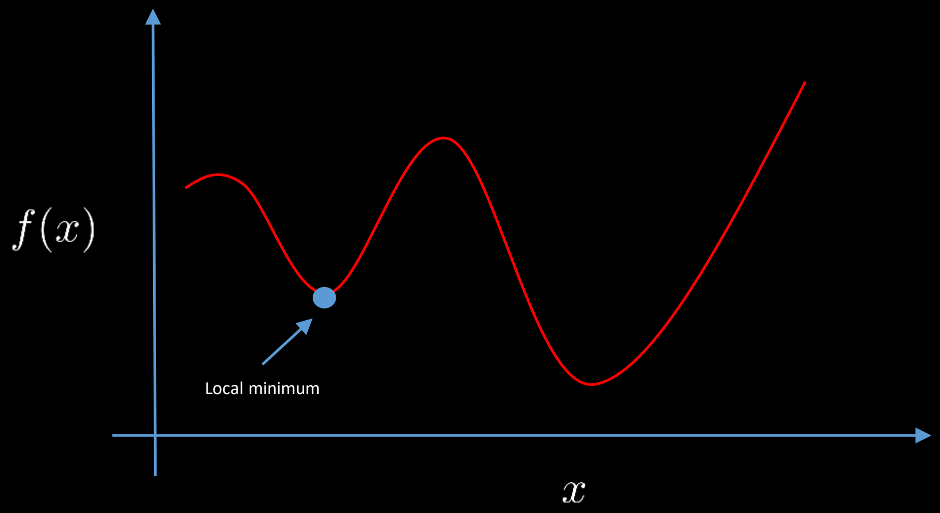
- Temperature

$$T(i) = \max\left(0, -\sin\left(\frac{2\pi i}{f}\right)\right) T_{max} \frac{I-i}{I}$$

Max. temperature; Max. iterations; Current iteration; Frequency

- Two issues specific to insertion-based optimization:
  - Search space is huge → stochastic pruning
  - Parallel processing → conflict resolution

Hill climbing



SIMULATED ANNEALING FOR INSERTION-BASED BVH OPTIMIZATION

- Idea: Accept also positions that may increase the cost to avoid getting stuck in local minima
- Acceptance probability given by Boltzmann factor

$$P(\Delta d, T) = \begin{cases} \min(e^{-\Delta d/T}, 1) & T > 0 \\ 0 & T = 0 \end{cases}$$

Surface area difference

- Temperature

$$T(i) = \max\left(0, -\sin\left(\frac{2\pi i}{f}\right)\right) T_{max} \frac{I-i}{I}$$

Max. temperature

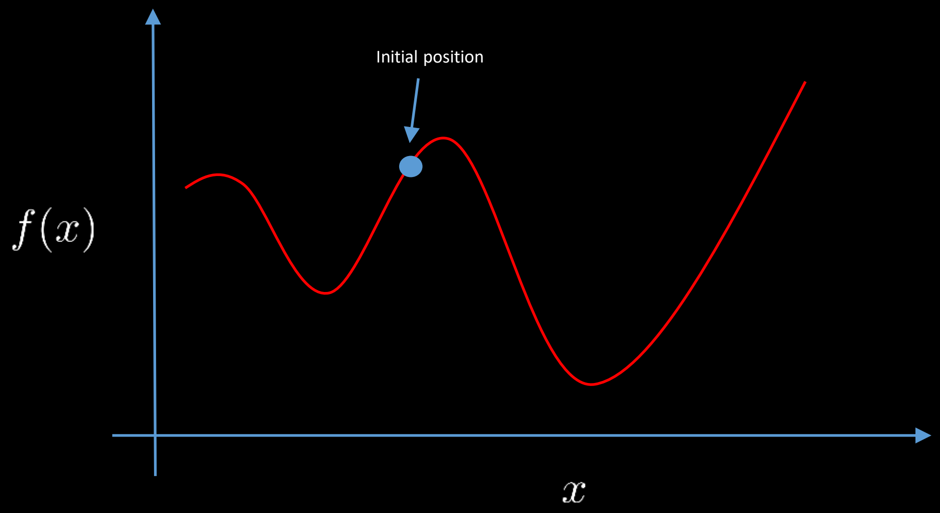
Max. iterations

Current iteration

Frequency

- Two issues specific to insertion-based optimization:
  - Search space is huge → stochastic pruning
  - Parallel processing → conflict resolution

Simulated annealing



SIMULATED ANNEALING FOR INSERTION-BASED BVH OPTIMIZATION

- Idea: Accept also positions that may increase the cost to avoid getting stuck in local minima
- Acceptance probability given by Boltzmann factor

$$P(\Delta d, T) = \begin{cases} \min(e^{-\Delta d/T}, 1) & T > 0 \\ 0 & T = 0 \end{cases}$$

Surface area difference

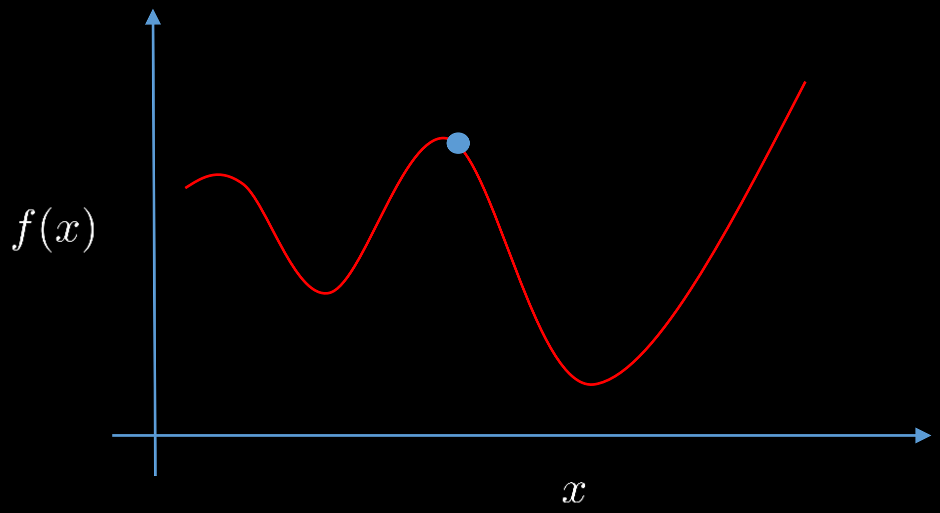
- Temperature

$$T(i) = \max\left(0, -\sin\left(\frac{2\pi i}{f}\right)\right) T_{max} \frac{I-i}{I}$$

Max. temperature (T_{max}), Max. iterations (I), Current iteration (i), Frequency (f)

- Two issues specific to insertion-based optimization:
  - Search space is huge → stochastic pruning
  - Parallel processing → conflict resolution

Simulated annealing



SIMULATED ANNEALING FOR INSERTION-BASED BVH OPTIMIZATION

- Idea: Accept also positions that may increase the cost to avoid getting stuck in local minima
- Acceptance probability given by Boltzmann factor

$$P(\Delta d, T) = \begin{cases} \min(e^{-\Delta d/T}, 1) & T > 0 \\ 0 & T = 0 \end{cases}$$

Surface area difference

- Temperature

Max. temperature

Max. iterations

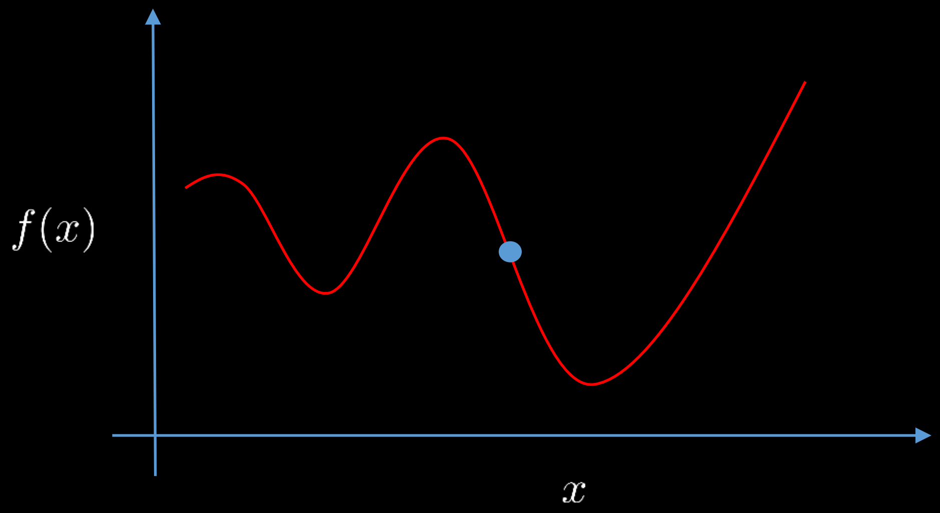
$$T(i) = \max\left(0, -\sin\left(\frac{2\pi i}{f}\right)\right) T_{max} \frac{I-i}{I}$$

Current iteration

Frequency

- Two issues specific to insertion-based optimization:
  - Search space is huge → stochastic pruning
  - Parallel processing → conflict resolution

Simulated annealing



SIMULATED ANNEALING FOR INSERTION-BASED BVH OPTIMIZATION

- Idea: Accept also positions that may increase the cost to avoid getting stuck in local minima
- Acceptance probability given by Boltzmann factor

$$P(\Delta d, T) = \begin{cases} \min(e^{-\Delta d/T}, 1) & T > 0 \\ 0 & T = 0 \end{cases}$$

Surface area difference

- Temperature

$$T(i) = \max\left(0, -\sin\left(\frac{2\pi i}{f}\right)\right) T_{max} \frac{I-i}{I}$$

Max. temperature

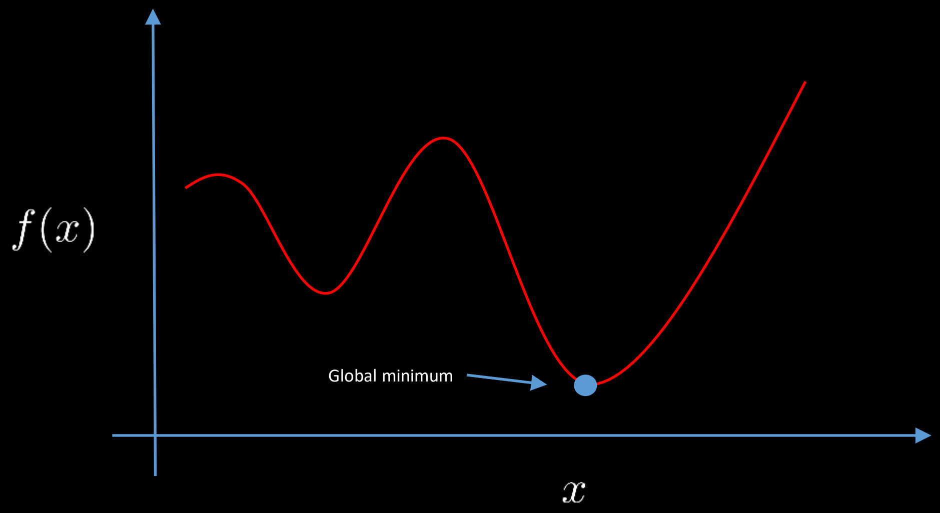
Max. iterations

Current iteration

Frequency

- Two issues specific to insertion-based optimization:
  - Search space is huge → stochastic pruning
  - Parallel processing → conflict resolution

Simulated annealing



SIMULATED ANNEALING FOR INSERTION-BASED BVH OPTIMIZATION

- Idea: Accept also positions that may increase the cost to avoid getting stuck in local minima
- Acceptance probability given by Boltzmann factor

$$P(\Delta d, T) = \begin{cases} \min(e^{-\Delta d/T}, 1) & T > 0 \\ 0 & T = 0 \end{cases}$$

Surface area difference

- Temperature

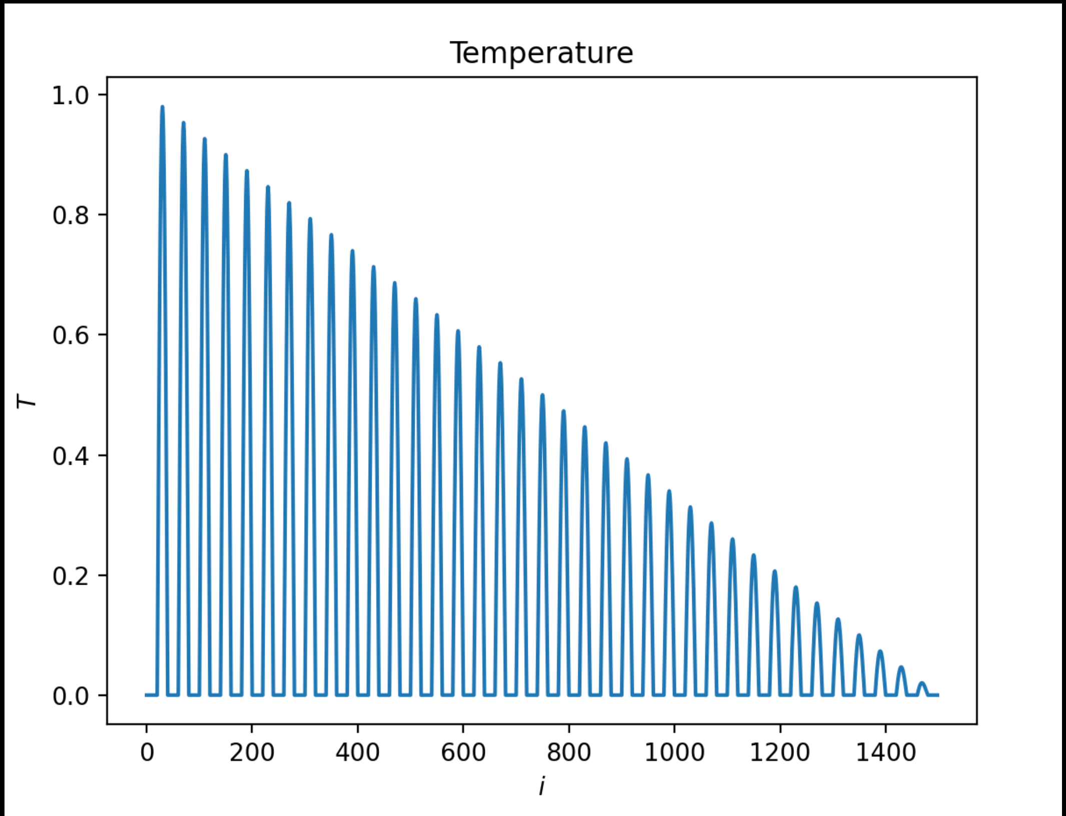
$$T(i) = \max\left(0, -\sin\left(\frac{2\pi i}{f}\right)\right) T_{max} \frac{I-i}{I}$$

Max. temperature

Max. iterations

Current iteration

Frequency



More details in the paper

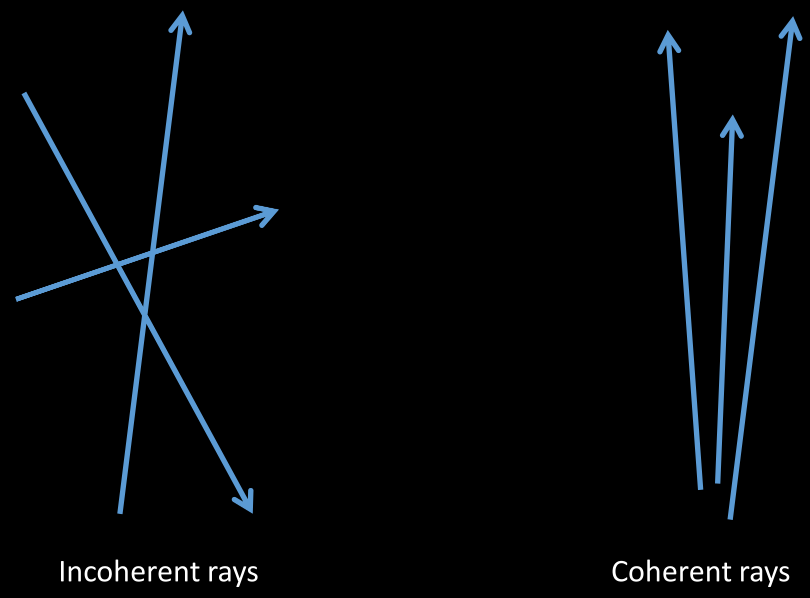
- Two issues specific to insertion-based optimization:
  - Search space is huge → stochastic pruning
  - Parallel processing → conflict resolution

RAY TRAVERSAL

- Ray traversal on GPU [Aila and Laine 2009]
  - *Understanding the Efficiency of Ray Traversal on GPUs*
  - Stack-based algorithm
  - Persistent warps and dynamic fetch

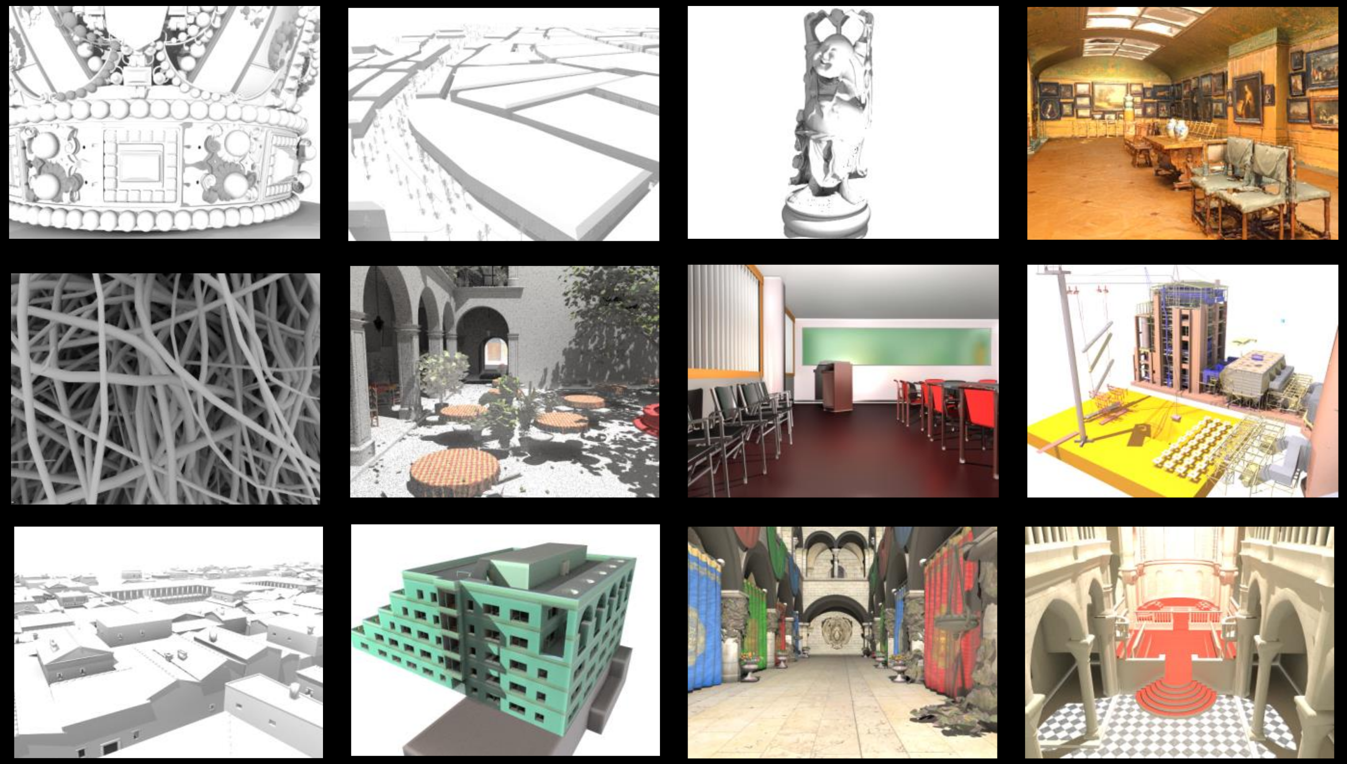
- Wide-BVHs [Lier et al. 2018]
  - *CPU-style SIMD Ray Traversal on GPUs*
  - Node process by *k* lanes for branching factor *k*

- Ray reordering [Meister et al. 2020]
  - Improving ray coherence by grouping similar rays
    - Sorting along Morton curve
    - Encoding ray (origin and direction) is challenging
  - Speedup must outweigh additional overhead



EXPERIMENTAL SETUP

- Aila's framework ported to HIP [Aila and Laine 2009]
  - Publicly available implementations
  - Our own implementations
- 12 scenes (75k – 12759k tris)
- Wavefront path tracing with NEE
  - 32 samples per pixel
  - 2 shadow rays per hit
  - Up to 8 bounces (no Russian roulette)
- Resolution 1024x768
- Three camera views for each scene
- AMD Radeon RX 6800 XT GPU
- Branching factors: 2, 4, 8

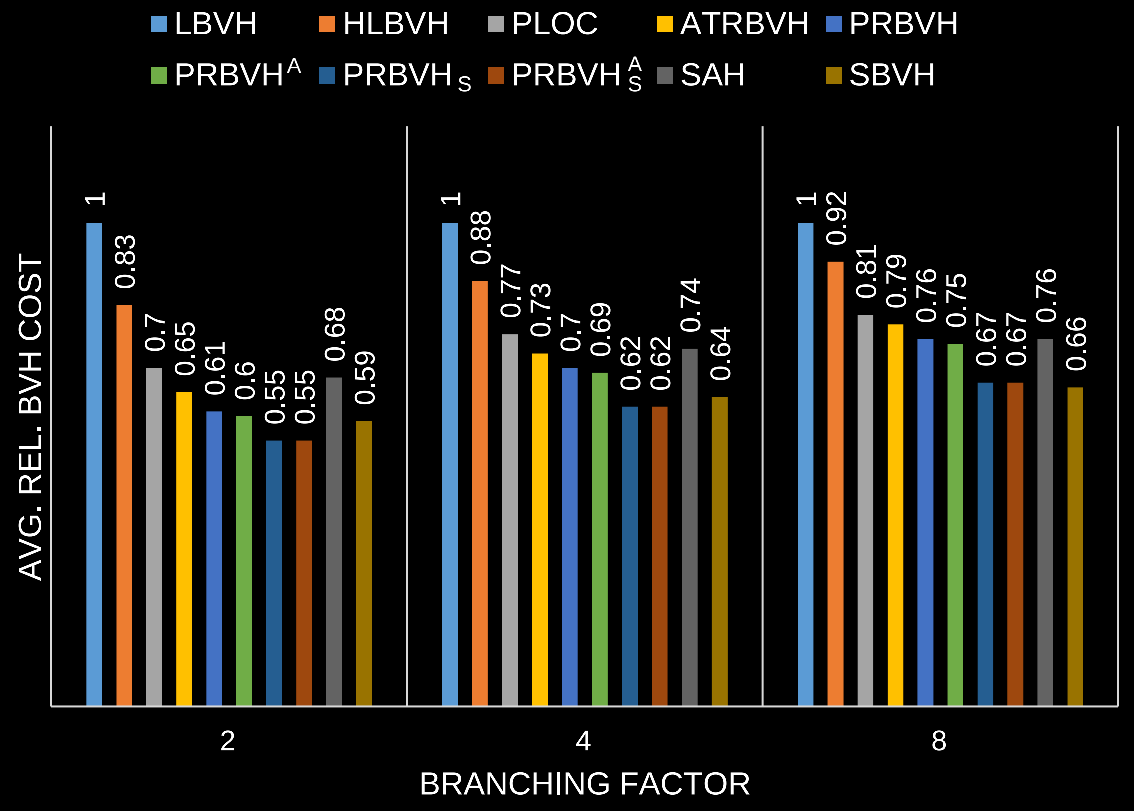


TESTED METHODS

- LBVH: 60-bit Morton Codes
- HLBVH: 60-bit Morton Codes and 15 bits for SAH splits
- PLOC: 60-bit Morton Codes and search radius 100
- ATRBVH: 20 iterations with LBVH as a base BVH
- SAH: full-sweep top-down (SBVH with disabled spatial splits)
- SBVH: 128 bins for spatial splits
- PRBVH: hill climbing with LBVH as a base BVH
- $PRBVH^{A}$: simulated annealing with LBVH as a base BVH
- $PRBVH_{S}$: hill climbing with SBVH as a base BVH
- $PRBVH_{S}^{A}$: simulated annealing with SBVH as a base BVH

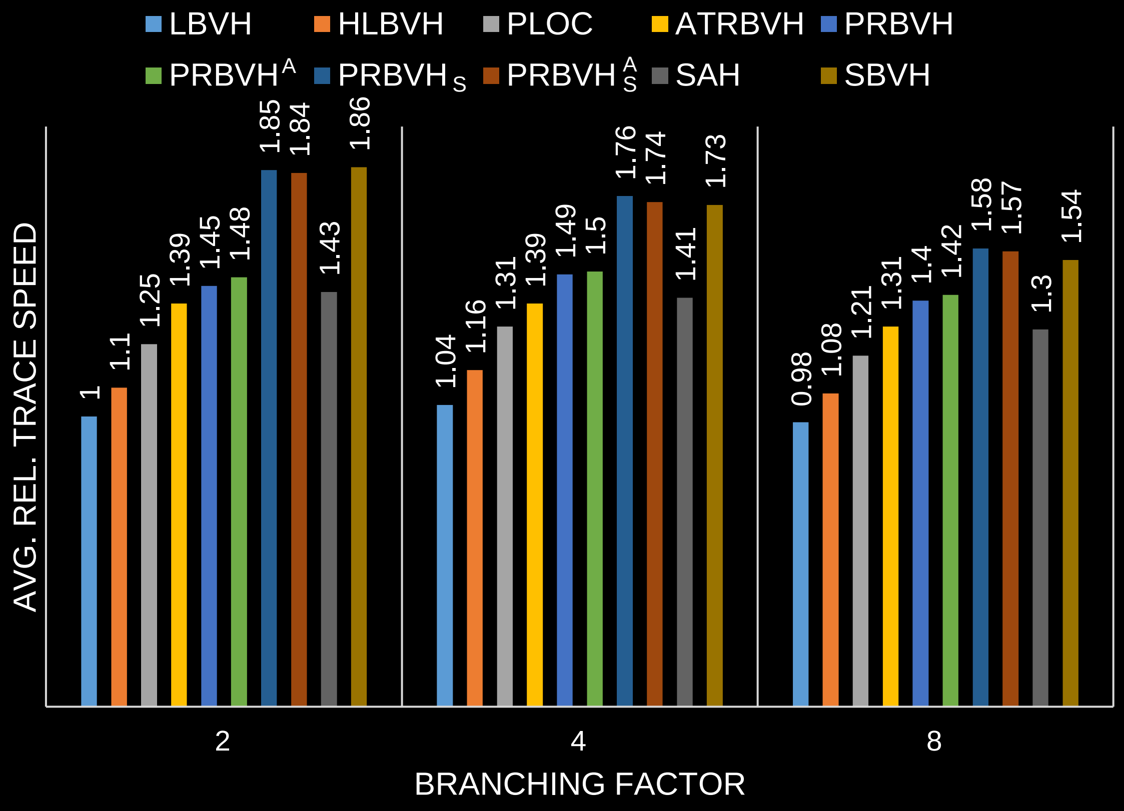
AVERAGE RELATIVE BVH COST

- Normalized by LBVH of the corresponding branching factor and averaged over all scenes
  - Higher branching factors have lower nodes and thus lower (absolute) BVH costs
- PRBVH can improve SBVH about 7% on average
- Only marginal improvement for simulated annealing
- Spatial splits provides significant improvement about 13% on average



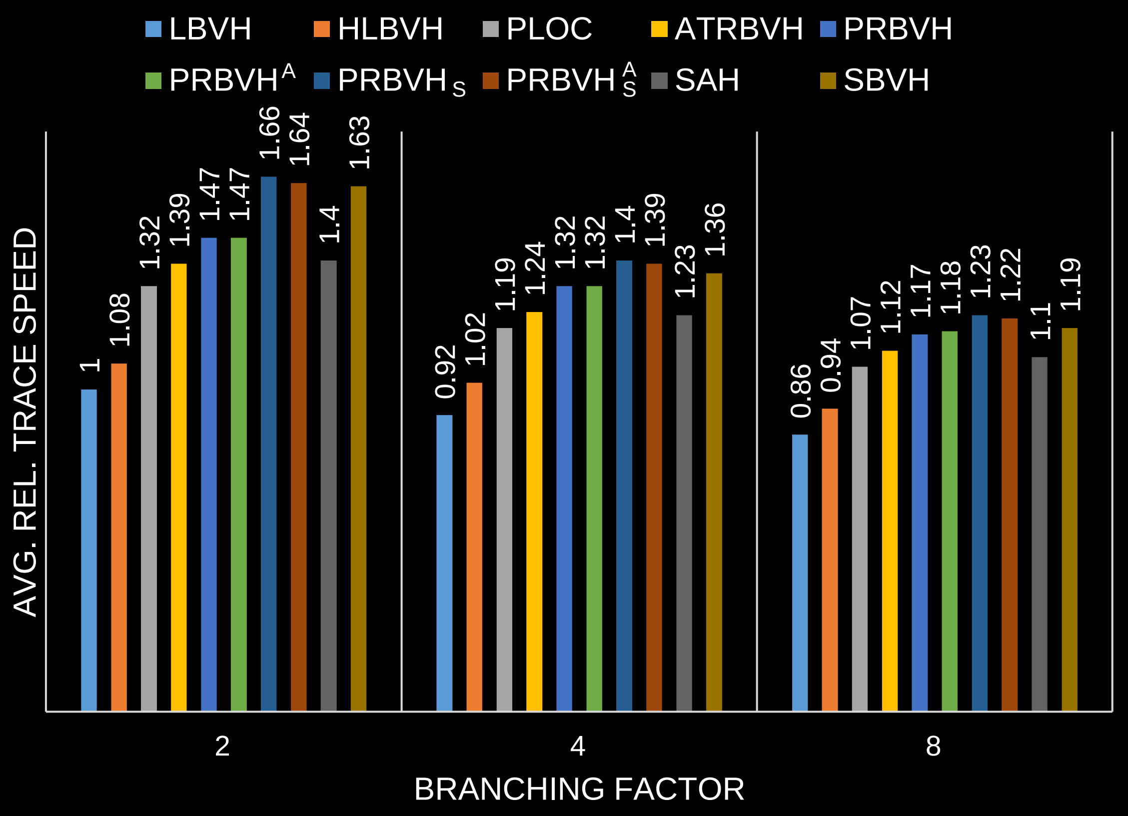
AVERAGE RELATIVE TRACE SPEED – SECONDARY RAYS

- Normalized by binary LBVH and averaged over all scenes
- PRBVH improves SBVH in most of the cases
- Simulated annealing worse than SBVH
  - Breaks well optimized top splits
- Spatial splits improve trace speed about 24% on average
  - Heavily depends on a particular scene
- Trace speed drops with increasing branching factor
  - Different traversal algorithm



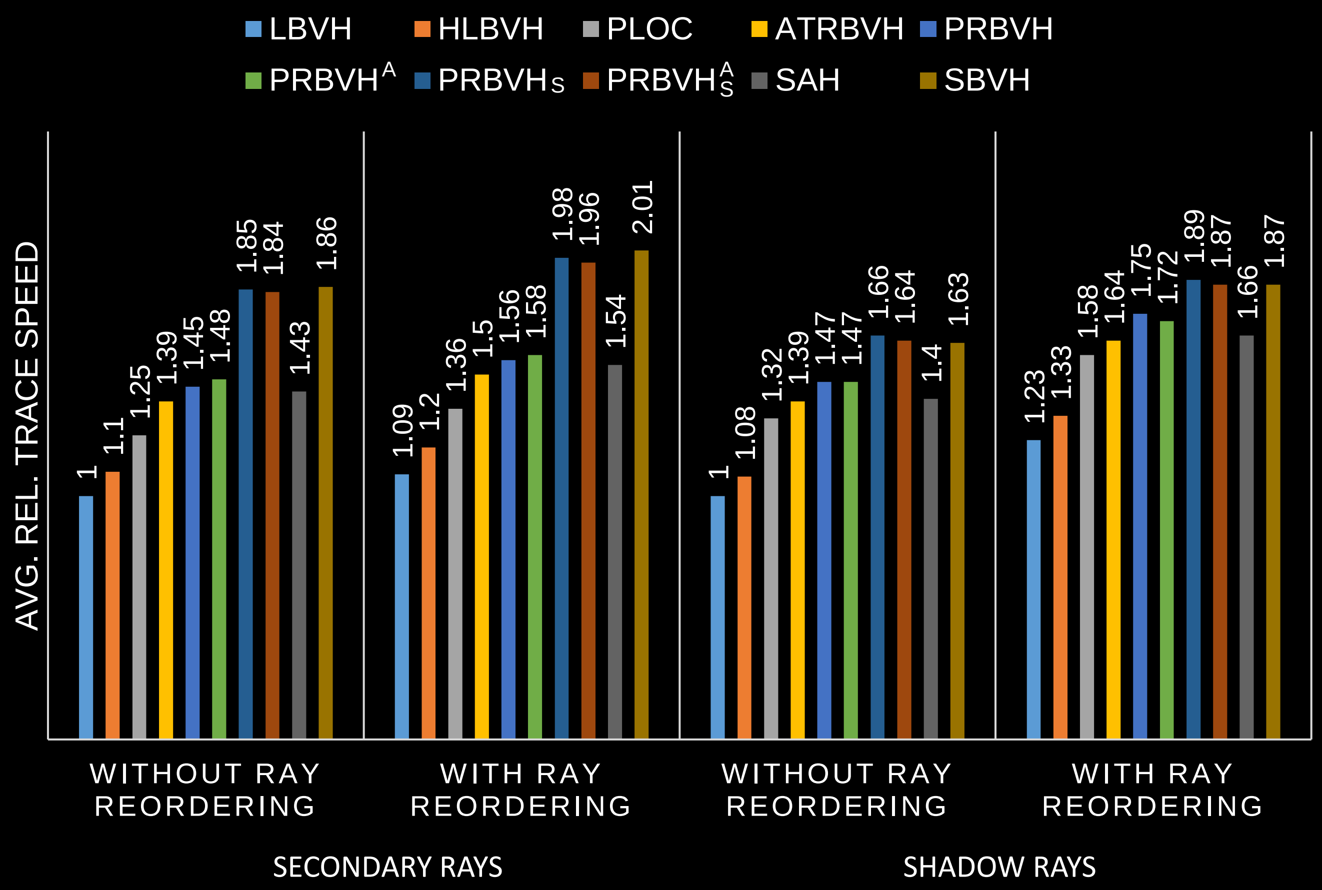
AVERAGE RELATIVE TRACE SPEED – SHADOW RAYS

- Normalized by binary LBVH and averaged over all scenes
- PRBVH improves SBVH in most of the cases
- Simulated annealing worse than SBVH
  - Breaks well optimized top splits
- Spatial splits improve trace speed about 12% on average
  - Heavily depends on a particular scene
- Trace speed drops with increasing branching factor
  - Different traversal algorithm



AVERAGE RELATIVE TRACE SPEED – RAY REORDERING

- Normalized by binary LBVH and averaged over all scenes
  - Ray reordering overhead included in trace times
- Speedup 7% for secondary rays and 8% for shadow rays
- It pays off in complex scenes with many rays
  - Extracting ray coherence
  - Sorting algorithm is faster for large data
- Not good for object-like scenes
  - Very few rays as most of the primary rays escape the scene after the first hit



CONCLUSION

Contribution

- Extensive empirical performance comparison
- Unified framework implementing state-of-the-art algorithms
- Simulated annealing as an extension of PRBVH

Observations

- Binary SBVH provides excellent results
- SBVH can be improved by PRBVH
- Ray reordering pays off in most of the cases

THANK YOU FOR YOUR ATTENTION!

The framework source code available on Github:
https://github.com/meistdan/hippie