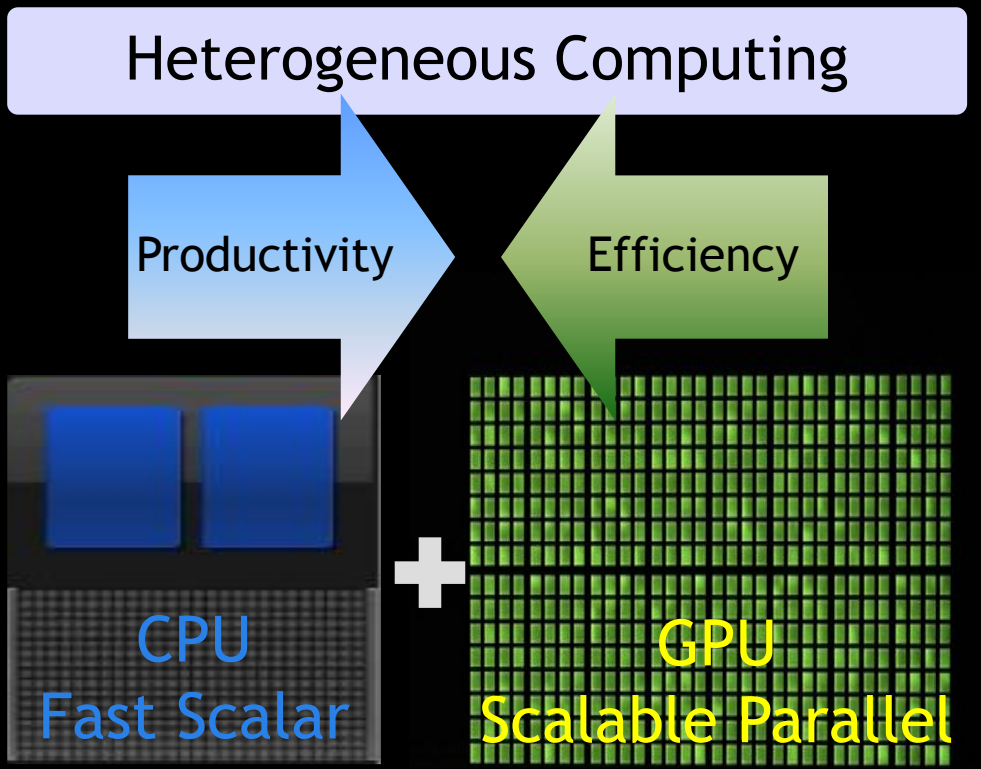
Parallel Compact Roadmap Construction of 3D Virtual Environments on the GPU

Avi Bleiweiss
NVIDIA Corporation

PRESENTED BY NVIDIA.

Programmability

- GPU Computing
  - CUDA C++
  - Parallel Debug

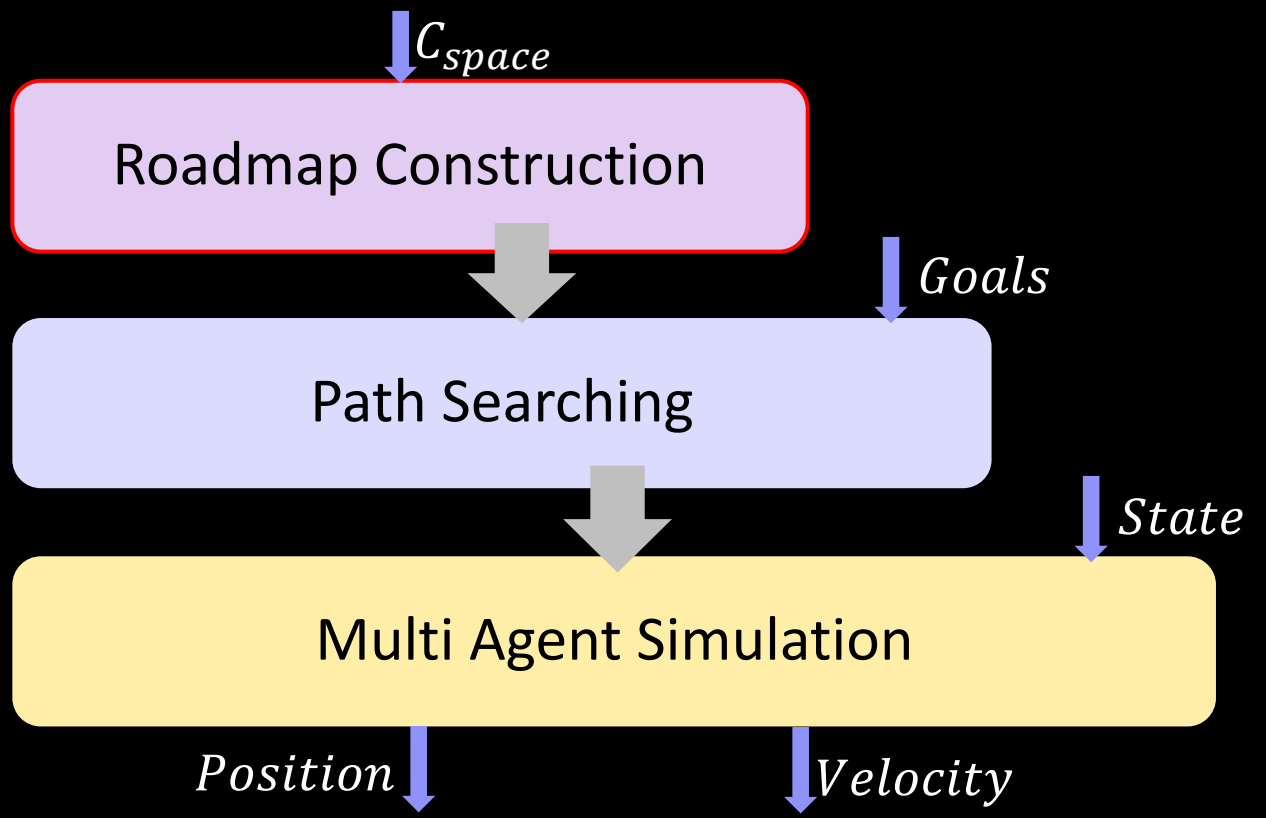


IROS 2010

Fermi Architecture

- 512 cores
- 1024 threads/SM
- L1/L2
- 1.5 Tflops

Planning System



Roadmap

Rapidly-exploring Random Tree (RRT)

- Simple, fast
- Unexplored space bias
- In-lined build, query

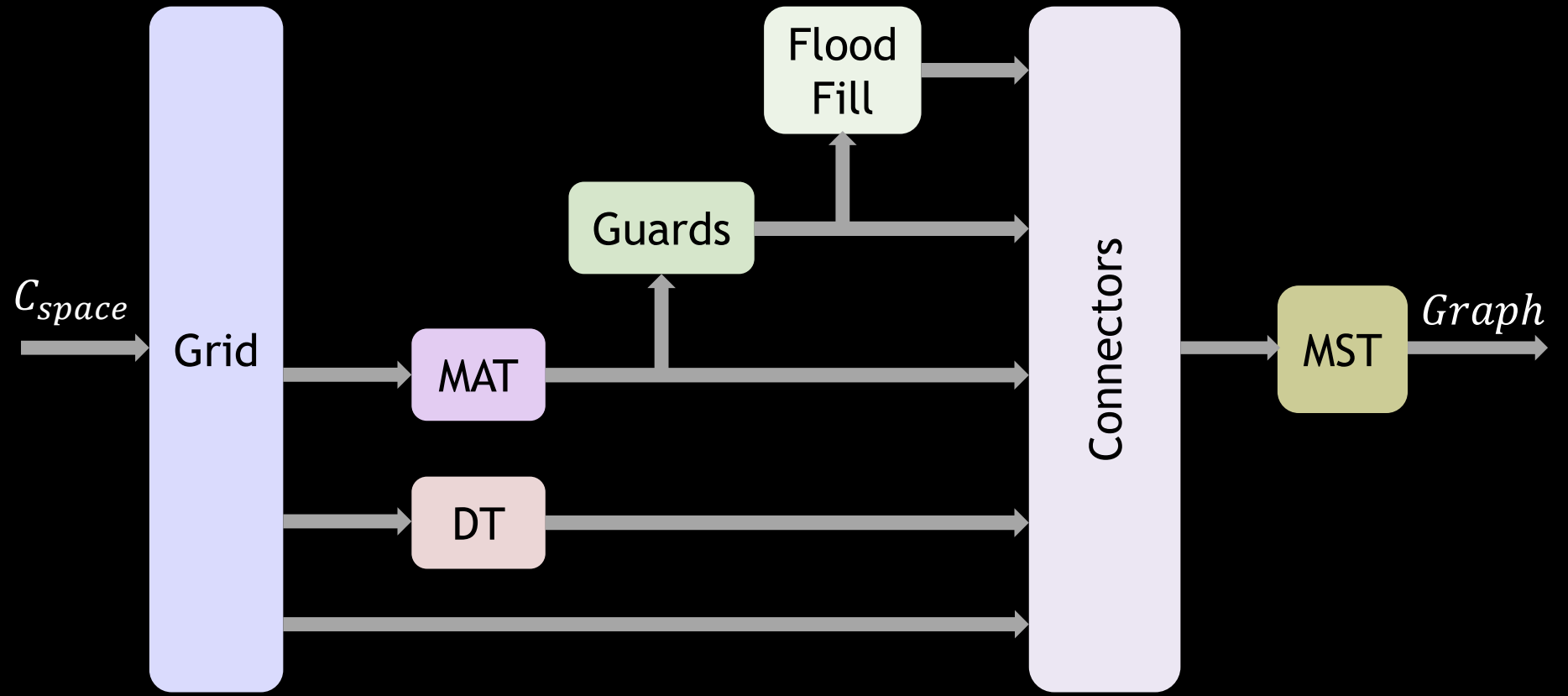
Probabilistic Roadmap (PRM)

- Uniform sampling
- Completion unbound
- Often large footprint
- Highly scalable

Reachability Roadmap (RRM)

- Resolution complete
- 2D/3D environments
- Compact graph

Construction

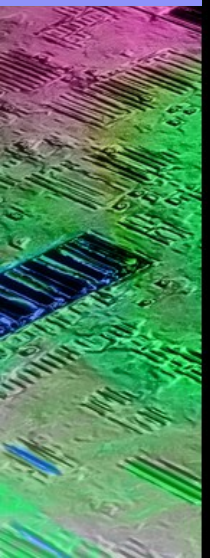


[Geraerts and Overmars 2005]

IROS 2010

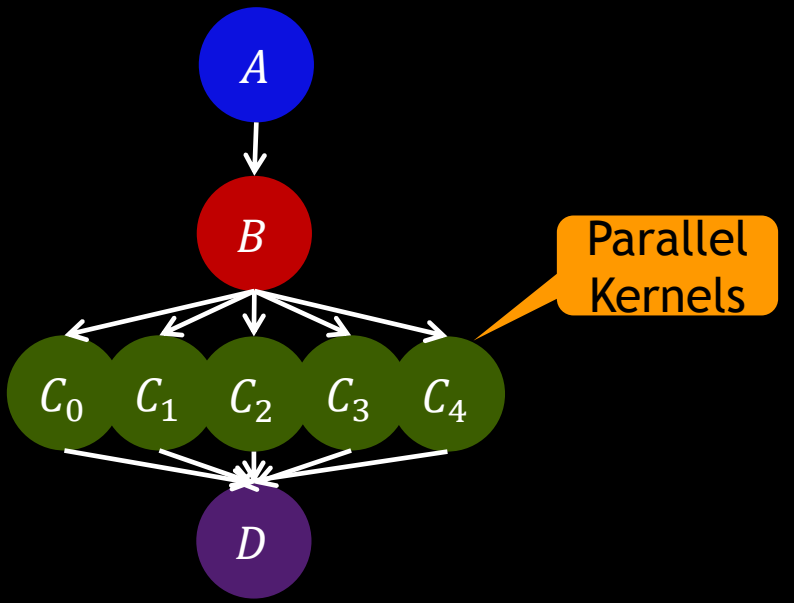
Challenges

- Large, dependent 3D data structures
- Divergent, irregular threads
- Coverage inherently serial
- Limited Flood Fill stack



Implementation

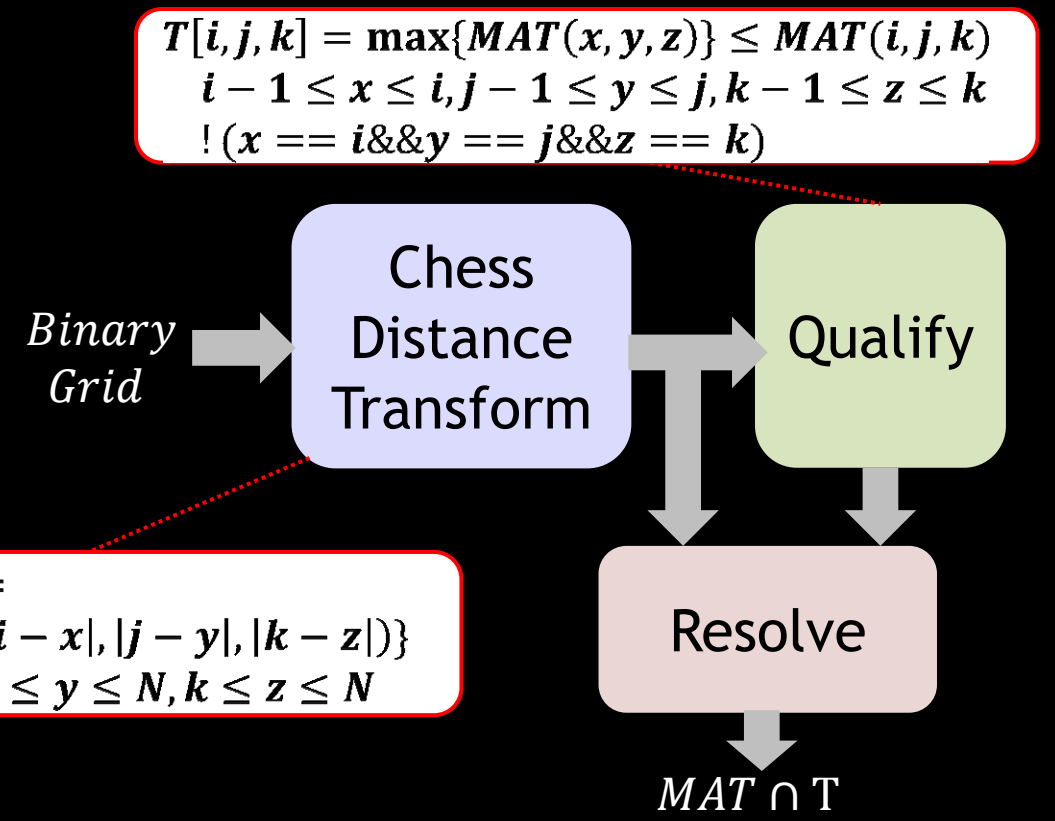
- Global memory resources
  - 2D/3D pitched linear
- Kernel per RRM stage
  - Implied synchronization
  - Free intermediate data
- Dynamic parallelism



DAG Task Manager

Medial Axis Transform

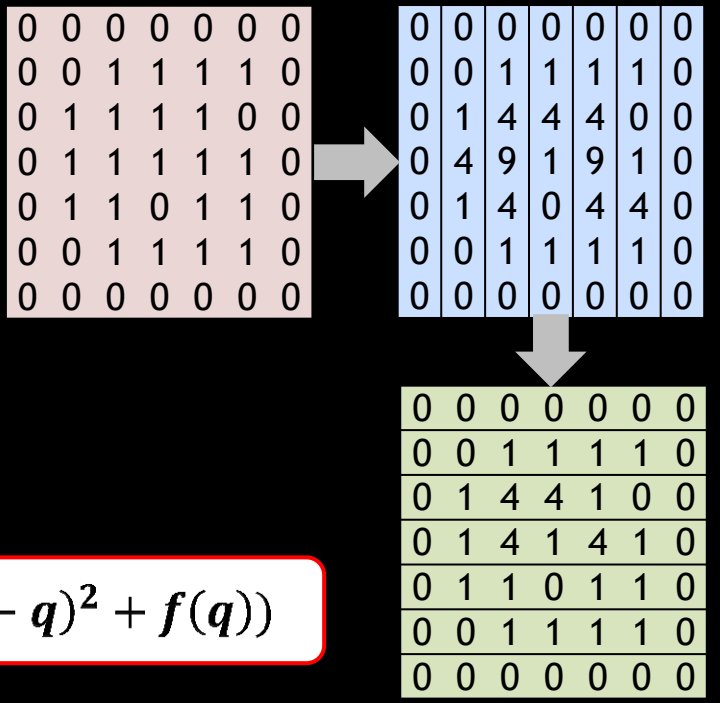
- Serial running time $O(kn^3)$
- n^3 GPU threads, per pass
  - $O(k)$time for CDT
  - $O(1)$for qualifier
  - $O(1)$for resolve



[Lee and Horng 1996]

Distance Transform

- Squared Euclidian distance
- Serial running time $O(n^3)$
- Parallel linear time $O(n)$
  - Slice, column, row passes
  - n^2 GPU threads, per pass



$$DT_f(p) = \min((p-q)^2 + f(q))$$

[Felzenszwalb and Huttenlocher 1996]

Flood Fill

- Obstacle aware
  - 3D line drawing
- Parallel initial guards
- Single cell
  - Large stack
- Scan line slower!
  - Smaller stack

```
S  = stack of cells (c)
G  = list of guards (g)
C  = 3D coverage data structure

S ← g
while  S not empty do
  if g visible from g_i ∈ G break
  c ← pop S
  if g not visible from c continue
  C[c] = C[c] ∪ g
  foreach neighbor n_i of c do
    if n_i ∈ C_free && n_i not covered do
      S ← n_i
```

Connectors

- Read-only produced data
  - 3D grid, MAT, DT, coverage
- Parallel surviving guards
  - $n(n-1)/2$ threads
- A connector per thread

```
G = list of guard indices
C = 3D coverage data structure
N = list of connectors

foreach pair (g_i, g_j) ∈ G: i < j do
  foreach cell c ∈ C do
    if g_i ∄ c_set or g_j ∄ c_set continue
    if (g_i, g_j) ∈ N do
      resolve connector
    else do
      insert connector
```

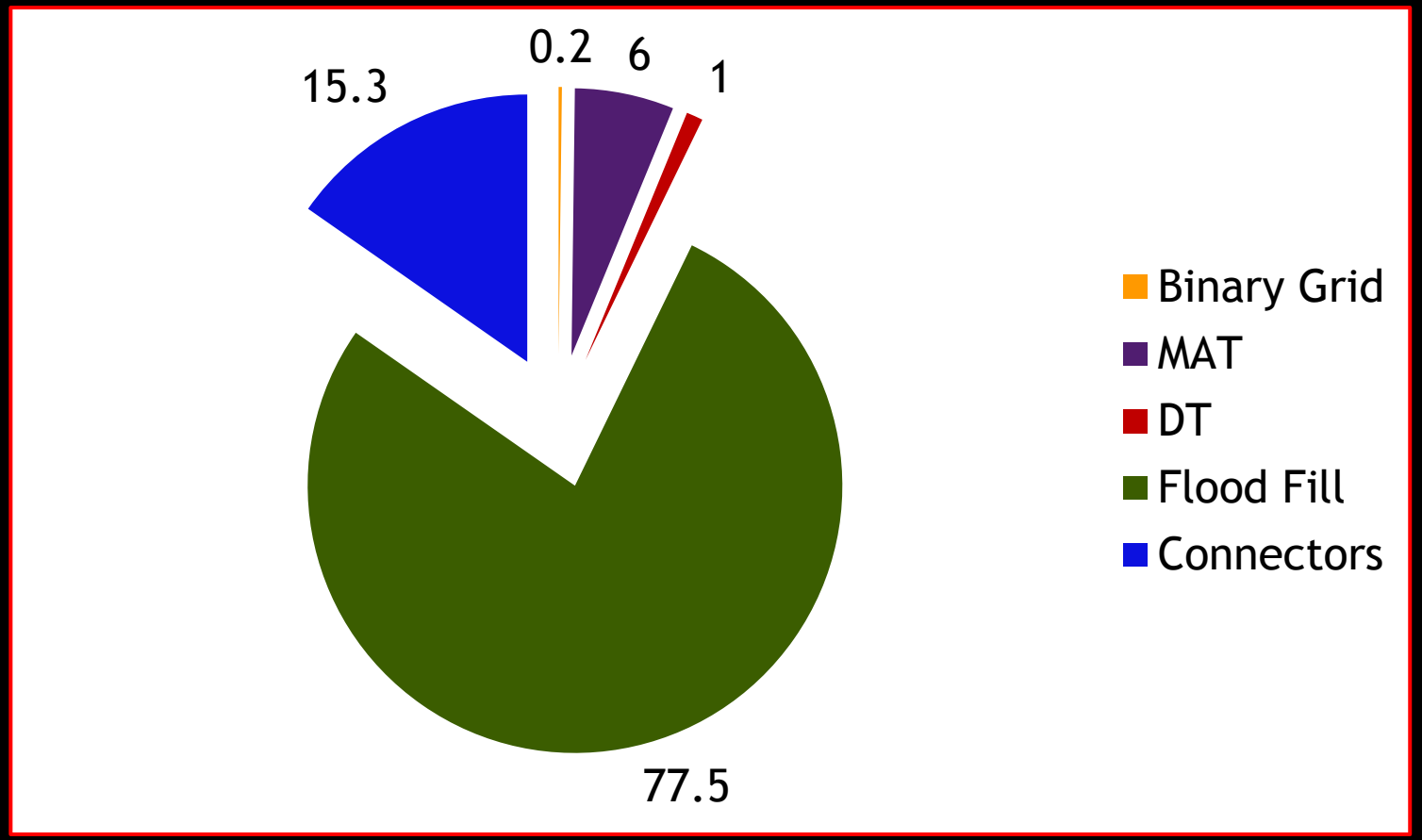
Experiments

C_{space}		Closure Resolution	GPU Threads	
Vertices	Faces		MAT	DT
82800	34750	33	35397	1089
161463	64451	40	64000	1600
347223	170173	55	166375	3025

Statistics

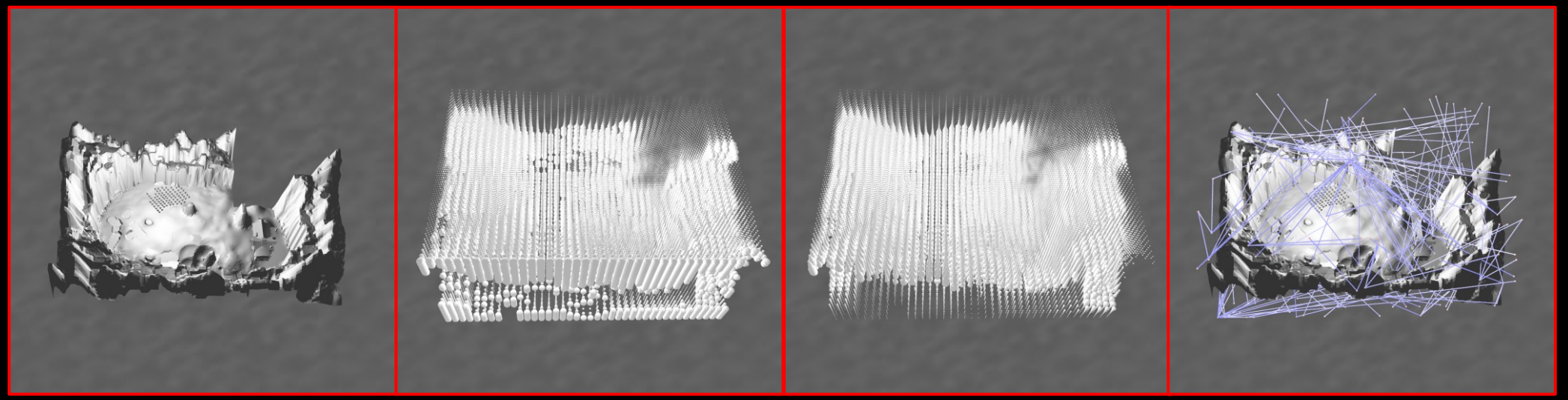
Guards	Coverage	Connectors	Graph		
			Nodes	Edges	Weight
15956	99.79%	404	114	109	128.72
27747	99.74%	1373	287	286	352.42
71599	99.89%	3154	782	764	406.08

Process



IROS 2010

Results



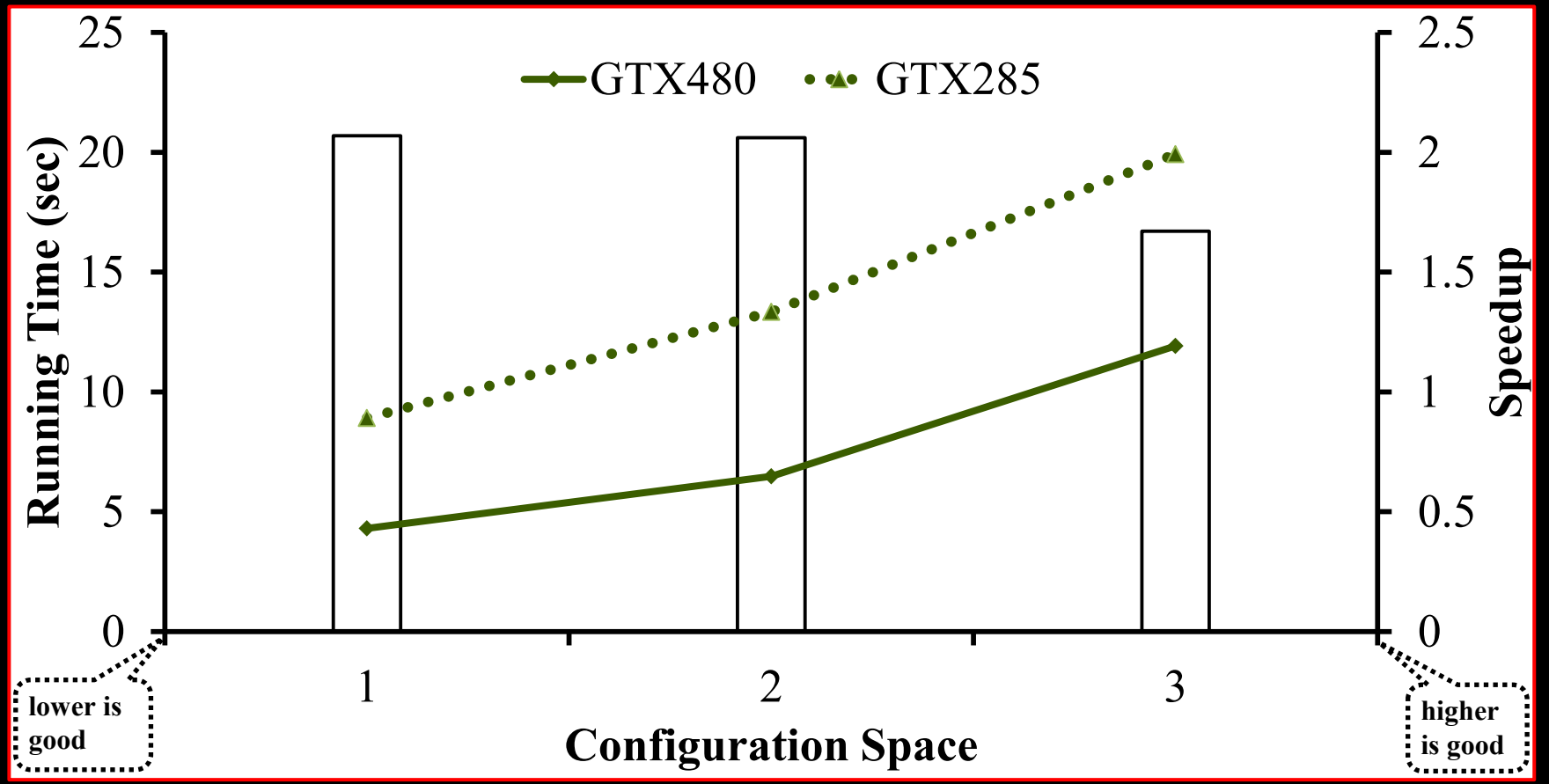
C_{space} MAT DT $Graph$

Processors

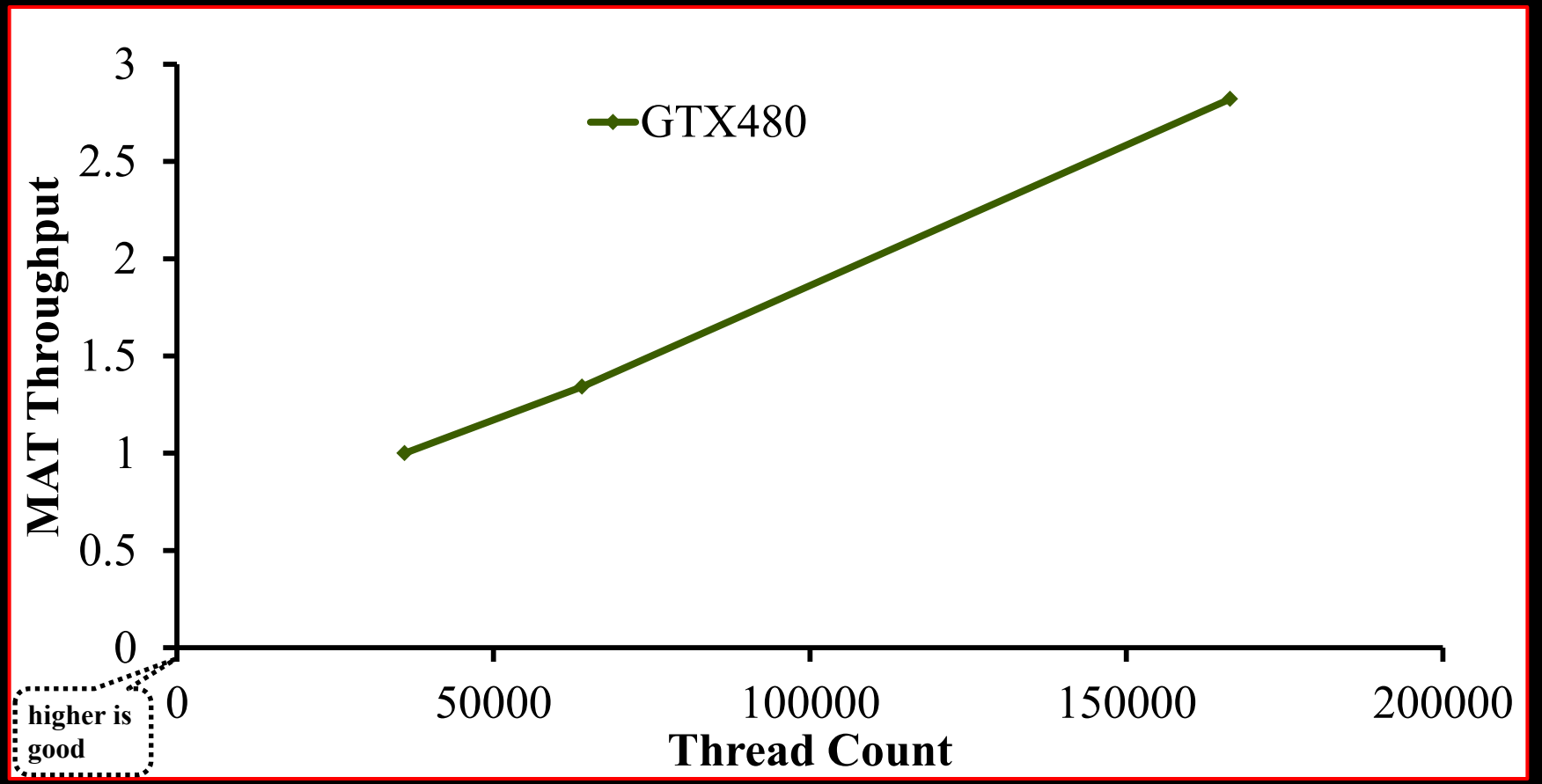
GPU	SMs	Warps/SM	Clocks (MHz)	L1/Shared (KB)
GTX480	15	2	723/1446/1796	48/16
GTX285	30	1	648/1476/1242	NA

Fermi Scale	
compute	0.98
memory	1.08

Running Time



Throughput


GTX480
MAT Throughput
3
2.5
2
1.5
1
0.5
0
0
50000
100000
150000
200000
Thread Count
higher is good

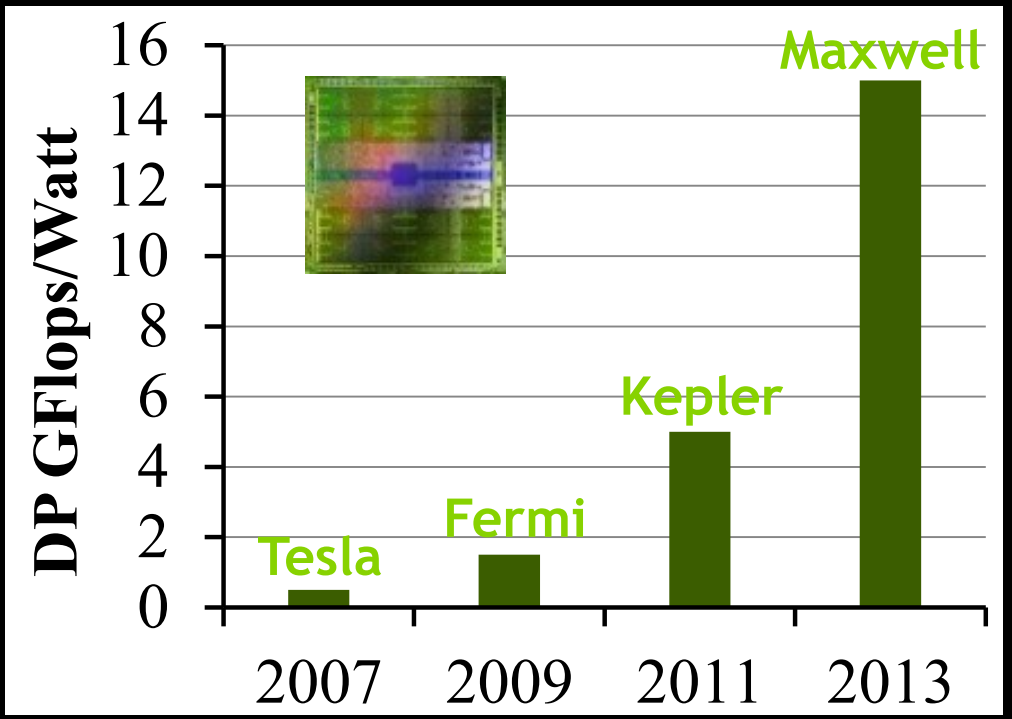
Limitations

- Stack space bounds
  - Less concurrency
- <100% coverage
  - MAT samples insufficient
- MST single threaded

Guards	Flood Fill Launches		
1024	16	28	72

Future Work

- Medial axis retraction
  - Bucket DT cells
- Shorter path extractions
  - Add useful cycles
- High clearance paths

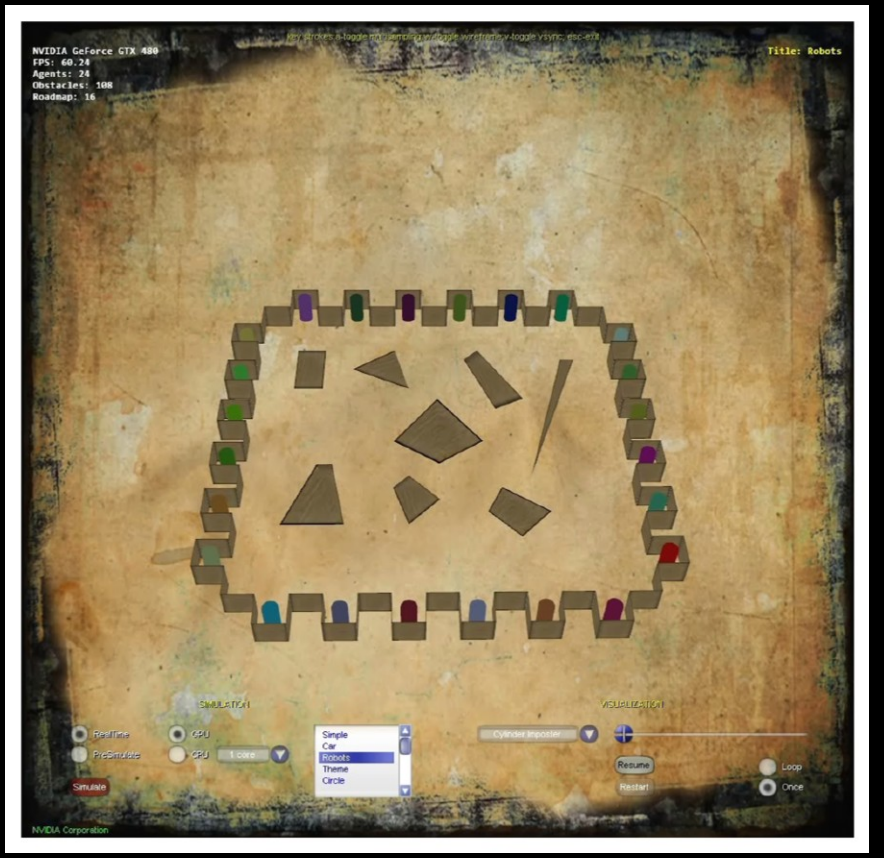


[Geraerts and Overmars 2006]

IROS 2010

Summary

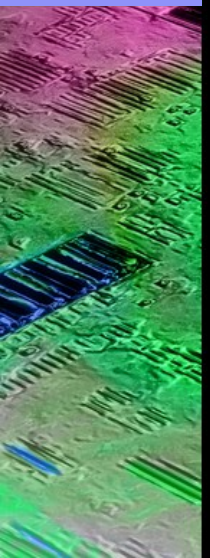
- Parallel RRM
  - More work remains
- GPU roadmap
  - Dynamic environments
- Programming tools
  - Constantly improving



Thank You!

Questions?

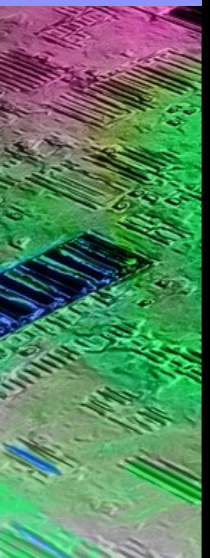
IROS 2010



Info

- Base
  - http://developer.nvidia.com
- GPU AI
  - Technology Preview
- Toolkit
  - CUDA Zone
- Debugger
  - Parallel Nsight

IROS 2010



Backup

IROS 2010

Appendix

- Compute scale
  - $\left(\frac{SMClk_{GTX480}}{SMClk_{GTX285}}\right) * \left(\frac{((Warps/SM)*SMs)_{GTX480}}{((Warps/SM)*SMs)_{GTX285}}\right)$
- Memory scale
  - $\left(\frac{MemClk_{GTX480}}{MemClk_{GTX285}}\right) * \left(\frac{MemBusWidth_{GTX480}}{MemBusWidth_{GTX285}}\right)$
- GTX480 L1/Shared (KB) config
  - Up to 1.35X faster in 48/16 vs. 16/48

Running Time (1)