Stanford CS193G Spring 2010

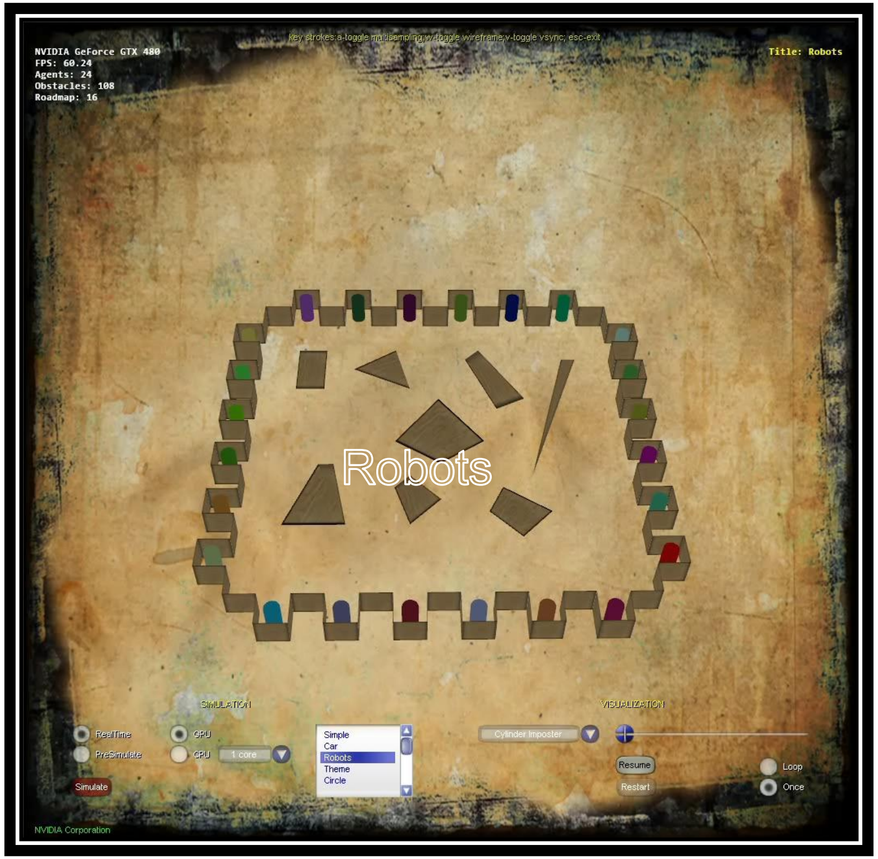
Path Planning System on the GPU

Avi Bleiweiss
NVIDIA Corporation

Stanford CS193G Spring 2010

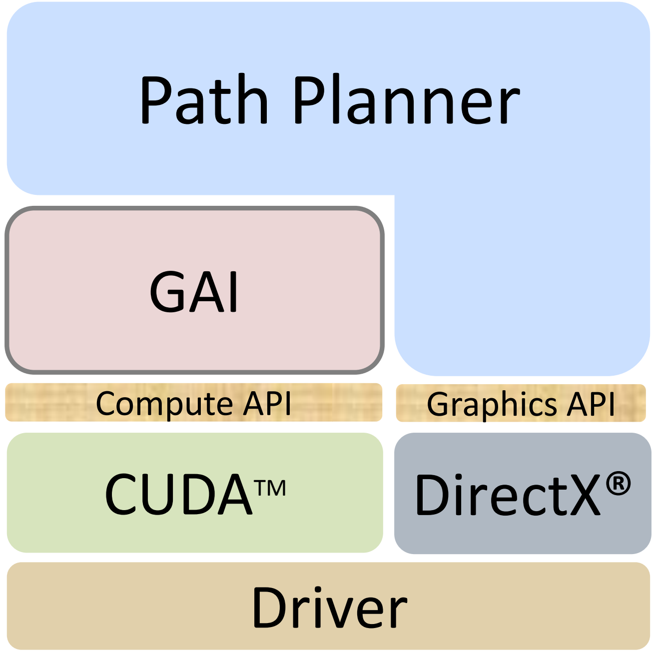
Reasoning

- Explicit
  - State machine, serial
- Implicit
  - Compute intensive
  - Fits SIMT well
- Path planning



Motivation

- GPU accelerated AI
- Congestion games
- Effective team tasks
  - Virtual robots, humans
- Scalable, real time



Problem

Planner

- Efficient roadmap construction
  - From 3D virtual environment
- Searches a global, optimal path
  - From start to goal
- Locally, avoids collisions with
  - Static, dynamic objects

Simulator

- Visually compelling motion
- Economical memory footprint
- A subset of compute units
- Linear scale with # characters

Stanford CS193G Spring 2010

Solution

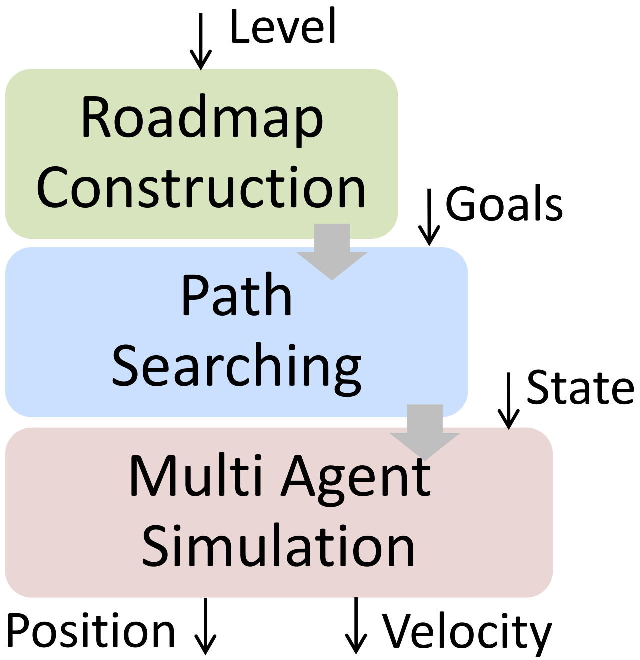
- Compact, quality roadmap
- Heterogeneous agents
- Velocity Obstacles
- GPU optimizations
  - Spatial hash
  - Nested parallel

Outline

- Algorithm
- Implementation
- Results
- Takeaways

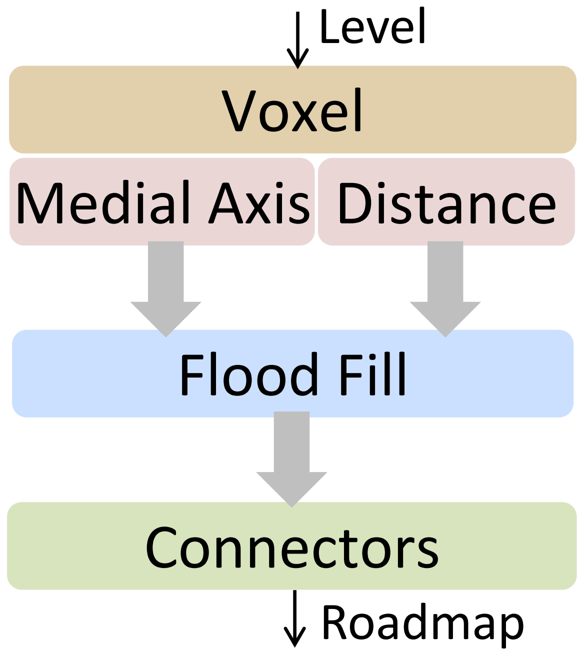
Pipeline

- 3D level, C_{space} mesh input
- Inline computed roadmap
- Goals, roadmap decoupled
- Discrete time simulation



Roadmap Construction

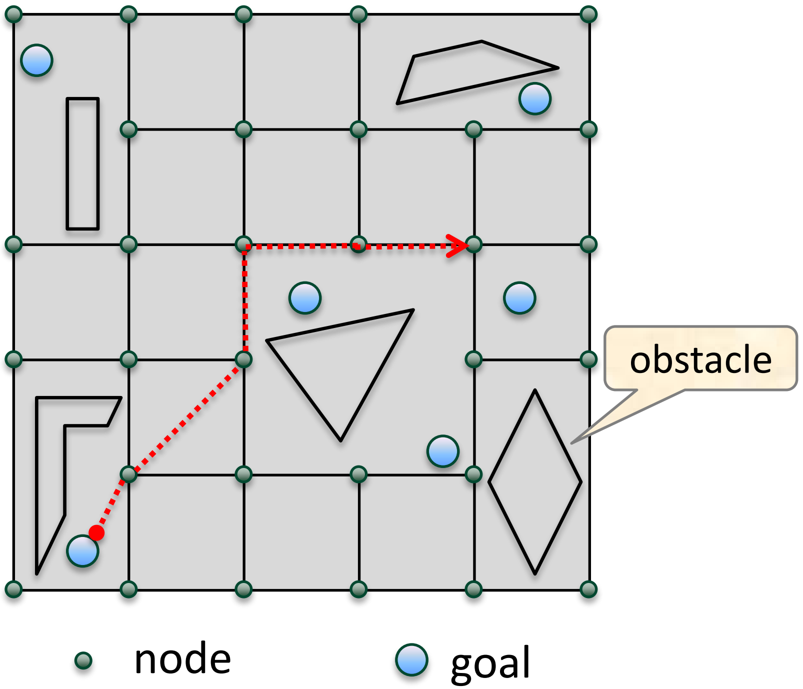
- An existed C_{free} path
  - Guaranteed in roadmap
- Predictable termination
- 3D grid operators
  - Highly parallelizable



[Geraerts and Overmars 2005]

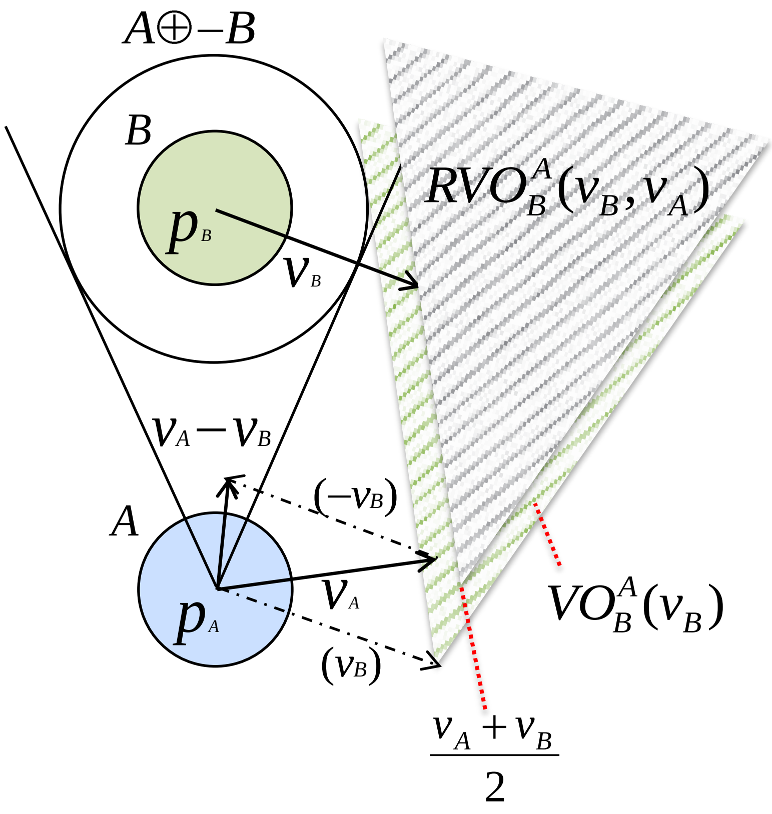
Visibility

- Two sets of edges
  - Visible roadmap node pairs
  - Goals to unblocked nodes
- Static obstacles outline
- A* search, shortest path
  - From goal to any node



Velocity Obstacles

- Well defined, widely used
- Avoidance velocity set[1]
- Reciprocal Velocity Obstacles[2]
  - Oscillation free motion
- Agents moving in 2D plane



1 [Fiorini and Shiller 1998]
2 [Van Den Berg et al. 2008]

Multi Agent Simulation

- Simulator advances until
  - All agents reached goal
- Path realigned towards
  - Roadmap node or goal
- Agent, velocity parallel

```
do
  hash
    construct hash table
  simulate
    compute preferred velocity
    compute proximity scope
    foreach velocity sample do
      foreach neighbor do
        if OBSTACLE then VO
        elseif AGENT then RVO
    resolve new velocity
  update
    update position, velocity
    resolve at-goal
while not all-at-goal
```

Challenges

Hiding memory latency

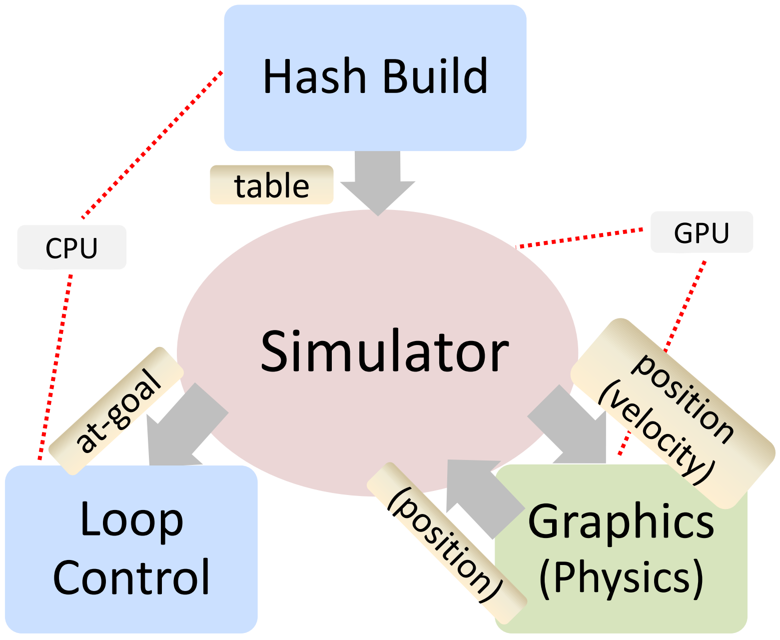
Divergent, irregular threads

Small agent count (≤32)

Hash construction cost

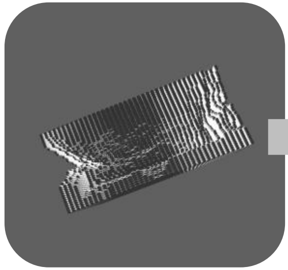
Workflow

- Roadmap static for
  - 100s simulation steps
- Dependent resources
  - Linear, pitched 3D
- Dozen compute kernels
- Split frame, multi GPU



Medial Axis Transform

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



Chess Distance Transform → Qualify → Resolve

$$MAT(i,j,k) = \min\{\max(|i-x|,|j-y|,|k-z|)\}$$
$$i \le x \le N, j \le y \le N, k \le z \le N$$

$$T[i,j,k] = \max\{MAT(x,y,z)\} \le MAT(i,j,k)$$
$$i-1 \le x \le i, j-1 \le y \le j, k-1 \le z \le k$$
$$!(x == i \,\&\&\, y == j \,\&\&\, z == k)$$

[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

0	0	0	0	0	0	0
0	0	1	1	1	1	0
0	1	1	1	1	0	0
0	1	1	1	1	1	0
0	1	1	0	1	1	0
0	0	1	1	1	1	0
0	0	0	0	0	0	0

0	0	0	0	0	0	0
0	0	1	1	1	1	0
0	1	4	4	4	0	0
0	4	9	1	9	1	0
0	1	4	0	4	4	0
0	0	1	1	1	1	0
0	0	0	0	0	0	0

0	0	0	0	0	0	0
0	0	1	1	1	1	0
0	1	4	4	1	0	0
0	1	4	1	4	1	0
0	1	1	0	1	1	0
0	0	1	1	1	1	0
0	0	0	0	0	0	0

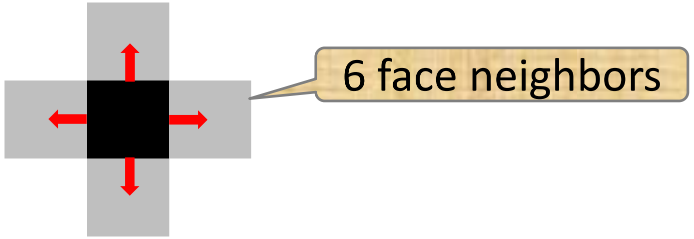
$$DT_f(p) = \min((p-q)^2 + f(q))$$

[Felzenszwalb and Huttenlocher 1996]

Flood Fill

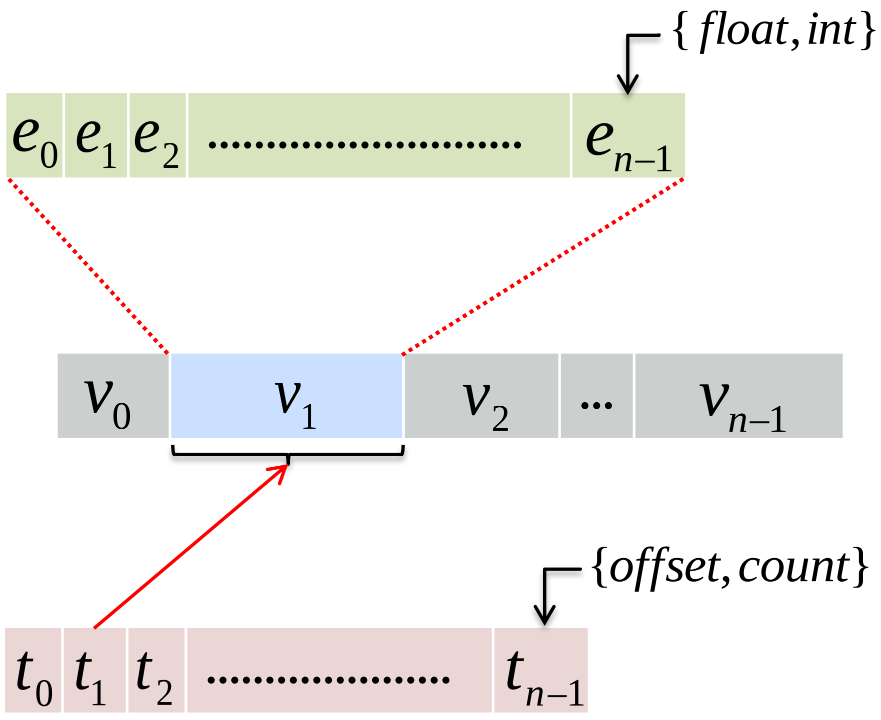
- Obstacle aware
  - 3D line drawing
- Parallel guards
- Single cell, private stack
- Scan line stack smaller
  - Runs slower!

```
push guard on to stack
while stack not empty do
  pop stack
  if guard not visible from cell continue
  add guard to cell's coverage set
  foreach adjacent neighbor cell do
    if neighbor in C_free && not covered do
      push neighbor on to stack
```



Data Layout

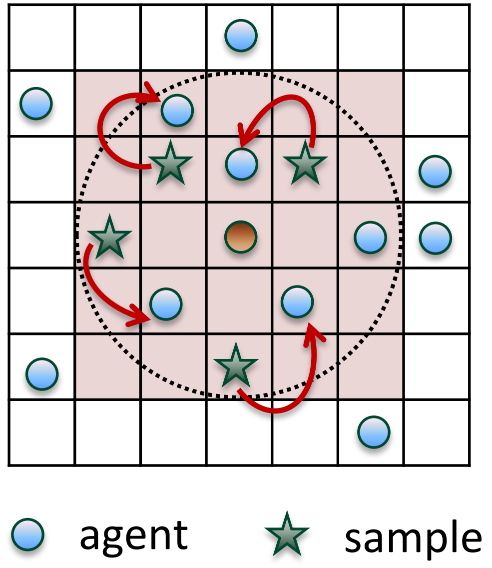
- Persistent resources
  - Reside in global memory
- Thread aligned data
  - Better coalescing
- Consistent access pattern
  - Improves bandwidth



Variable Length Vector Access

K-Nearest Neighbor

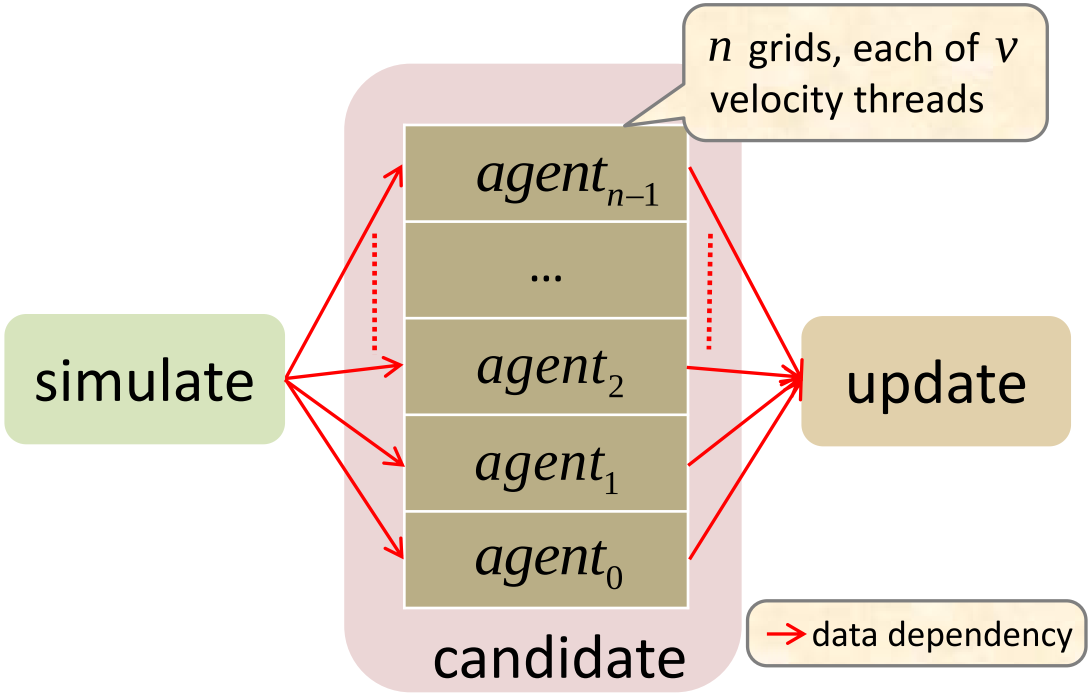
- Naïve, exhaustive search
  - $O(n^2)$ system running time
- Spatial hash
  - 3D point to a 1D index
- Per frame table build
  - Current agents' position



$$h(p) = \text{determinant}(p, p_{ref})$$

Nested Parallel

- Flat parallel limiting
- Thread grid DAG
  - Independent grids
  - Same kernel per level
- Thread amplification
  - Improved occupancy



Velocity Threads

- Hundreds of threads
- Graceful grid sync
- Fine reduce-min
  - Into Shared memory
- Global atomic CAS
  - Inter thread block

```
__global__ void
 candidate(CUAgent* agents,
                  int index,
                  CUNeighbor* neighbors)
{
   float3 v, float t;
   CUAgent a = agents[index];
   if(!getThreadId()) v = a.prefvelocity;
   else v = velocitySample(a);
    t = neighbor(a, agents, neighbors, v);
   float p = penalty(a, v, t);
    reduceMinAtomicCAS(a, p);   sync
    if(p == a.minpenalty)  a.candidate = v;
}
```

Methodology

- CUDA 3.1 Beta
- GPU properties

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[1]

compute	0.98
memory	1.08

[1] More info in appendix

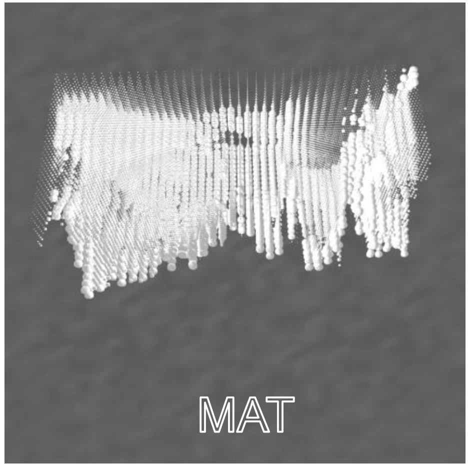
Views

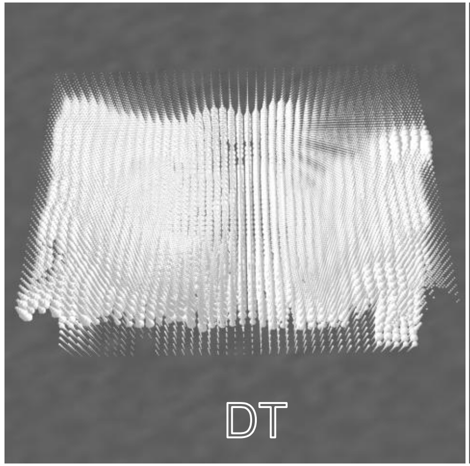
- Three views per stage
  - vs. GTX285
  - Relative throughput
  - vs. CPU
- Running time, frame rate
- Speedup vertical bars

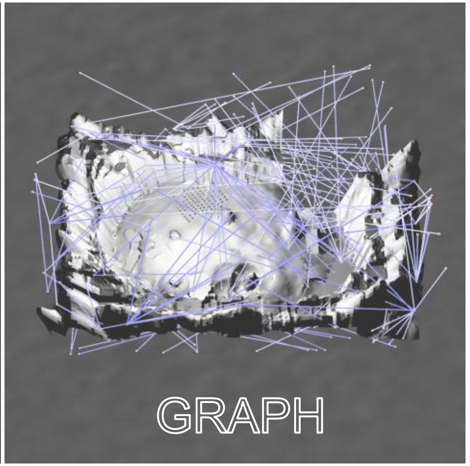
Property	GTX480	GTX285
Threads / SM	1024	512
L1 Cache (KB)	48	None
L2 Cache (KB)	768	None
Parallel Kernels	16	1

Roadmap Construction Experiments

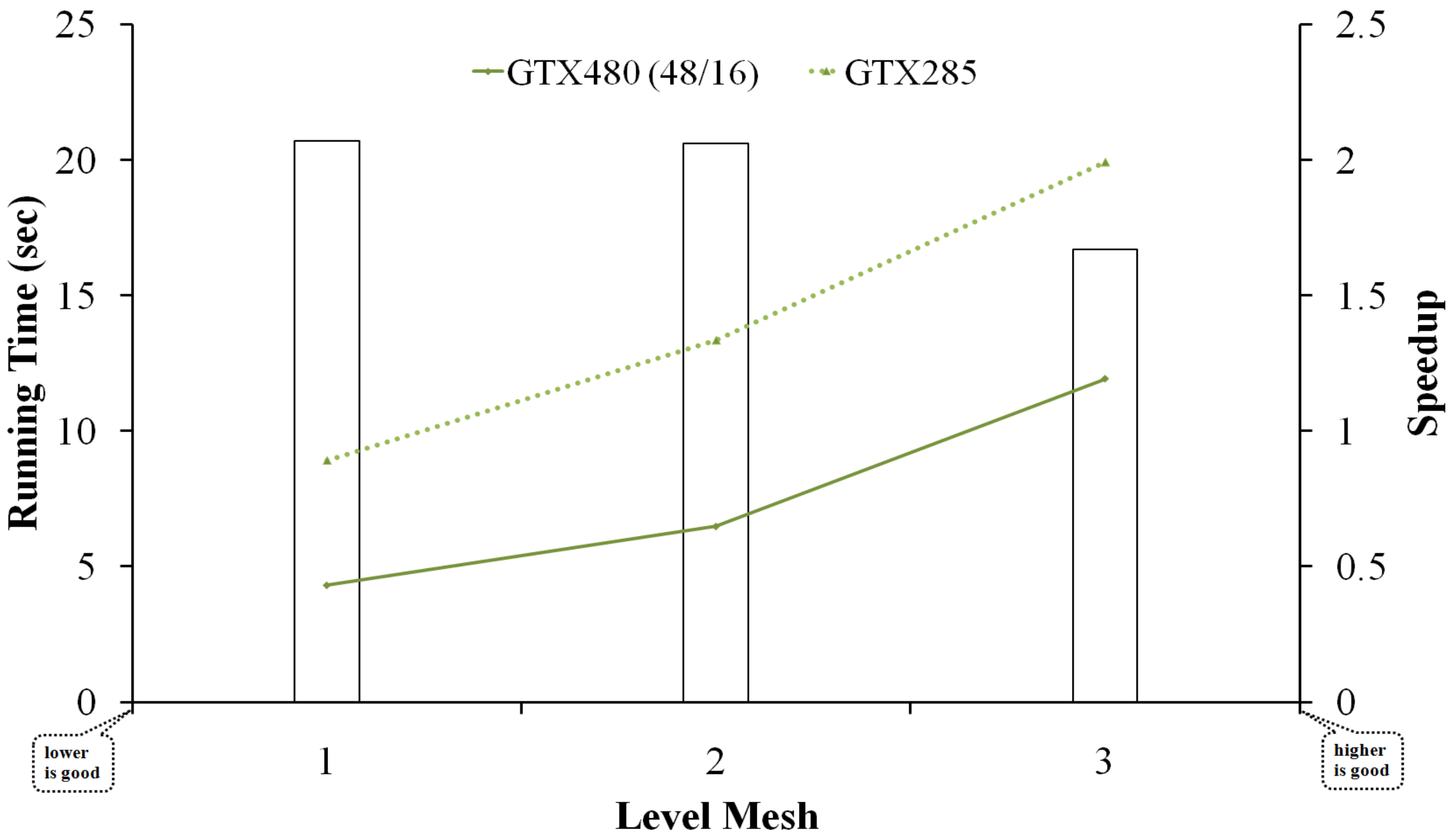
Level, C_{free}		Grid Resolution	GPU Threads		Graph	
Vertices	Faces		Distance	Medial Axis	Nodes	Edges
82800	34750	33	1089	35937	114	109
161463	64451	40	1600	64000	287	286
347223	170173	55	3025	166375	782	764



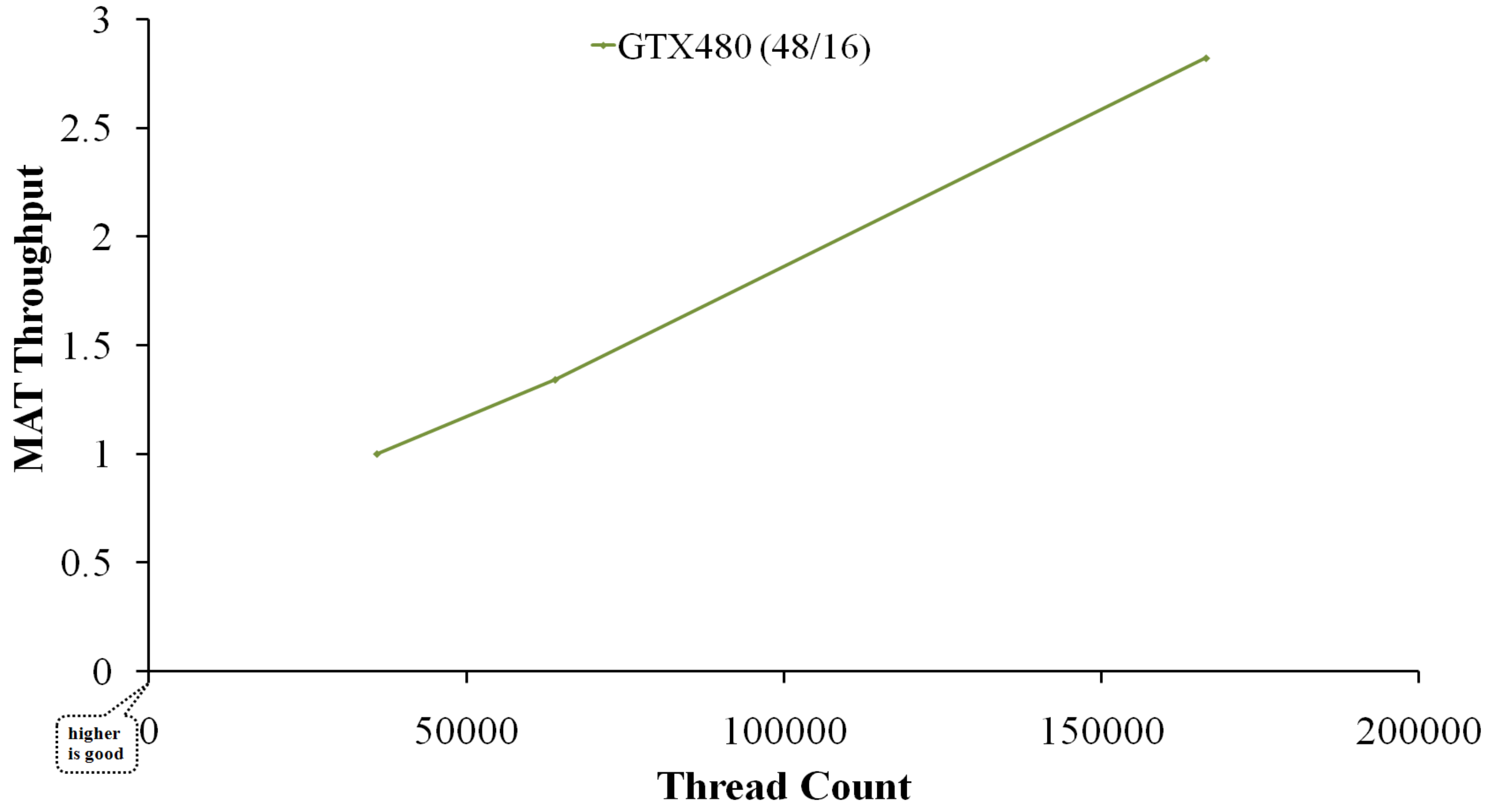




Roadmap Construction - vs. GTX285



Roadmap Construction - Throughput



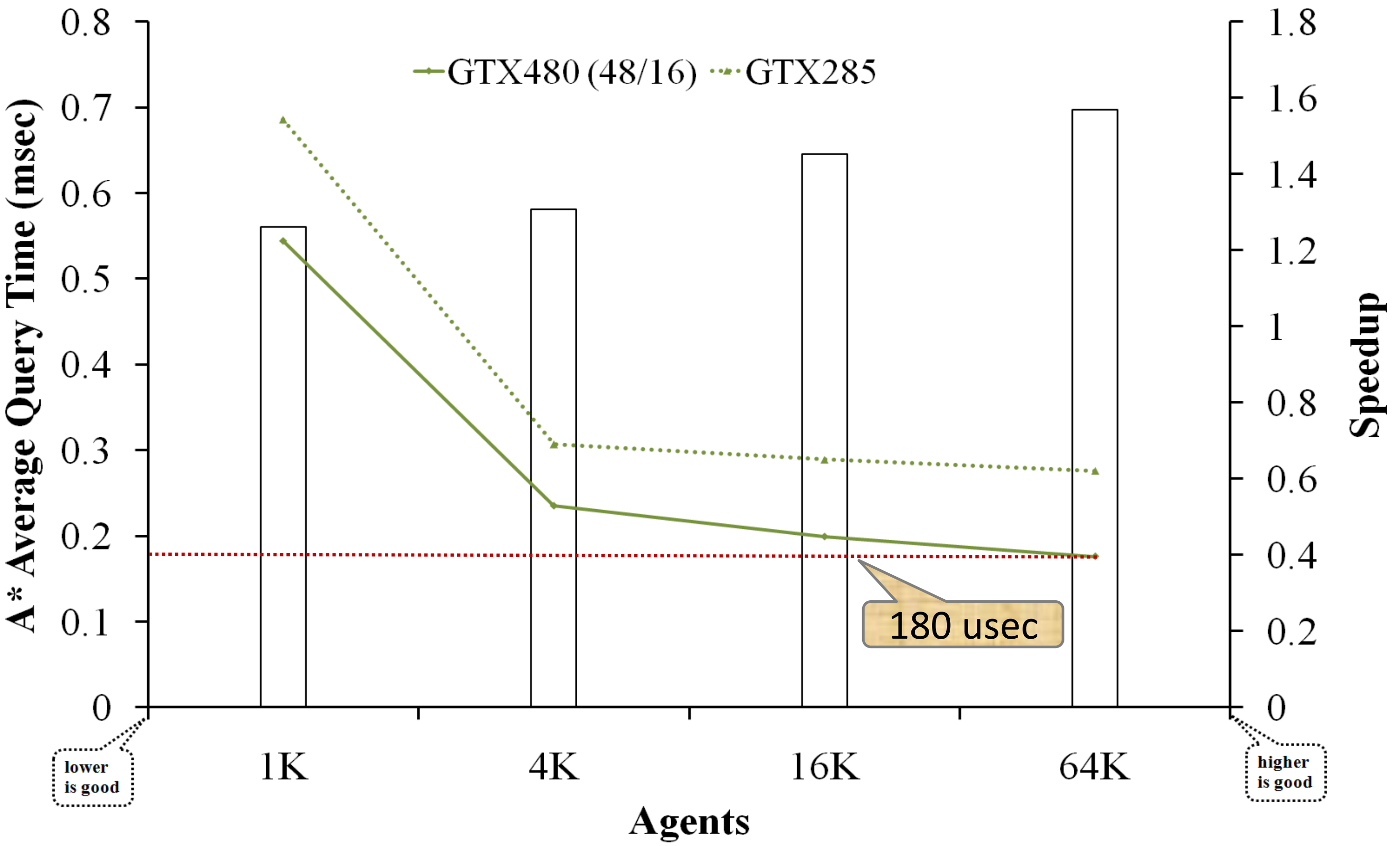
Stanford CS193G Spring 2010

Path Searching Experiments

Graph	Nodes	Edges	Agents	CTAs
Large	5706	39156	1024—65536	4—256

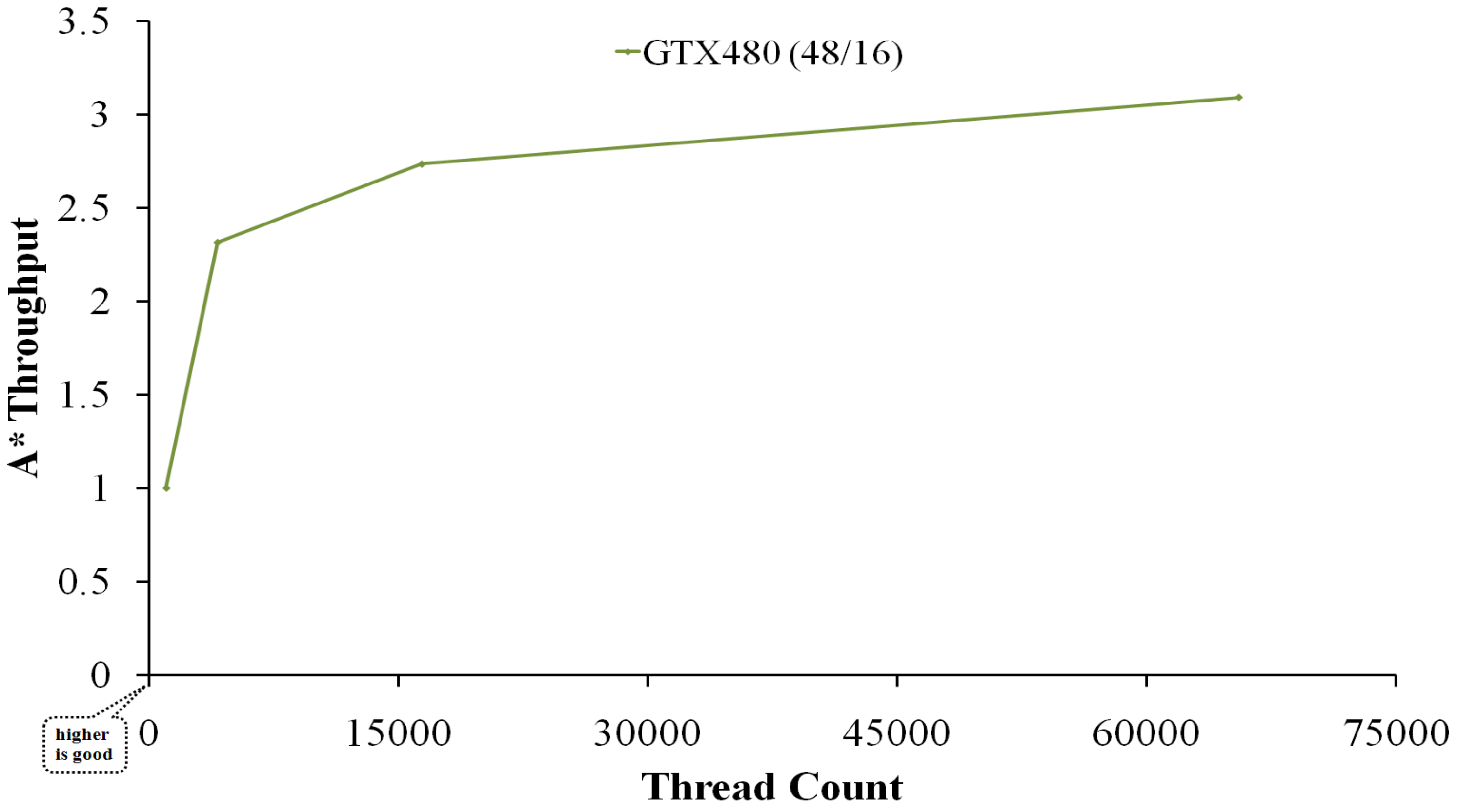
Agents of random start and goal pair configurations

Path Searching - vs. GTX285



average query time = total running time / agent #

Path Searching - Throughput

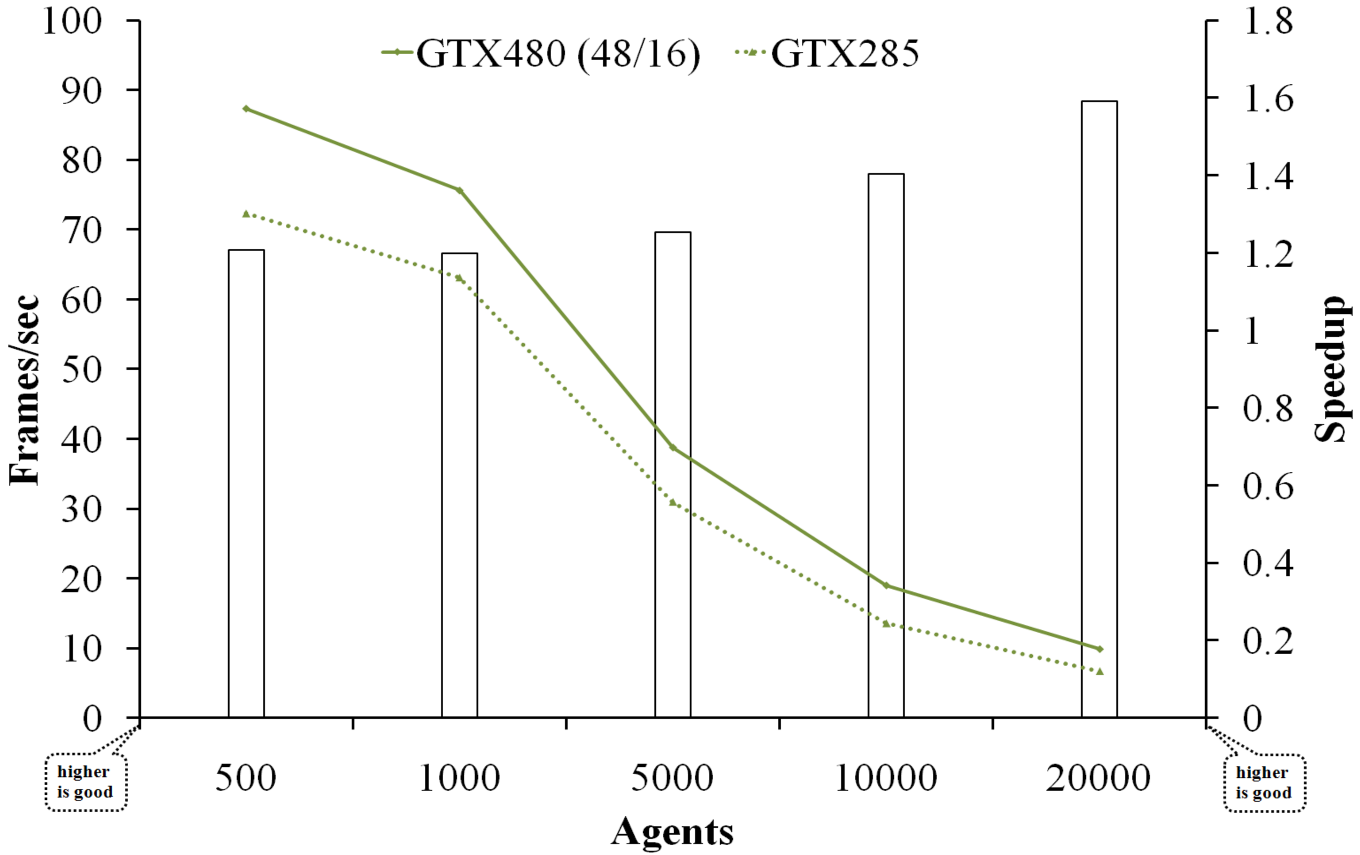


Multi Agent Simulation Experiments

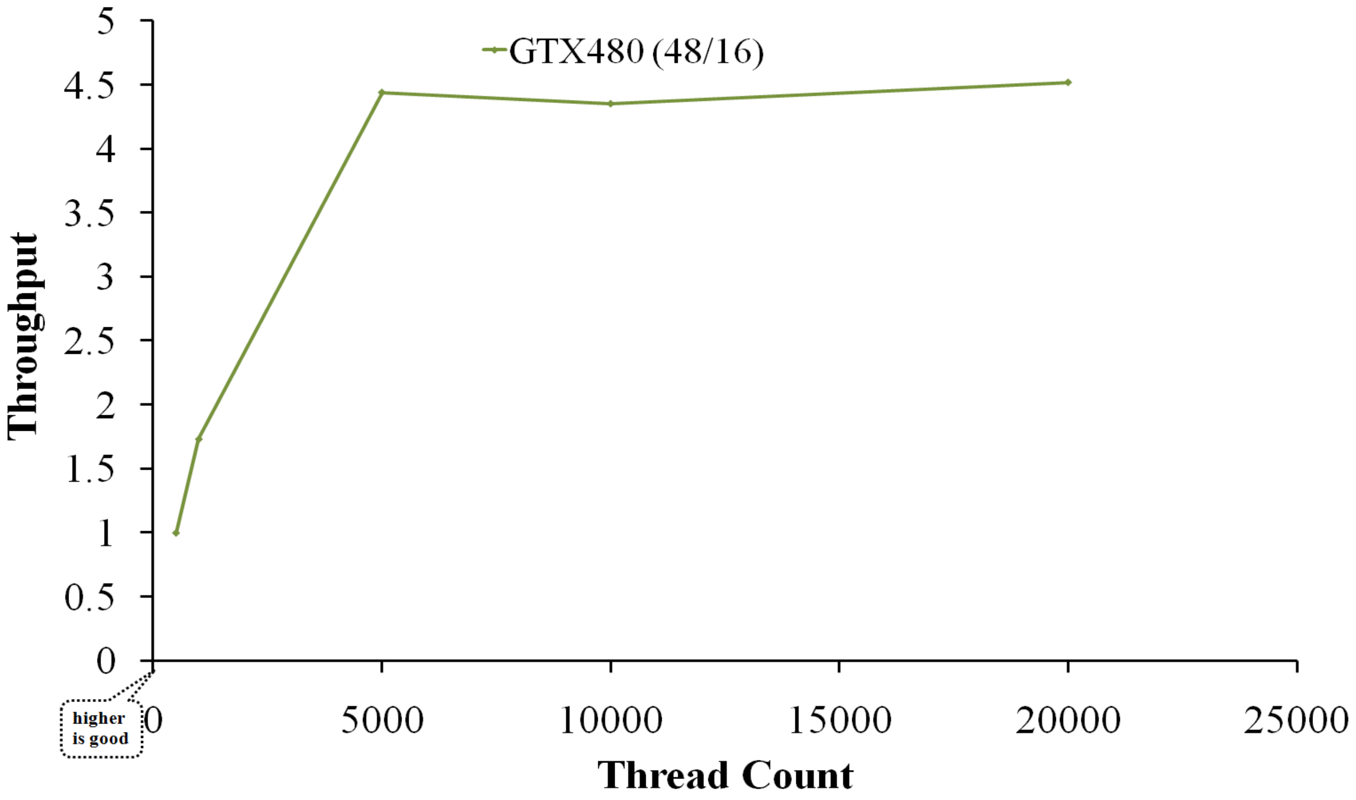
Timestep	Proximity		Velocity Samples	Frames
	Neighbors	Distance		
0.1	10	15	250	1200

Dataset	Segments	Nodes	Agents	CTAs
Evacuation	211	429	500—20000	4—157

Multi Agent Simulation – vs. GTX285

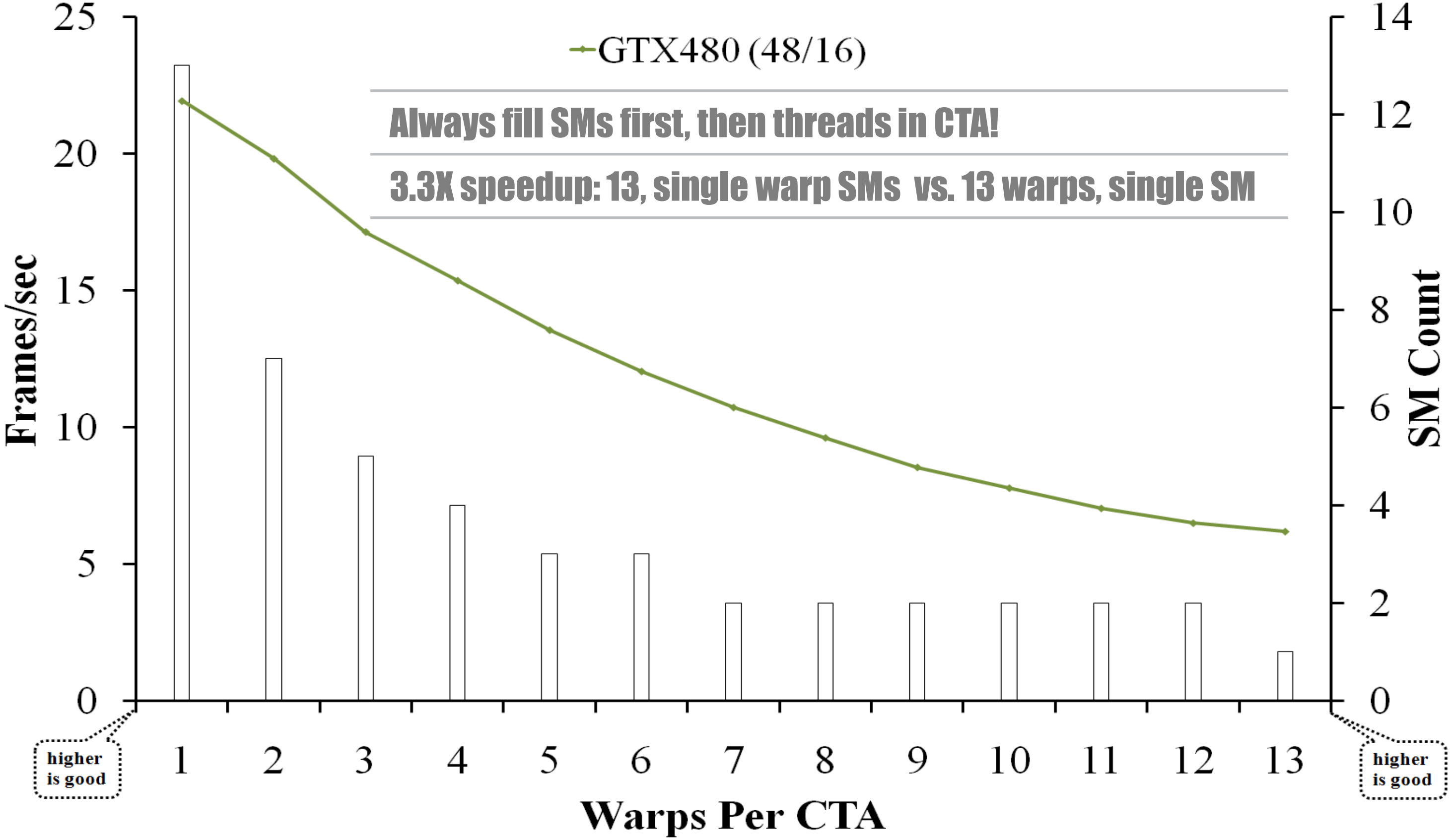


Multi Agent Simulation - Throughput



NVIDIA.

Multi Agent Simulation - Distribution



Limitations

- Flood fill large stack
- A* I/O limited
- One thread, hash build
- Hash under sampling
- Thread load imbalance
  - Non, at-goal agent mix

Fermi Performance

Metric	Roadmap Construction	Path Searching	Multi Agent Simulation
Speedup vs. GTX285 (up to)	2.07X	1.52X	1.59X
Arch Gain vs. GTX285 (%)	91	40	47
Hash vs. Naïve (up to)	NA	NA	4X
Nested vs. Flat (up to)	NA	NA	6.2X

Nested parallel limited to agent count <32

Future Work

- 3D collision avoidance
- Shorter path extractions
- Complex behavior, flocking
- Parallel hash build
- Zero-Copy A*

Summary

- Multi agent solution
  - Compact, scalable
  - Fermi speedup
- Nested parallel potential
- Broad application set

Thank You!

Info

- SDK: foundation libraries, sample applications
  - Technology Preview
- Papers:
  - Scalable Multi Agent Simulation on the GPU, RA09
  - GPU Accelerated Pathfinding, GH08
- Video:
  - Simulation Clips

Appendix

- Compute scale

$$\left(\frac{SMClk_{GTX480}}{SMClk_{GTX285}}\right)*\left(\frac{(Warps/SM)_{GTX480}*SMs_{GTX480}}{(Warps/SM)_{GTX285}*SMs_{GTX285}}\right)$$

- 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

Backup

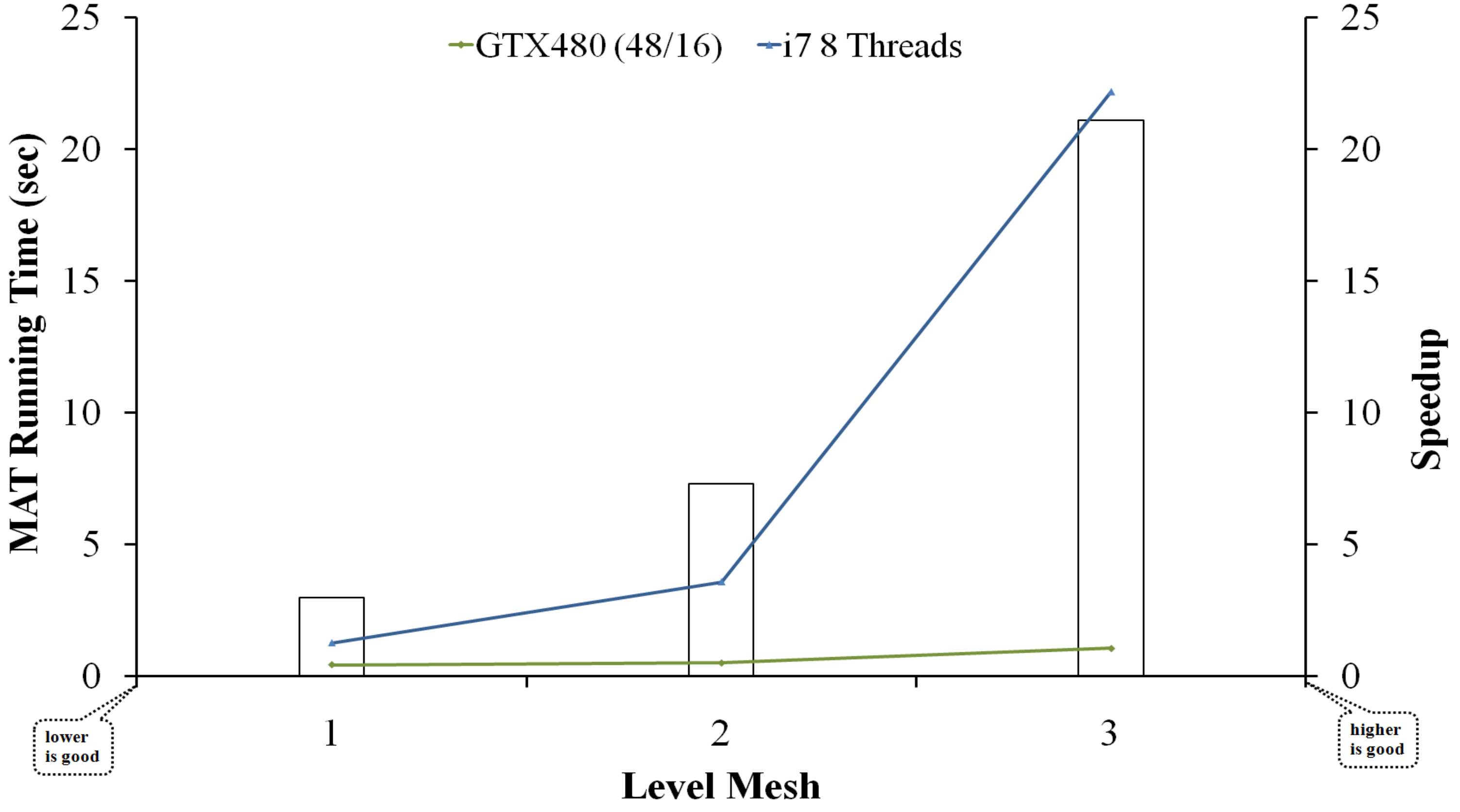
CPU

- Properties

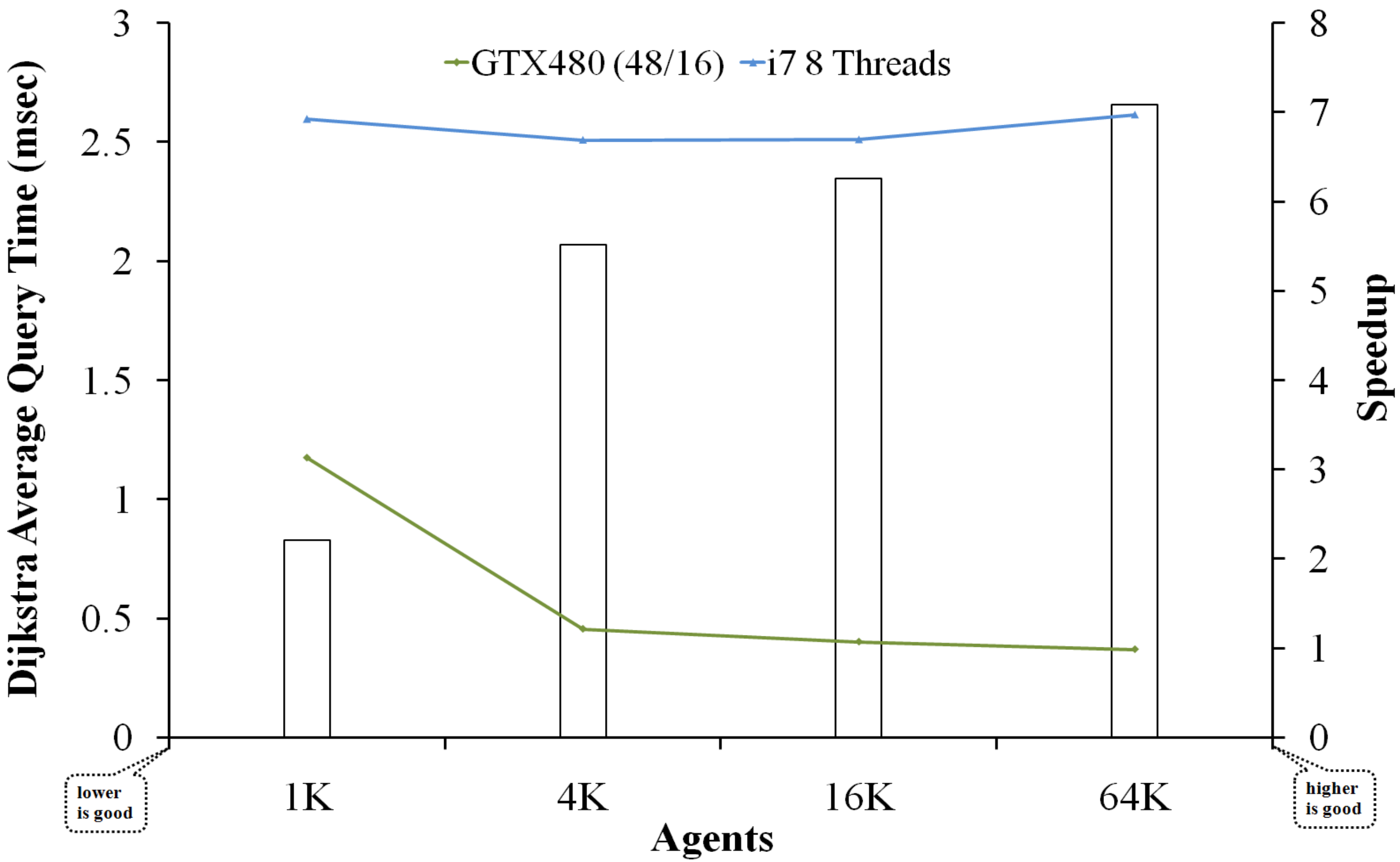
CPU	Cores	Clocks (MHz)	L1/L2 (KB)
Intel i7-940	8	2942/(3*1066)	32/8192
Intel X7350	4	2930/1066	32/8192

- C++ code
  - Not highly optimized
- Multi threading
  - OpenMP, Windows threads

Roadmap Construction - vs. CPU



Path Searching- vs. CPU



average query time = total running time / agent #

Multi Agent Simulation – vs. CPU