GPU Reinforcement Learning for Crowd Simulations

Benjamín Hernández, PhD

Advanced Data and Workflows Group
Oak Ridge Leadership Computing Facility
Oak Ridge National Laboratory
hernandezarb@ornl.gov

Collaborators

Sergio Ruiz, PhD
Tecnologico de Monterrey, Mexico City Campus
sergio.ruiz.loza@itesm.mx

Contents

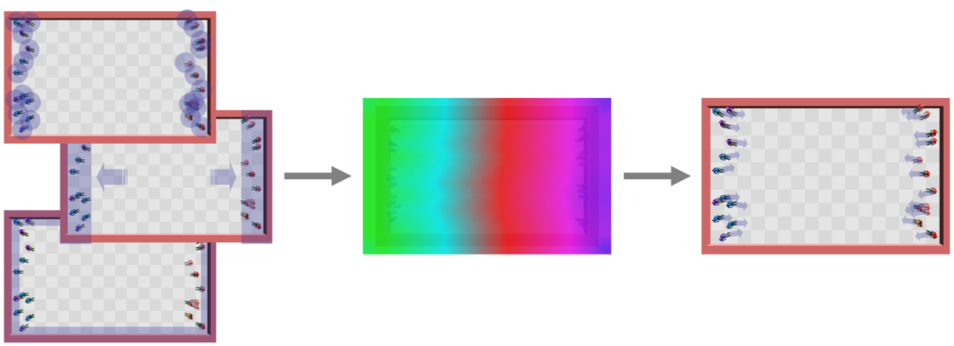
- Introduction
- Crowd behavior models
- Markov Decision Process / Reinforcement Learning in Crowd Simulations
- Parallelization strategy
- Results
- Conclusions

Why a Crowd simulation?

Crowd Behavior Models

Macroscopic

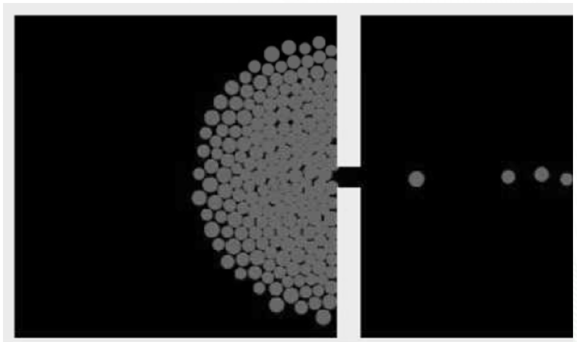
- Describe global interactions with the environment and the crowd itself
- The crowd dynamics are studied as a whole.
- Flow based methods.
- Useful for Navigation



Treuille et al.

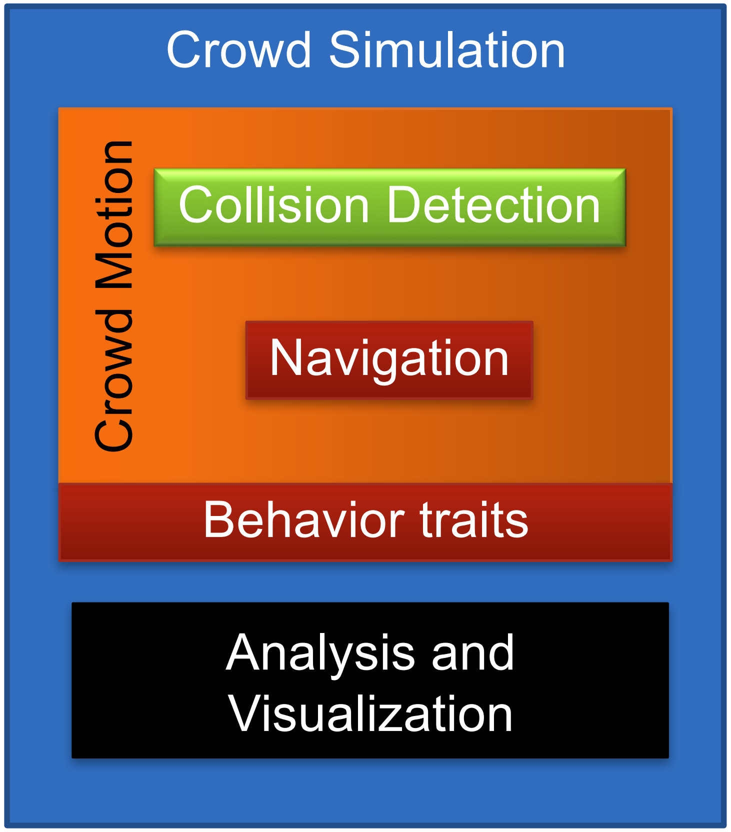
Microscopic

- Exposes the interactions between individuals within a group
- Model the behavior of the individuals in the crowd.
- Agent Based Modeling
- At a large scale Microscopic modeling exposes Macroscopic behavior
- Useful for Collision Detection



Helbing et al.

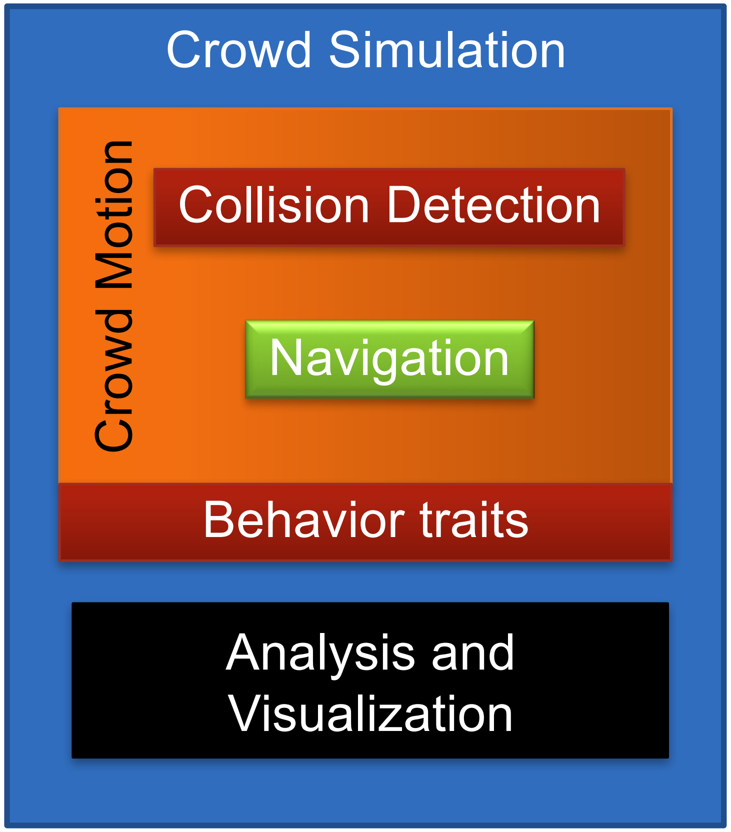
Crowd Behavior Models



Collision Detection

- The ability of agents to effectively avoid collisions against other agents.
- Challenge: It is a **O(N²)**
- Approach: reduce the collision search space.
- Hierarchical Methods, reduces the collision search space by dividing the space in smaller areas, or by defining radius of search around an agent (Reynolds, Bonner and Kelley)
- Rule based make use of Inference Engines, Knowledge Bases and set of rules (Fernandez et al., Li et al.).
- Geometrical Based Methods, uses geometrical primitives (spheres, cones, planes) or projections. Berg et al., Guy et al., Li et al.

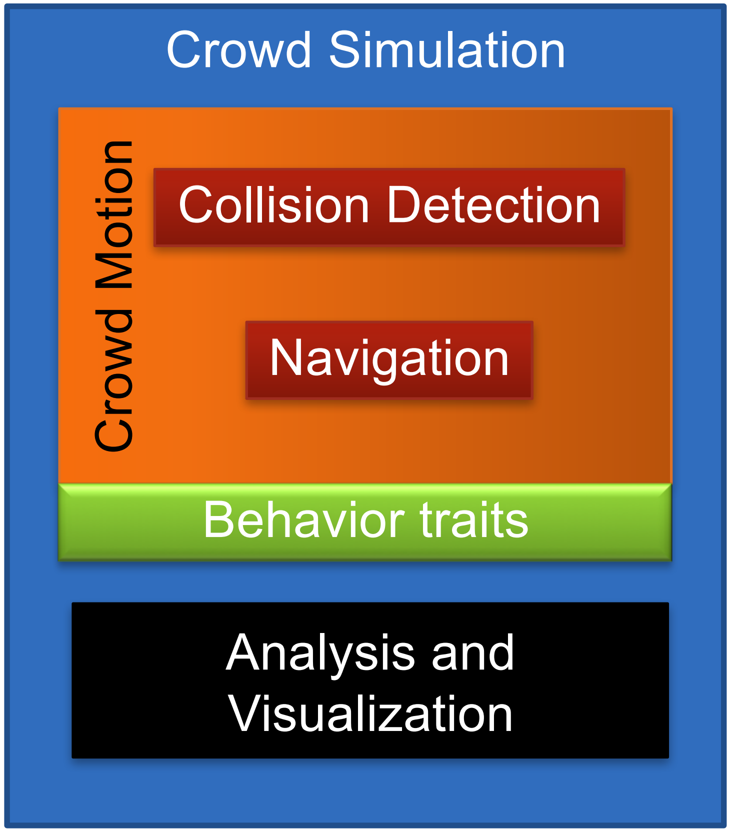
Crowd Behavior Models



Navigation

- The ability of agents to effectively avoid collisions against objects and the environment while moving toward their goal.
- Challenge: how to generate navigation routes free of obstacles efficiently.
- Social Forces (Helbing and Molnar) a collection of attraction or repulsion forces, helps pedestrians to get to its destination and avoid obstacles or other pedestrians.
- Grid Partitioning
  - Cellular Automata
  - Lattice Gas Cellular Automata
  - Vector or Motion Fields
  - Markov Decision Process
- Navigable Areas

Crowd Behavior Models



Behavior Traits

- They define psychological characteristics at individual and crowd level:
  - OCEAN Model Openness, Conscientiousness, Extraversion, Agreeableness, Neuroticism
  - Boids (Reynolds): Separation, Cohesion and Alignment.
  - Thalmann et al. extend Reynolds Observations for Crowd Behavior control: Flocking, Following, Goal Changing, Attraction, Repulsion, Split, Space Adaptability, Safe-Wandering
- Allows the representation of heterogeneous behavior based on psychological observation
- Challenge: How to map psychological features to your crowd model. There is no a general approach.

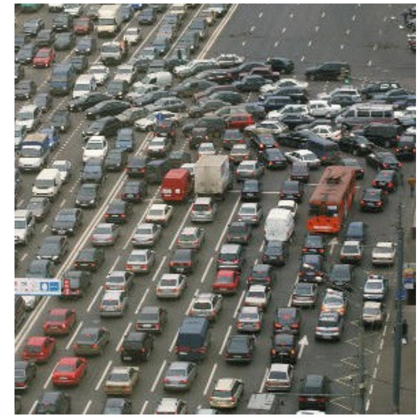
Crowd Simulation and HPC

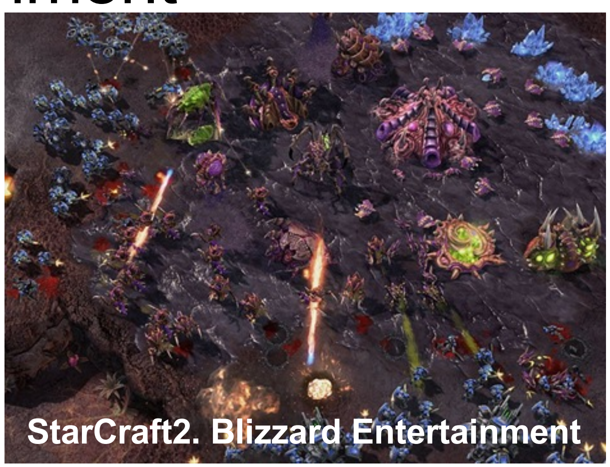
- A crowd model* may demand:
  - More memory (input datasets, internal, auxiliary or temporal data structures, data structure for coupling models).
  - More computational power (more complex models, handling with different models, simulate not just a few thousands of pedestrians but several hundred of thousands of them, analysis and visualization).
  - Interactive supercomputing for fast decision making support

*These models are referred as large scale models.

Applications of Crowd simulation

- Applications
  - Emergency simulations or Special Events
  - Anthropology
  - Training and crowd control
  - Vehicular traffic
  - Public Health (global health security)
  - Entertainment




StarCraft2. Blizzard Entertainment


World War Z. Paramount Pictures

Observation

An agent, while moving through an environment, makes sequential decisions to find a path that goes from its current position to a goal, constructing a set of additive rewards as it gets closer to its goal.

This behavior can be modeled by a Markov Decision Process (MDP).

Markov Decision Process

A Markov Decision Process is a tuple $M = \langle S, A, T, R \rangle$

- S is a finite set of states.
- A is a finite set of actions.
- T is a transition model T(s, a, s').
- R is a reward function R(s).
- a policy π is a solution that specifies what action follows given a state.

Challenges

- Compute Intensive
- Memory Demanding

Modeling Agent Navigation using Markov Decision Process

Assumptions:

- Considering a scenario for which the position of static obstacles is determined prior to the crowd simulation.
- A constrained, fully observable MDP can be evaluated in order to determine optimal navigation directions for groups of agents enclosed in cells.

Modeling Agent Navigation using Markov Decision Process

- Therefore, from the formal definition of a MDP, $M = \langle S, A, T, R \rangle$
  - S (finite set of states) is composed by every cell resulting from partitioning the navigable space.
  - A (finite set of actions) is a set of actions representing an agent's available movement directions, e.g. forward, left, right, etc.
  - T (transition model) is defined by the probabilities of taking a given action.
  - R (reward function) are cells marked as points of interest (high valued rewards), navigable space (medium valued rewards) and obstacles (low valued rewards).

Modeling Agent Navigation using Markov Decision Process

- Then, to determine the optimal navigation directions, we find the optimal policy $\pi_t^*(s)$ i.e. a set of actions that maximizes the reward function *R*

Value Iteration

$$\pi_t^*(s) = argmax_a Q_t(s,a)$$

$$Q_t(s,a) = R(s,a) + \gamma \sum_{j=0}^{7} T_{sj}^a V_{t-1}(j)$$

$$V_t(s) = Q_t\left(s, \pi^*(s)\right);\ V_0(s) = 0$$

A simplified example

Rewards $R(s,a)$

	1	2	3	4
a	-3	-3	-3	+100
b	-3	[redacted]	-3	-100
c	-3	-3	-3	-3

Values $V_0(s)=0$

	1	2	3	4
a	0	0	0	+100
b	0	[redacted]	0	-100
c	0	0	0	0

A = {**E**, **W**, **N** }
γ = 1 (for simplicity)
Transitions:
p = 0.8 (chance of taking the current direction)
q = 0.1 (chance of taking another direction)

$$\pi_t^*(s) = argmax_a Q_t(s,a)$$

$$Q_t(s,a) = R(s,a) + \gamma \sum_{j=0}^{2} T_{sj}^a V_{t-1}(j)$$

What is π for cell a3 ?
$\pi(a3) = \max\{Q(a3,W), Q(a3,N), Q(a3,E)\}$

$Q(a3,\boldsymbol{E}) =$ -3 + 1.0(0.8(100) + 0.1(0) + 0.1(0))
$Q(a3,\boldsymbol{W})$= -3 + 1.0 (0.1(100) + 0.8(0) + 0.1(0))
$Q(a3,\boldsymbol{N}) =$ 0 + 1.0 (0.1(100) + 0.1(0) + 0.8(0))
=> max is $Q(a3,\boldsymbol{E})$

This is one cell !

$$\begin{aligned} Q(a3,\mathbf{E}) &= \boxed{-3} + \boxed{1.0}\,\boxed{(0.8(100) + 0.1(0) + 0.1(0))} \\ Q(a3,\mathbf{W}) &= \boxed{-3} + \boxed{1.0}\,\boxed{(0.1(100) + 0.8(0) + 0.1(0))} \\ Q(a3,\mathbf{N}) &= \boxed{0} + \boxed{1.0}\,\boxed{(0.1(100) + 0.1(0) + 0.8(0))} \end{aligned}$$

$R(s,a)$ γ $\sum_{j=0}^{2} T_{sj}^a V_{t-1}(j)$

Oak Ridge National Laboratory

Parallelization approach

This is one cell !

$$
\begin{aligned}
Q(a3,\mathbf{E}) &= \boxed{-3} + \boxed{1.0}\,\boxed{(0.8(100) + 0.1(0) + 0.1(0))} \\
Q(a3,\mathbf{W}) &= \boxed{-3} + \boxed{1.0}\,\boxed{(0.1(100) + 0.8(0) + 0.1(0))} \\
Q(a3,\mathbf{N}) &= \boxed{0} + \boxed{1.0}\,\boxed{(0.1(100) + 0.1(0) + 0.8(0))}
\end{aligned}
$$

$R(s,a)$ $\quad \gamma \quad$ $\sum_{j=0}^{2} T_{sj}^{a} V_{t-1}(j)$

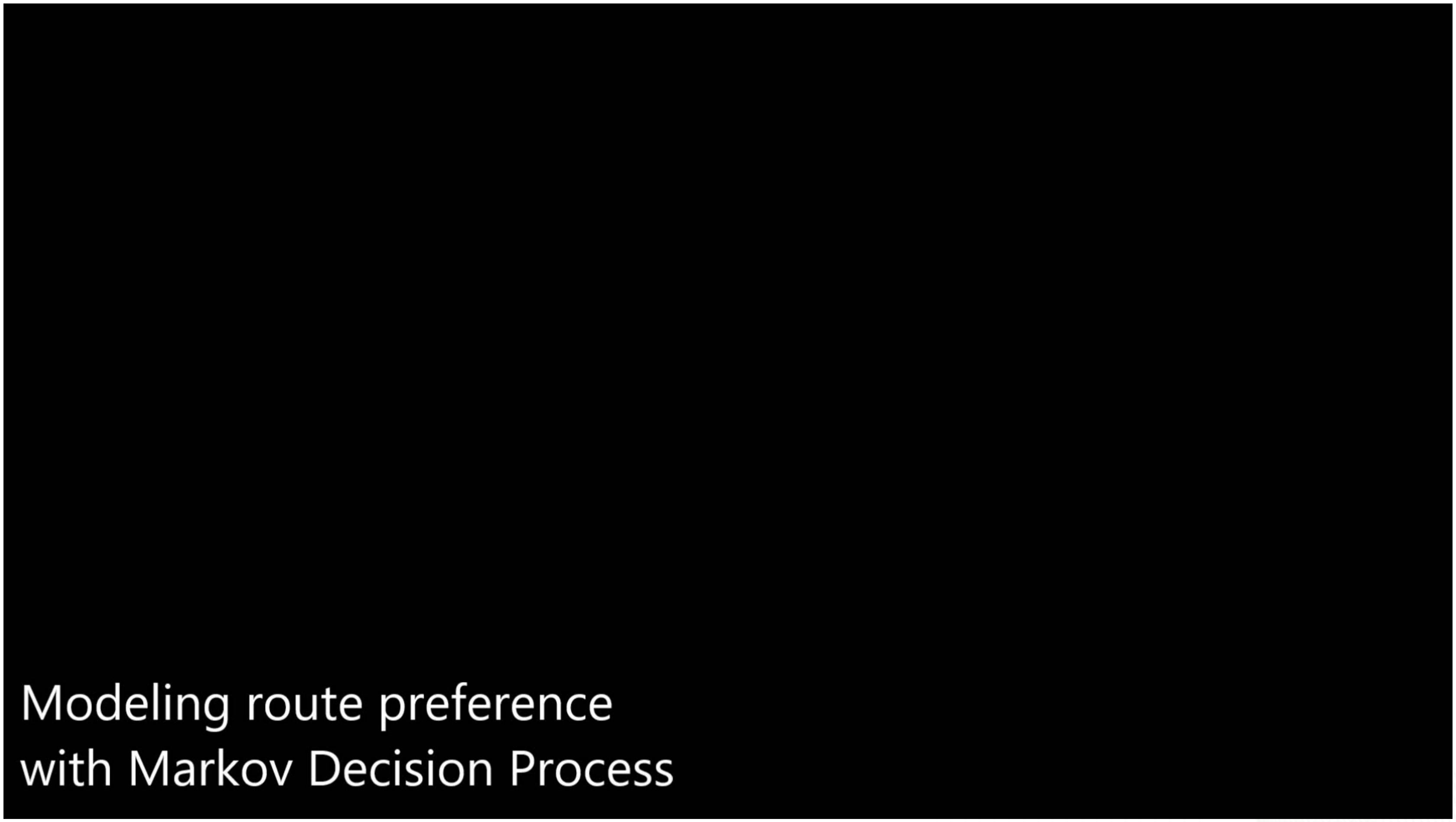
- From this expression note that:
  - For each cell or state, a similar set of equations is to be solved.
  - The set of equations can be solved in parallel, if each cell knows the rewards from neighboring cells and out-of-bounds rewards.
  - Variables p, q, γ and rewards values $R(s,a)$ can be stored in arrays for each cell.
  - Expressions in parenthesis can be solved by parallel reductions.
  - A second parallel reduction using conditionals will solve

$$\pi(a3) = \max\{Q(a3,W), Q(a3,N), Q(a3,E)\}$$

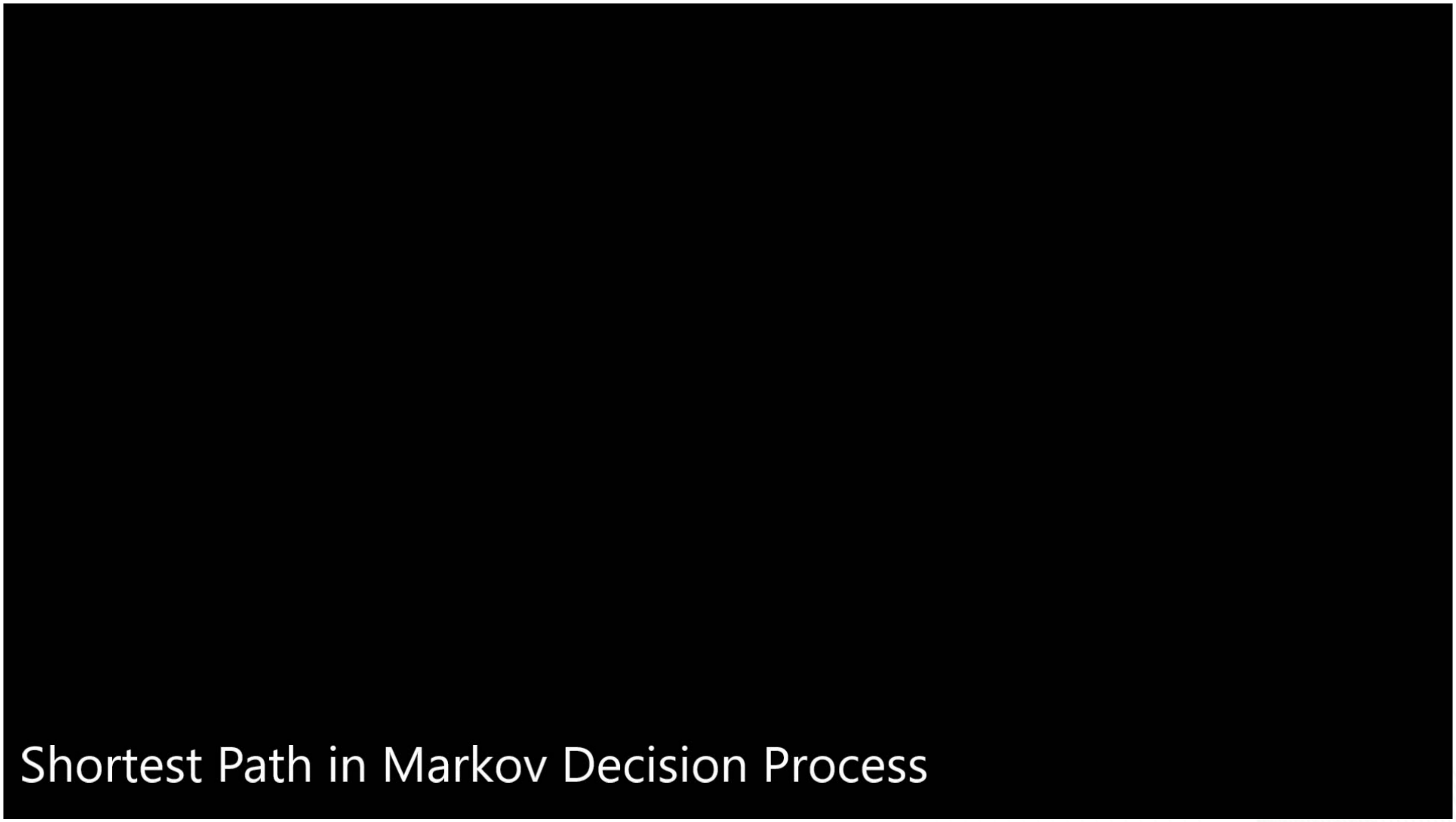
OAK RIDGE National Laboratory

Bottleneck

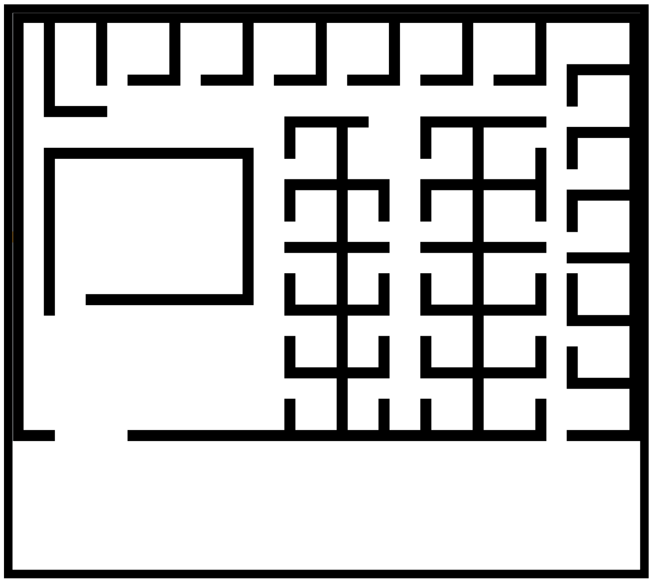
Route Preference



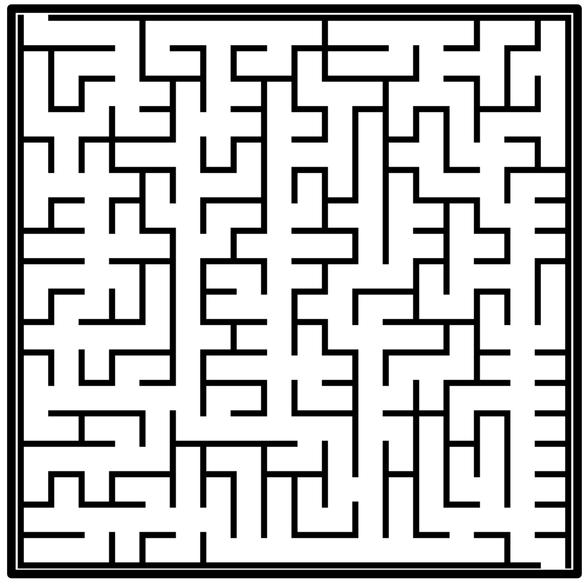
Shortest Path



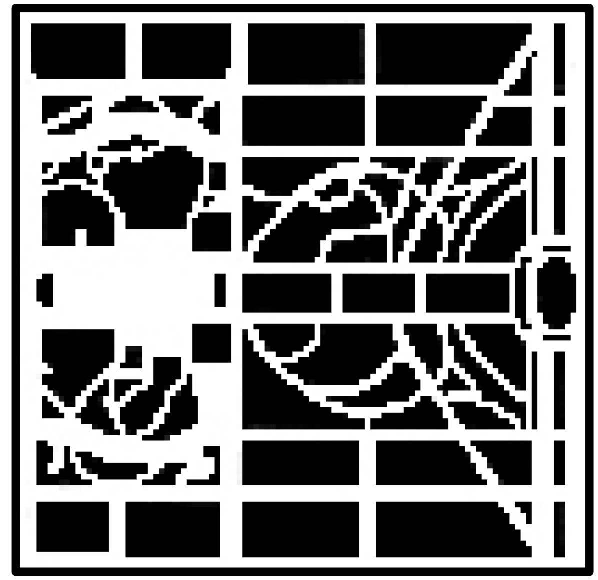
Results: test scenarios



Office (1,584 cells)



Maze (100x100 cells)



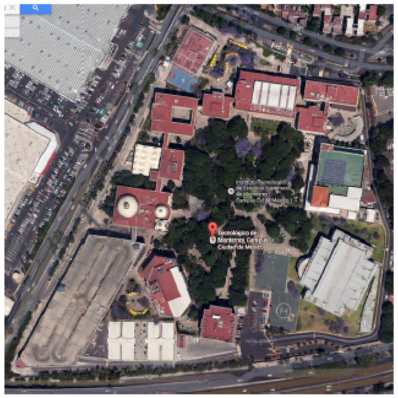
Champ de Mars (100x100 cells)

Implementation: CUDA Thrust, OpenMP and CUDA Backbends

CPU: Intel Core i7 4th Gen CPU running at 3.40GHz.
ARM (Jetson TK1): 32 bit ARM quad-core Cortex-A15 CPU running at 2.32GHz.
GPUs: Tegra K1 192 CUDA Cores, Tesla K40c 2880 CUDA cores, Geforce GTX TITAN 2688 CUDA cores.

OAK RIDGE National Laboratory

Results: test scenarios

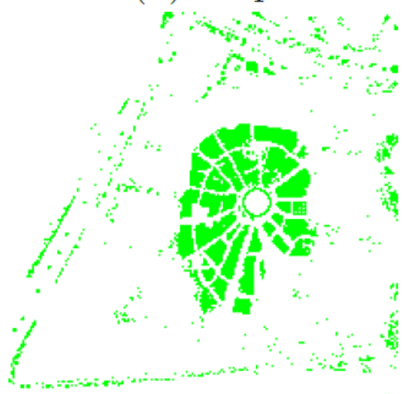


(a) Satellite image

(b) Map

(c) Solid structures

(d) Grass

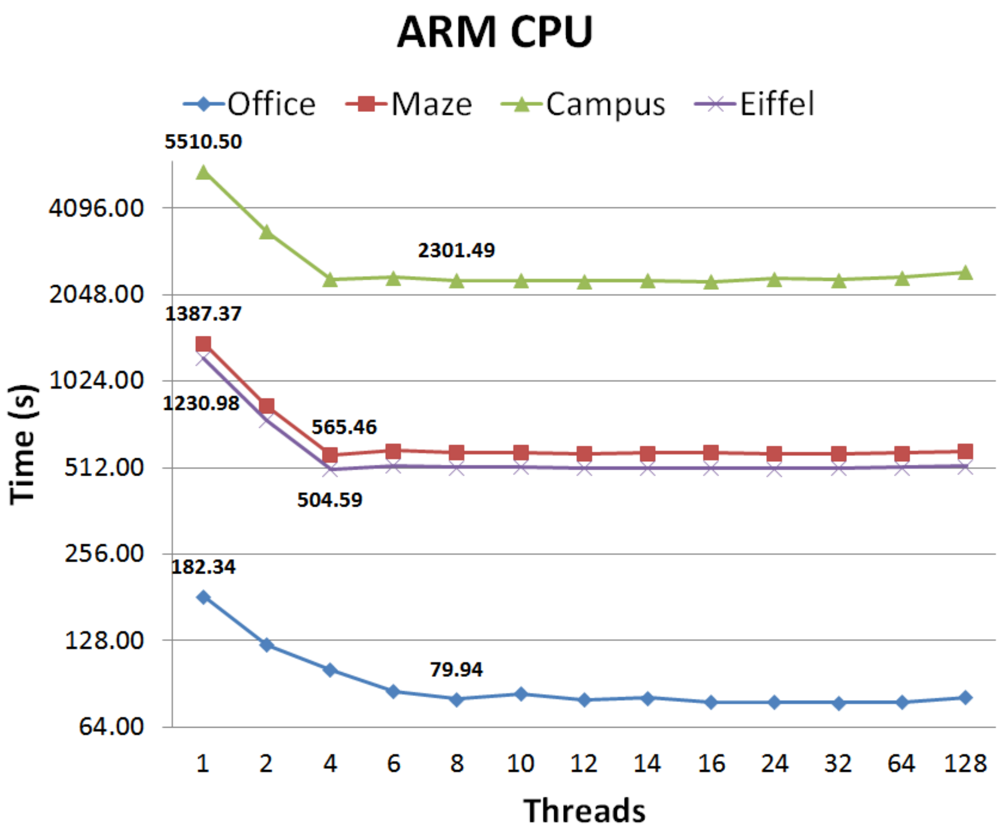


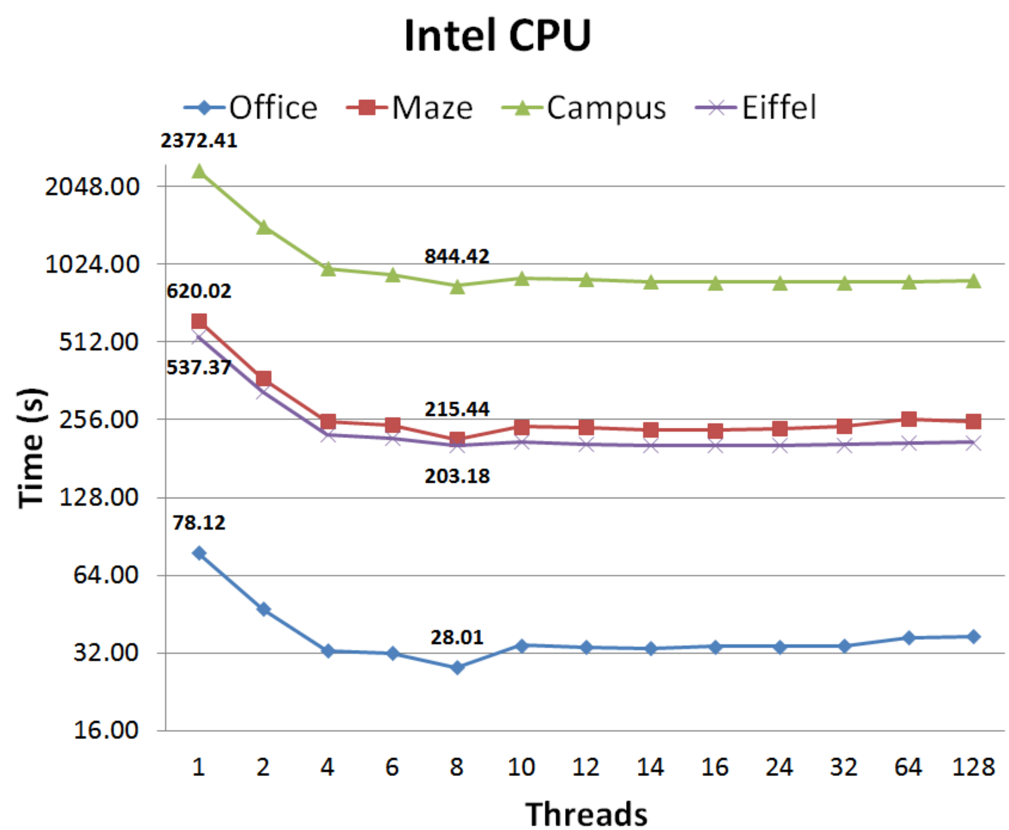
(e) Trees

(f) Combined texture

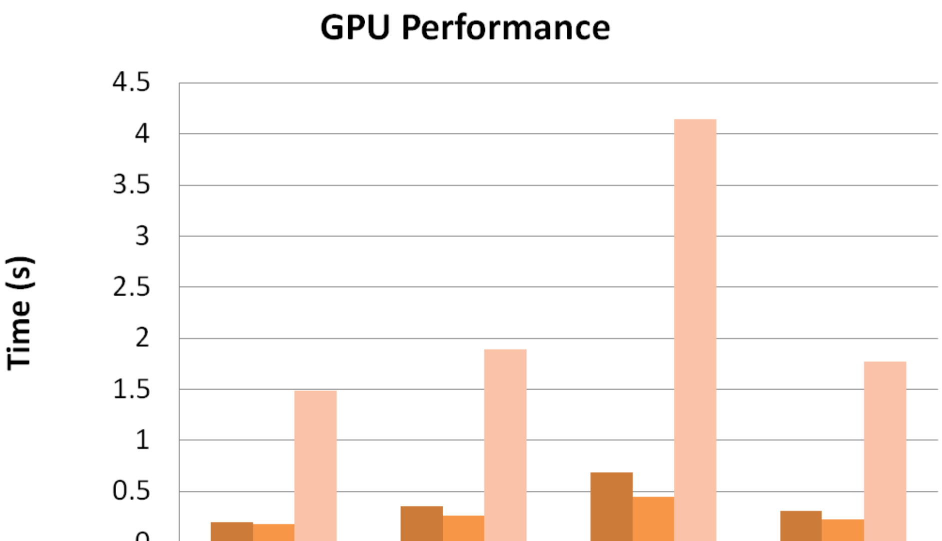
Campus

Results: CPU Performance

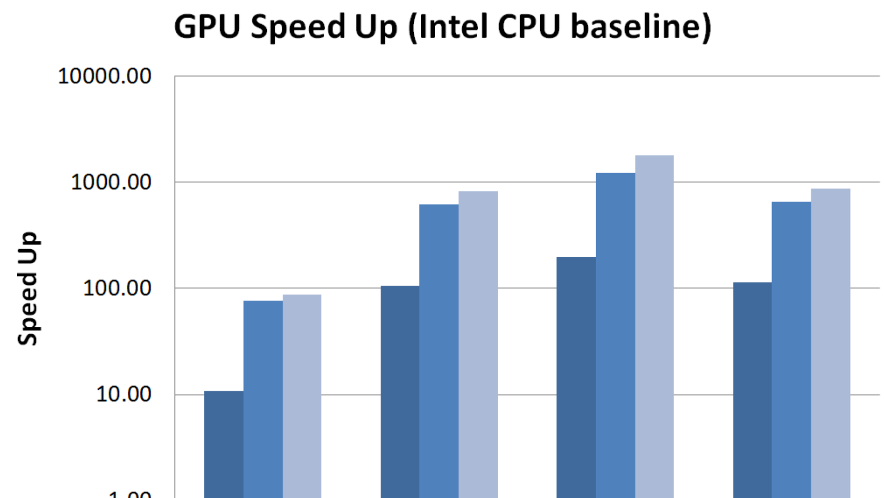




Results: GPU Performance (2015)



	office	maze	campus	eiffel
K40c	0.201112	0.353646	0.688652	0.312195
Titan	0.1794	0.26751	0.450385	0.228
Jetson TK1	1.484512	1.894599	4.14232	1.768087

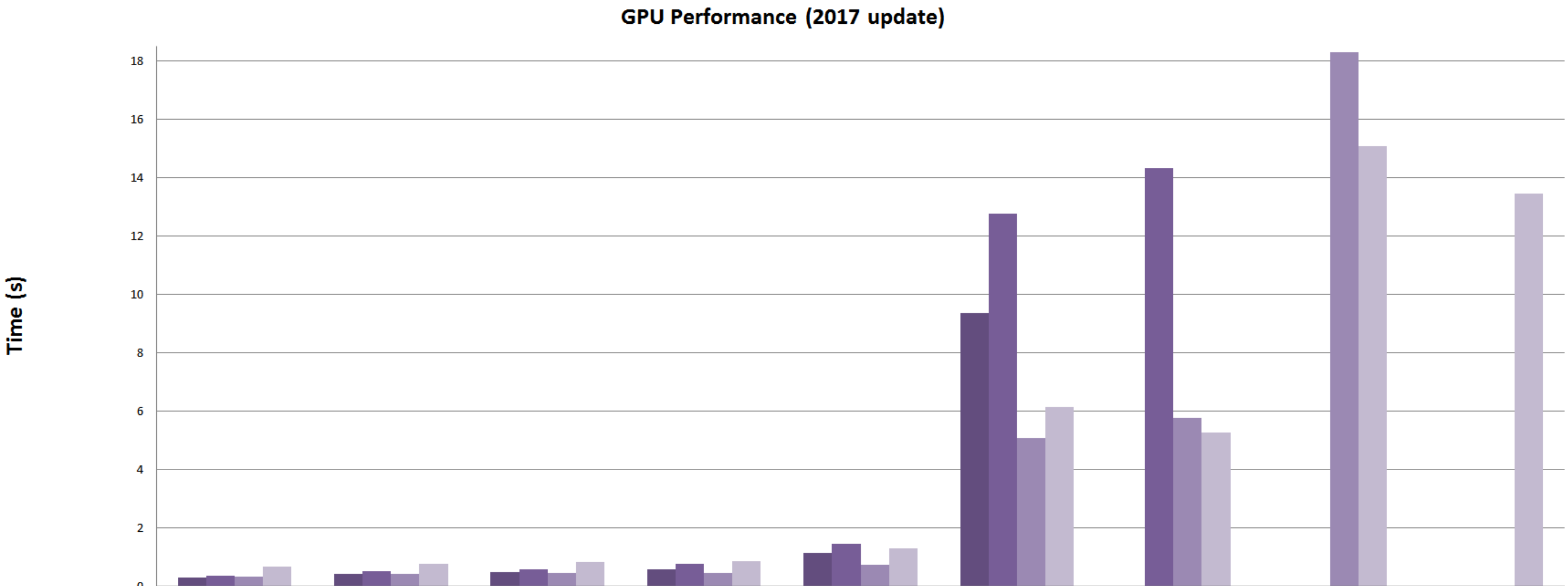


	Office	Maze	Campus	Eiffel
JetsonTK1	10.70	105.97	197.92	113.57
K40c	76.98	620.84	1224.88	656.65
GTX Titan	87.30	820.75	1794.11	876.08

Office 1,584 cells
Maze 100x100 cells
Eiffel 100x100 cells
Campus 200x200 cells

OAK RIDGE National Laboratory

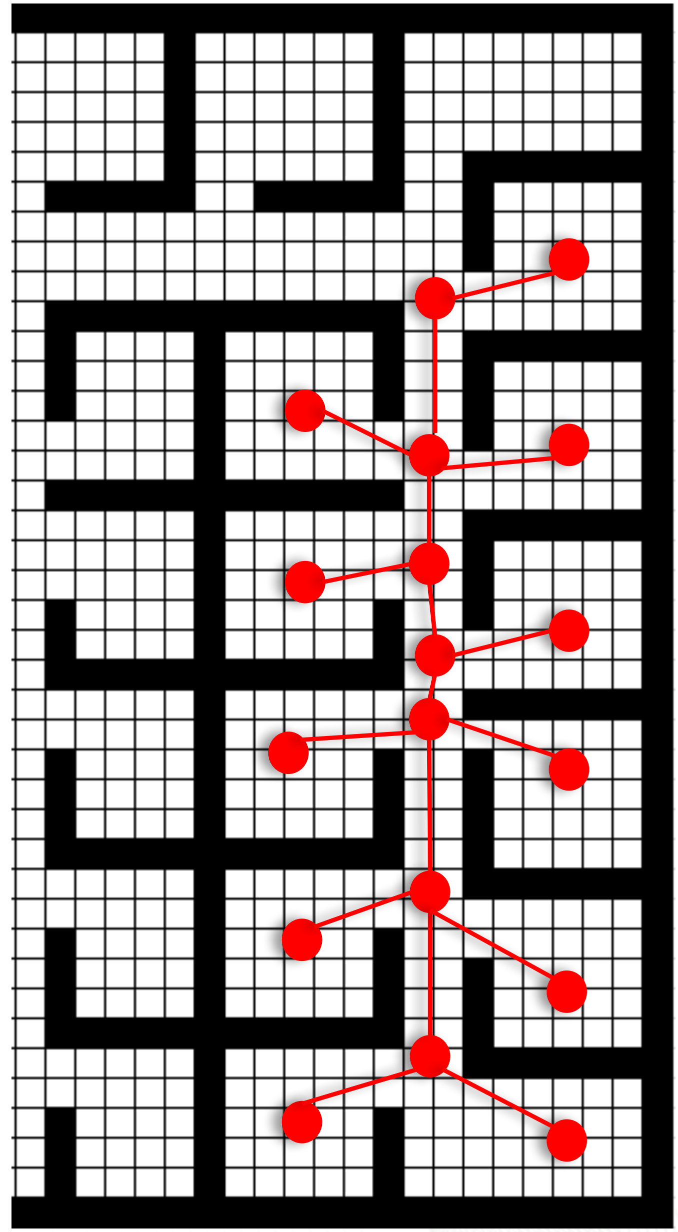
Results: GPU Performance (2017)



	Office 40x40 (188)	eiffel_100x100 (195)	maze 100x100 (211)	campus_200x200 (212)	maze_328x328 (240)	shortestPath_512x512 (985)	maze_1042x1042 (317)	maze_1602x1602 (433)	maze_1922x1922 (276)
780i 3GB	0.291349	0.431602	0.472132	0.585332	1.147594	9.36683	0	0	0
K20 5GB	0.348235	0.526534	0.568637	0.772634	1.450227	12.777888	14.32353	0	0
TitanX-Pascal 12 GB	0.345443	0.430957	0.450674	0.456638	0.751324	5.073929	5.754213	18.293665	0
P100-SXM2 16GB	0.677629	0.766241	0.836798	0.862457	1.298245	6.1598	5.281873	15.092348	13.470682

Conclusions

- MDP is a powerful tool for crowd modeling, but it should be handled with care. Demands
  - Computer power
  - Memory
- The use of GPUs in the Crowd Simulation Domain can deliver faster simulations in situations where massive crowds are expected (Olympic games, Concerts, concentrations around public transportation).



What's next?

- Feed the model with real data.
  - Validation
  - Calibration
- Analysis on the variation of the transition function, discount values and set of actions *A* .
  - Model different kind of agents ?
- Implement higher level behaviors to expose behavior traits
  - MDPs
  - Decision Trees
  - Finite State Machines

OAK RIDGE
National Laboratory

What's next?

- Reinforcement learning. Evaluate different parameter values to obtain policy convergence in the least number of iterations without losing precision in the generated paths.

- Investigate further applications of our MDP solver beyond the context of crowd simulation.

Further Reading

- Sergio Ruiz, Benjamín Hernández. 2017, "Real Time Markov Decision Processes for Crowd Simulation", GPU Zen. Engel W. (Editor), Black Cat Publishing (In preprint).
- Ruiz, S. Hernandez, B. *"A parallel solver for Markov Decision Process in Crowd Simulation"*. MICAI 2015, 14th Mexican International Conference on Artificial Intelligence, At Cuernavaca, Mexico, IEEE proceedings volume.
- Benjamín Hernández, Hugo Perez, Isaac Rudomin, Sergio Ruiz, Oriam deGyves, Leonel Toledo. 2014, "Simulating and Visualizing Real-Time Crowds on GPU Clusters". Computacion y Sistemas, Vol. 18, No. 4, 2014, pp. 651 664. ISSN 2007-9737
- Hugo Perez, Benjamín Hernández, Isaac Rudomín, Eduard Ayguade. "Scaling Crowd Simulations in a GPU Accelerated Cluster". High Performance Computer applications. Eds: Isidoro Gitler, Jaime Klapp. Springer 2016
- Hugo Perez, Benjamín Hernández, Isaac Rudomin. "Task-based Crowd Simulation for Heterogeneous Architectures". Innovative Research and Applications in Next-Generation High Performance Computing. Ed. Qusay F. Hassan. IGI Global. 2016
- Agent-Based Models of Geographical Systems. 2012. Heppenstall, A.J., Crooks, A.T., See, L.M., Batty, M. (Eds.). Springer Netherlands
- Suiping Zhou, Dan Chen, Wentong Cai, Linbo Luo, Malcolm Yoke Hean Low, Feng Tian, Victor Su-Han Tay, Darren Wee Sze Ong, and Benjamin D. Hamilton. 2010. Crowd modeling and simulation technologies. *ACM Trans. Model. Comput. Simul.* 20, 4, Article 20

OAK RIDGE National Laboratory

Crowd Simulation in High Performance Computing Systems

Benjamín Hernández, PhD

Advanced Data and Workflows Group
Oak Ridge National Laboratory
hernandezarb@ornl.gov

Collaborators

Sergio Ruiz, PhD
Tecnologico de Monterrey, Mexico City Campus
sergio.ruiz.loza@itesm.mx

Thank you!

This research was partially supported by: CONACyT SNI-54067, CONACyT PhD scholarship 375247, Nvidia Hardware Grant and Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, under DOE Contract No. DE-AC05-00OR22725.

OAK RIDGE
National Laboratory

Optimization Approaches

- According to (Reyes et al. 2009, Foka and Trahanias 2003), Markov Decision Processes (MDPs) are computationally inefficient: as the state space grows, the problem becomes intractable.
- Decomposition offers the possibility to solve large MDPs (Sucar 2007, Meuleau et al. 1998, Singh and Cohn 1998), either in State Space decomposition, or Process decomposition.
- (Mausam and Weld. 2004) follow the idea of concurrency to solve MDPs generating solutions close to optimal extending the Labeled Real-time Dynamic Programming method.

Optimization Approaches

- (Sucar 2007) proposes a parallel implementation of weakly coupled MDPs.
- (Jóhansson 2009) presents a dynamic programming framework that implements the Value Iteration algorithm to solve MDPs using CUDA.
- (Noer 2013) explores the design and implementation of a point-based Value Iteration algorithm for Partially Observable MDPs (POMDPs) with approximate solutions. The GPU implementation supports belief stat pruning which avoids calculations.