Pathfinding in 3D Space:
A*, Theta*, Lazy Theta* in Octree Structure

Chia-Man Hung, Ruoqi He

CDT Autonomous Intelligent Machines & Systems, Department of Engineering Science, University of Oxford

Introduction

Pathfinding addresses the problem of finding shortest paths from source to destination avoiding obstacles. It is an essential component of Machine Intelligence and finds many applications in the fields of robotics, logistics and video games. In particular, pathfinding in 3D space may be useful for drone navigation and real-time 3D strategy games. There exist different algorithms to solve exact shortest paths on graphs (Dijkstra) or 3D surfaces (exact geodesics). However, finding exact Euclidean shortest paths in three or higher dimensions is an **NP-hard problem**. Standard methods for pathfinding in a 2D plane could be extended to 3D, but they are either inefficient in time or memory.

The **objective** of this study is to find approximate shortest paths efficiently in 3D space with obstacles with reasonable memory consumption.

Screenshot from Homeworld

Methods

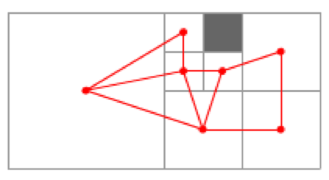
The general **pipeline** of our method is as follows. First, we construct an octree representing the 3D space with obstacles. Second, we construct a graph from the octree. Third, we inject source and destination in the graph. Finally, we apply a graph-based search algorithm to find approximate shortest paths.

Octree construction:

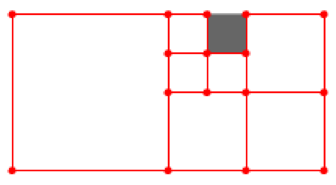
We subdivide the octree recursively wherever the space it represents intersects an obstacle, until a fixed lowest level. Since the obstacles are normally objects represented by triangle meshes, we implement a fast method to detect triangle-cube intersection [1].

A *progressive octree* is an octree with an additional constraint: we require that the difference of levels between neighbouring octree leaves be no more than 1. This constraint can potentially reduce approximation error while applying graph-based pathfinding algorithms.

Graph construction:

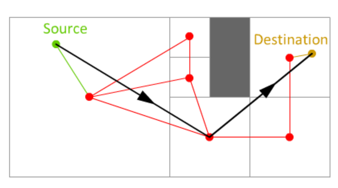


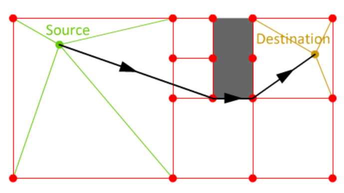
Dual graph



Edge-corner graph

Injection of source and destination:

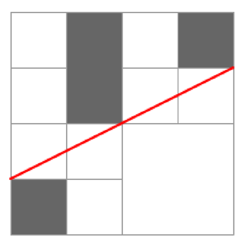


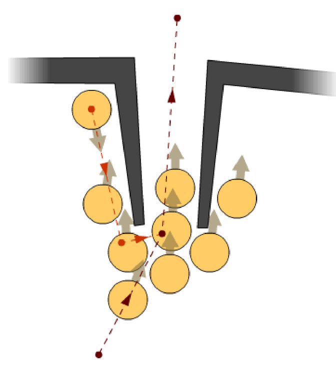


Avoid exhaustive search: precompute connectivity using union-find

Multisource: reuse information

Line of sight: integer-based, robust



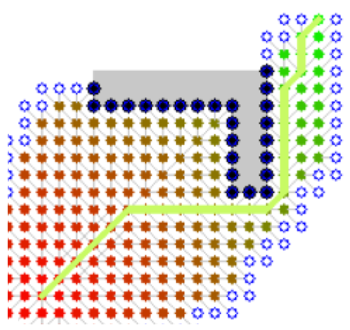


Multi-agent pathfinding: waypoints, repulsive force, replanning

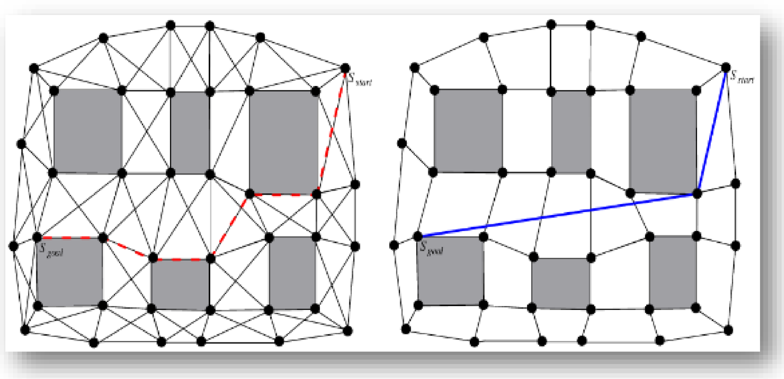
Algorithms

Graph-based search algorithms:

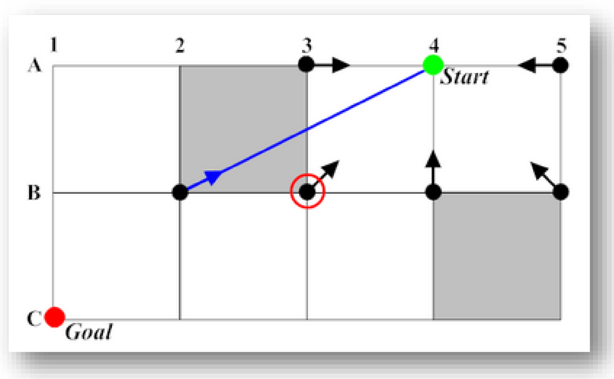
A* (1968 Hart [2]) – generalization of Dijkstra
Theta* (2007 Nash [3]) – line-of-sight check
Lazy Theta* (2010 Nash [4]) – delay of line-of-sight check



A path found by A*



Theta* allows path outside of the edges by performing a line-of-sight check.

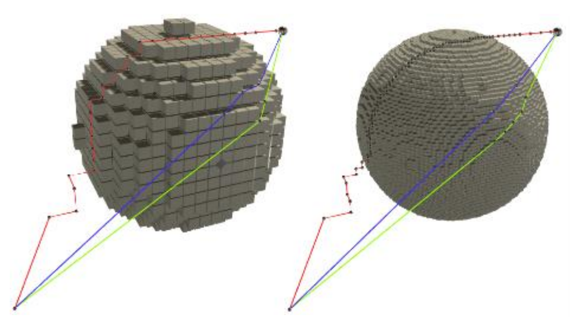


Lazy Theta* checks whether the parent of a node is in its line of sight only before exploring its neighbours.

Results

Comparison of data structures and algorithms:

Red: A*,
Green: Theta*,
Blue: Lazy Theta*

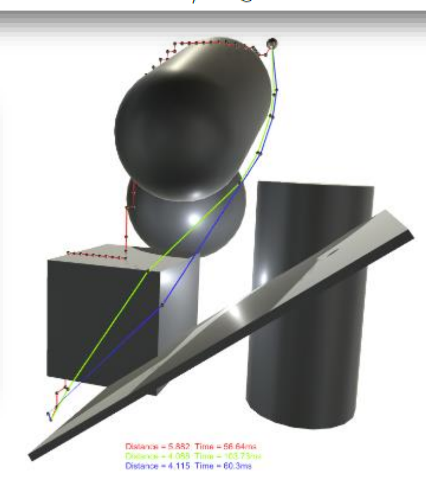


Data Structure	Algorithm	distance	time cost	distance	time cost
Octree	A*	121.38%	1.5ms	121.03%	28.0ms
	Theta*	104.59%	6.7ms	102.71%	236.1ms
	Lazy Theta*	104.65%	4.2ms	102.34%	114.8ms
Progressive Octree	A*	127.43%	3.3ms	126.88%	55.6ms
	Theta*	103.49%	6.3ms	101.23%	229.4ms
	Lazy Theta*	103.68%	3.2ms	101.16%	108.8ms

Table 1: Comparison - A single sphere - Left: level = 7/ Right: level = 9

Data Structure	Algorithm	distance	time cost	distance	time cost
Octree	A*	3.2472	9.5ms	3.3302	5.3ms
	Theta*	2.4108	23.9ms	2.4600	45.5ms
	Lazy Theta*	2.4135	9.5ms	2.4592	16.3ms
Progressive Octree	A*	3.3949	14.1ms	3.3222	7.15ms
	Theta*	2.4009	17.59ms	2.4158	43.1ms
	Lazy Theta*	2.4057	7.78ms	2.4205	14.6ms

Table 2: Comparison - A complex scene - Left: edge-corner graph/ Right: dual graph



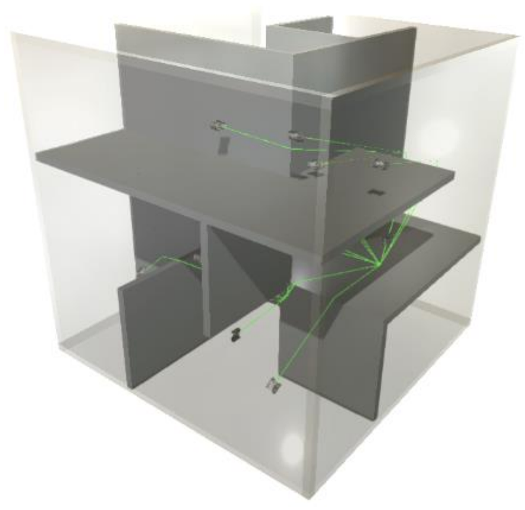
Conclusion

Best combination: Lazy Theta* + progressive octree + edge-corner graph

We proposed methods to find approximate short paths in 3D space efficiently. We achieved short paths in acceptable error (<2%) for reasonable resolution in all of our test cases with known exact solution.

Future works / Possible improvements:

Distributed computation at each frame
Other possible heuristics in A*-family algorithms
Post-processing with local optimisation



For more information:

chiaman@robots.ox.ac.uk

https://ascane.github.io/

Unity demo

References
[1] Tomas Akenine-Möller. Fast 3d triangle-box overlap testing. In *ACM SIGGRAPH 2005 Courses*, page 8. ACM, 2005.
[2] Nilsson Hart and Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems, Science, and Cybernetics*, SSC-4(2):100–107, 1968.
[3] Alex Nash, Kenny Daniel, Sven Koenig, and Ariel Felner. Theta*: Any-angle path planning on grids. *Proceedings of the National Conference on Artificial Intelligence*, 22(2):1177, 2007.
[4] Alex Nash, Sven Koenig, and Craig Tovey. Lazy theta*: Any-angle path planning and path length analysis in 3d. In *Proceedings of the National Conference on Artificial Intelligence*, 2010.