Lazy Theta*: Any-Angle Path Planning and Path Length Analysis in 3D

Alex Nash[*] and **Sven Koenig**
Computer Science Department
University of Southern California
Los Angeles, CA 90089-0781, USA
{anash,skoenig}@usc.edu

Craig Tovey
School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, GA 30332-0205, USA
craig.tovey@isye.gatech.edu

Introduction

We are interested in path planning for robotics and video games. Path planning consists of discretizing a continuous environment into a graph (*generate-graph* problem) and propagating information along the edges of this graph in search of a short path from a given start vertex to a given goal vertex (*find-path* problem) (Wooden 2006). Roboticists and video game developers solve the *generate-graph* problem by discretizing the continuous environment using many different techniques resulting in regular 2D grids composed of squares (square grids), regular 3D grids composed of cubes (cubic grids), visibility graphs and navigation meshes (NavMeshs) (Tozour 2004). They typically solve the *find-path* problem with A* because A* is simple, efficient and guaranteed to find shortest paths formed by graph edges when used with admissible heuristics. A* and other traditional *find-path* algorithms propagate information along graph edges *and* constrain paths to be formed by graph edges. However, shortest paths formed by graph edges are not necessarily equivalent to the truly shortest paths (in the continuous environment). This can be seen in Figure 1 (right), where a continuous environment has been discretized into a NavMesh. Each polygon is either blocked (grey) or unblocked (white). The path found by A* can be seen in Figure 1 (left) while the truly shortest path can be seen in Figure 1 (right). In this paper, we therefore develop sophisticated any-angle *find-path* algorithms that, like A*, propagate information along graph edges (to achieve short runtimes) but, unlike A*, do not constrain paths to be formed by graph edges (to find short “any-angle” paths). The shortest paths formed by the edges of square grids can be up to $\approx 8\%$ longer than the truly shortest paths. We show in this paper that the shortest paths formed by the edges of cubic grids can be $\approx 13\%$ longer than the truly shortest paths, which highlights the need for any-angle *find-path* algorithms in 3D. We therefore extend Theta* (Nash et al. 2007), an existing any-angle *find-path* algorithm, from an algorithm that only applies to square grids to an algorithm that applies to any Euclidean solution of the *generate-graph* problem. However, Theta* performs a line-of-sight check for each unexpanded visible neighbor of each expanded vertex and thus it performs many more line-of-sight checks per expanded vertex on 26-neighbor cubic grids than on 8-neighbor square grids. We therefore introduce Lazy Theta* (Nash, Koenig, and Tovey 2010), a variant of Theta* which uses lazy evaluation to perform only one line-of-sight check per expanded vertex (but with slightly more expanded vertices). We show experimentally that Lazy Theta* finds paths faster than Theta* on cubic grids, with one order of magnitude fewer line-of-sight checks and without an increase in path length.

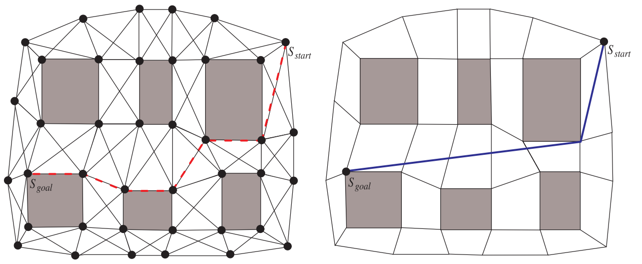

Figure 1: NavMesh: Shortest Path Formed by Graph Edges (left) vs. Truly Shortest Path (right)

Path Planning in 3D

Path planning is more difficult in continuous 3D environments than it is in continuous 2D environments. Truly shortest paths in continuous 2D environments with polygonal obstacles can be found by performing A* searches on visibility graphs. However, truly shortest paths in continuous 3D environments with polyhedral obstacles *cannot* be found by performing A* searches on visibility graphs because the truly shortest paths are not necessarily formed by the edges of visibility graphs. In fact, finding truly shortest paths in continuous 3D environments with polyhedral obstacles is NP-hard (Canny and Reif 1987). Roboticists and video game developers therefore often attempt to find reasonably short paths by performing A* searches on cubic grids. The shortest paths formed by the edges of 8-neighbor square grids can be up to $\approx 8\%$ longer than the truly shortest paths, however we show in this paper that the shortest paths formed by the edges of 26-neighbor cubic grids can be $\approx 13\%$ longer than

[*]This abstract is a short version of a AAAI10 paper. Alex Nash was supported by the Northrop Grumman Corporation. This material is based upon work supported by, or in part by, NSF under contract/grant number 0413196, ARL/ARO under contract/grant number W911NF-08-1-0468 and ONR in form of a MURI under contract/grant number N00014-09-1-1031. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the sponsoring organizations, agencies or the U.S. government.

the truly shortest paths. Thus, neither A* searches on visibility graphs nor A* searches on cubic grids work well in continuous 3D environments. We therefore develop more sophisticated *find-path* algorithms.

Notation and Definitions

Cubic grids are 3D grids composed of (blocked and unblocked) cubic grid cells, whose corners form the set of all vertices V. $s_{start} \in V$ is the start vertex of the search, and $s_{goal} \in V$ is the goal vertex of the search. The visible neighbors of vertex s are the neighbors of s with line-of-sight to s.

A*

Theta* and Lazy Theta* build on A*. To focus its search, A* uses an h-value for every vertex s, that approximates the goal distance of the vertex. A* maintains two values for every vertex s: **(1)** The g-value $g(s)$ is the length of the shortest path from s_{start} to s found so far. **(2)** The parent $parent(s)$ which is used to extract the path after the search halts by following the parent pointers from s_{goal} to s_{start}. A* updates the g-value and parent of an unexpanded visible neighbor s' of a vertex s by considering the path from s_{start} to s and from s to s' in a straight line. It updates the g-value and parent of s' if this new path is shorter than the shortest path from s_{start} to s' found so far.

A* finds shortest paths formed by graph edges when used with admissible heuristics. We proved that the shortest paths formed by the edges of 26-neighbor cubic grids and thus the paths found by A* can be $\approx 13\%$ longer than the truly shortest paths.

Existing Work: Theta*

Theta* is an any-angle *find-path* algorithm that empirically finds shorter paths on square grids than A* with a similar runtime (Nash et al. 2007). The key difference between Theta* and A* is that Theta* allows the parent of a vertex to be any vertex, while A* restricts the parent of a vertex to be a neighbor of that vertex. Theta* is identical to A* except that Theta* updates the g-value and parent of an unexpanded visible neighbor s' of a vertex s by considering two paths: **Path 1:** Like A*, Theta* considers the path from s_{start} to s and from s to s' in a straight line. **Path 2:** To allow for any-angle paths, Theta* also considers the path from s_{start} to $parent(s)$ and from $parent(s)$ to s' in a straight line. It considers Path 2 if s' and $parent(s)$ have line-of-sight since Path 2 is guaranteed to be no longer than Path 1 due to the triangle inequality. Otherwise, it considers Path 1. Theta* updates the g-value and parent of s' if the considered path is shorter than the shortest path from s_{start} to s' found so far.

Extending Theta*: Lazy Theta*

Theta* is slower on cubic grids than square grids because it performs a line-of-sight check for each unexpanded visible neighbor of each expanded vertex. Line-of-sight checks on grids can be implemented efficiently with line drawing algorithms, but the runtime per line-of-sight check can still be linear in the number of cells and there are many more line-of-sight checks per expanded vertex on 26-neighbor cubic grids than on 8-neighbor square grids. We therefore reduce the number of line-of-sight checks that Theta* performs. Our inspiration is provided by PRMs, where lazy evaluation has been used to reduce the number of line-of-sight checks (collision checks) by delaying them until they are absolutely necessary (Bohlin and Kavraki 2000). We therefore introduce Lazy Theta*, a variant of Theta* which uses lazy evaluation to perform only one line-of-sight check per expanded vertex (but with slightly more expanded vertices), while Theta* performs a line-of-sight check for each unexpanded visible neighbor of each expanded vertex.

Lazy Theta*, like Theta*, considers Path 2. However, unlike Theta, Lazy Theta* optimistically assumes that s' and $parent(s)$ have line-of-sight without performing a line-of-sight check. Thus, it delays the line-of-sight check and considers only Path 2. This assumption may be incorrect. Therefore, Lazy Theta* performs the line-of-sight check immediately before expanding vertex s'. If s' and $parent(s')$ indeed have line-of-sight, then the assumption was correct and Lazy Theta* does not change the g-value and parent of s'. If s' and $parent(s')$ do not have line-of-sight, then Lazy Theta* updates the g-value and parent of s' according to Path 1 by considering the path from s_{start} to each expanded visible neighbor s'' of s' and from s'' to s' in a straight line and choosing the shortest such path.

Experimental Results

We now compare the average path lengths, number of vertex expansions, number of line-of-sight checks and runtimes of Theta* and Lazy Theta when finding paths on $100 \times 100 \times 100$ 26-neighbor cubic grids. The start vertex is always the origin $(0, 0, 0)$, and the goal vertex is $(99, y, z)$ for randomly chosen values of y and z. We make the following observations. Path Length: The lengths of the paths found by Lazy Theta* and Theta* are similar to one another and $\approx 8\%$ shorter than the lengths of the paths found by A*. Vertex Expansions: Lazy Theta* expands more vertices than Theta*. Line-of-Sight Checks: Lazy Theta* performs more than one order of magnitude fewer line-of-sight checks than Theta*. Runtime: Lazy Theta* is up to ≈ 1.6 times faster than Theta*.

References

Bohlin, R., and Kavraki, L. 2000. Path planning using lazy PRM. In *Proceedings of the IEEE Transactions on Robotics and Automation*.

Canny, J., and Reif, J. 1987. New lower bound techniques for robot motion planning problems. In *Proceedings of the Symposium on the Foundations of Computer Science*.

Nash, A.; Daniel, K.; Koenig, S.; and Felner, A. 2007. Theta*: Any-angle path planning on grids. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Nash, A.; Koenig, S.; and Tovey, C. 2010. Lazy Theta*: Any-angle path planning and path length analysis in 3D. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Tozour, P. 2004. Search space representations. In Rabin, S., ed., *AI Game Programming Wisdom 2*. Charles River Media. 85–102.

Wooden, D. 2006. *Graph-based Path Planning for Mobile Robots*. Ph.D. Dissertation, Georgia Institute of Technology.