Benchmarks for Pathfinding in 3D Voxel Space

Daniel Brewer
Digital Extremes
London, Ontario, Canada
silentbob.za@gmail.com

Nathan R. Sturtevant
Department of Computer Science
University of Denver
Denver, CO, USA
sturtevant@cs.du.edu

Abstract

The problem of finding optimal paths in 3D space is computationally more complex than in a 2D plane. While a range of different approaches have been developed across different fields, there has been relatively little work studying the complexities and tradeoffs of implementations for path planning in 3D space. This paper describes the 3D path planning problem faced in the game Warframe. The solution used by industry was tuned for their own constraints, but little is known about the applicability of other approaches to solving this problem. Thus, the makers of this game have made their underlying planning data available for researchers to use. This paper describes this new data set which is publicly available for study and dissemination.

Introduction

While optimal path planning in visibility graphs or through discretized 2D grid maps can be done in polynomial time (Welzl 1985), optimal path planning through a three dimensional volume is NP-hard (Canny and Reif 1987). In visibility graphs or grids, optimal paths always pass through the corners of obstacles. But, in three dimensions an optimal path may cross an edges at an arbitrary location and thus may not correspond to the shortest path between two vertices of a finite graph. The location which a path cross an edge can require exponentially many bits to describe (Bajaj 1988). Additionally, the number of distinct edge sequences that need to be considered may be exponentially large.

When faced with a three dimensional planning problem the most common approach is to discretize the world representation in some way (randomly or uniformly) and then plan directly in that representation, restricting paths to only pass through discretize vertices (Yang and Sukkarieh 2008; Colas et al. 2013), as this reduces the overall complexity. Although planning in three dimensional space has been reasonably-well studied in robotics (Bortoff 2000; Petres et al. 2007), this is often in the context of a configuration space that has even higher dimension, and not directly in the three dimensional space.

Thus, in comparison with 2D grid maps, relatively little work has been done on planning directly in a 3D space representation. In particular, it is not clear how the many optimizations for grid maps (Sturtevant et al. 2015) or for road networks (Bast et al. 2016) apply in three dimensional space. One key to answering these questions is the availability of data for testing. This paper describes a new data set of 3D voxel[1] maps that come from the online game Warframe. The data is being made publicly available to encourage further research on 3D path planning that may be applicable not only to games like Warframe, but also to scenarios in augmented and virtual reality. Augmented reality is a particularly compelling future application for 3D path planning, as augmented reality agents must both model and interact with the real world in a believable manner.

This paper describes (1) the game from which the map sets are taken, (2) the origins of the maps within the game, (3) the format of the maps in the benchmark set, and (4) how sample benchmark problems were created. The benchmark problems are developed in a simple discretized environment, but the maps themselves can be used for general shortest path queries in free space. This is one of several future research directions that could build upon this data. Finally, the benchmark sets are available for download from http://www.movingai.com/benchmarks/ along with other previously created benchmark sets for 2D grid maps.

Warframe Description and Constraints

Warframe is an online, sci-fi, action game that originally shipped in March, 2013, although the game is continually updated. One part of the game which was introduced in November, 2014 is the archwing space combat. The archwing gameplay involves flying through procedurally generated debris fields and combating enemy spaceships. These enemy spaceships are required to navigate the debris field with full three dimensional flight paths. The debris fields are constructed procedurally at run-time, so it is unknown what the particular layout will be ahead of time. The debris is made up of various prefabricated objects representing large chunks of destroyed spaceships or asteroids and these objects are highly complex, concave collision meshes. It is important for the enemy ships to be able to navigate through these complex pieces of debris as they need to be able to

[1] voxels are *volume elements*, the 3D equivalent of pixels (*picture elements*)

follow and engage the player in these areas, and not just fly around them.

The overall size of the procedural levels can be up to a maximum volume of 2km×2km×2km and the density of the debris is non-uniform and very sparse. There are no dynamic changes to the map after it has been generated. There are typically between four and twenty enemy spaceships alive at any time that require paths. Before the enemy spaceships engage with the player, they fly in formation on patrol routes. After combat has started, the enemy spaceships select destinations within a set distance range from their target. As the player is almost always in constant motion, both paths and target destinations are recomputed for each agent approximately twice every second. Paths need to be found as fast as possible, with target ideal times of less than 1ms, and worst case of 100ms. Typical hardware requirements are similar to an XBOX ONE: 8GB Ram with a 8 Core 1.75 GHz CPU.

Existing Path Planning Approach

A complete description of the path planning approach in Warframe has been published elsewhere (Brewer 2017), but we provide a brief overview of the approach here. The approach uses sparse voxel octrees (Schwarz and Seidel 2010) to represent the free and passable space. The work is analogous to two-dimensional planning on quadtrees (Yahja et al. 1998) with planning performed between different levels of the octree. That is, a path will contain larger and smaller voxels and different levels in the octree, with special data structures designed to make these connections efficient. Note that this contrasts with alternate approaches that find paths in higher levels of abstraction and then refine them (Sturtevant and Jansen 2007). Overall the approach is optimized to jointly minimize memory and the time required for finding suboptimal paths.

It is important to note that the shipped approach in the game is optimizing three different metrics. It is easy to improve any one of these three metrics at the cost of the other two, but all three must be balanced in practice to be feasible.

Map Types

The data set contains 44 maps that range from 1.5 million to 500 million voxel cells. The majority of these maps are taken from the in-game generator, however two maps (simple and complex) are not gameplay maps. These maps are smaller than the other maps and were designed for testing purposes.

A description of the maps is found in Table 1, and pictures and the dimensions of each map can be found in Table 2. The data in Table 1 describes both how the maps are created (top) and how they are used in the game (bottom). A majority of the maps are composed of large debris fields where the players navigate, although different fields have different features. The images in Table 2 are a flat projection of the x/y plane, with the color determined by the relative z-depth, with the darkest pixels farthest away and the lighter pixels closest.

The game features other types of maps, such maps with gameplay primarily taking place inside a narrow trench, but these are not currently included in the official map set due to technical limitations. (Gameplay is restricted to certain playable spaces both by map and physics constraints, meaning that it is significantly more complex to reason about physics constraints when exporting maps.)

3D Benchmark Format

For the purpose of simplicity, a very simple text format has been used to represent the voxel maps. The file contains the dimensions of each of the x/y/z coordinates, followed by a list of voxels that are filled. While this makes the text significantly larger than a compact or binary representation, this ensures that the files will be easy to read and parse by those wishing to use the benchmarks in any language or application.

```
voxel 105 132 105
50 50 50
50 50 51
50 50 52
...
```

Figure 1: Map format

Sample data from the *Simple* map is shown in Figure 1. The first line indicates that the x coordinates range from $0 \ldots 104$, y coordinates range from $0 \ldots 131$, and the z coordinates range from $0 \ldots 104$. The first three blocked voxels are shown, with coordinates of $\{50, 50, 50\}$, $\{50, 50, 51\}$ and $\{50, 50, 52\}$ respectively.

In practice, a low-level bitwise representation can easily be created from the data, but this type of representation has two significant drawbacks that made it infeasible for use in Warframe. First, the amount of memory required would be too large. As a majority of the space in the maps is empty, it isn't efficient to represent this memory directly. Second, in three dimensional maps the locality of data in cache/memory is significantly reduced, since a simple action can move in three different dimensions of space. Thus, a good representation will also work to improve the cache locality of access. Warframe represents the data in a sparse voxel octree, and uses a Morton (or z-order) encoding to improve spatial locality (Morton 1966).

Sample Problem Instances

There are multiple problem definitions that can be used for path planning in the voxel maps provided. The most general definition would allow for arbitrary movement between free space, subject to the constraints of obstacles. As this problem is NP-hard (Canny and Reif 1987), we instead choose the simplest problem definition for our sample problem instances. This state space is defined by integer $\{x, y, z\}$ coordinates, where each coordinate corresponds to the center of a single voxel. The legal actions are based on movement to the 26 adjacent locations where the absolute change in each of the x/y/z coordinates is at most 1. Diagonal actions (where two or more coordinates change their value) are only allowed if each of the individual cardinal actions, where only a single coordinate changes, are also possible. That is, the

Table 1: Description of maps in the data set.

Map	Count	Description
Simple	1	A single pillar.
Complex	1	A single, large piece of debris from a destroyed spaceship. The debris is quite complex - a concave piece of hull with several holes and ruins of bulkheads and decking.
A*	5	Simple linear random debris field. The debris ranges from pieces of destroyed spaceship (a single piece is in Complex map) up to large asteroids with space stations attached.
C*	4	Debris field surrounding massive Fomorian capitol ship. The Fomorian is hollow and players are required to fly inside to destroy its power generators.
B*, D*, E*, F*	33	Denser debris fields generated along curved splines through space. The letters denote the procedural template used to generate the maps. The numbers denote different sample permutations of the generated maps.
A*, B*, D*	20	Exterminate missions, where the player is required to navigate the debris field and eliminate all enemies.
E*, F*	18	Interception missions, where players are required to capture and hold a number of positions in the map while waves of enemies try to recapture those positions.

representation assumes an agent which occupies a full voxel cell and thus cannot cut through the corners of obstacles.

In this problem definition we can define a simple memory-free heuristic between arbitrary locations called the *voxel heuristic*. The voxel heuristic that we used is the generalization of the two-dimensional octile distance heuristic. Given two points which are distance x and y apart, the octile distance heuristic for a two dimensional plane with 8-connected movement is $h_{octile} = \min(x, y) * (\sqrt{2} - 1) + \max(x, y)$.

The voxel heuristic for three-dimensional free space on a 26-connected grid is similar, although slightly more complex to compute. Given two points which are distance x_Δ, y_Δ, and z_Δ apart in each of the three dimensions respectively, let $d_{max} = \max(x_\Delta, y_\Delta, z_\Delta)$, $d_{min} = \min(x_\Delta, y_\Delta, z_\Delta)$ and $d_{mid} = \{x_\Delta, y_\Delta, z_\Delta\} \setminus \{d_{max}, d_{min}\}$. (This assumes that x_Δ, y_Δ, and z_Δ are distinct.) Then, the voxel grid heuristic $h_{voxel} = (\sqrt{3} - \sqrt{2})d_{min} + (\sqrt{2} - 1)d_{mid} + d_{max}$.

The easiest way to generate problem instances on a map is to generate random start and goal locations and to then find paths between them. However, in the voxel maps in this repository there is often significant amounts of empty space in the full voxel world, meaning that direct paths are possible. Additionally, players (and the AI) will be vulnerable in open space, so these areas are typically avoided. Finally, the regions inside obstacles are not stored, so random pathfinding requests may result in unsolvable queries.

Thus, to model more realistic problem instances, we followed the following procedure. First, we find all free voxels that were within 5 voxels of an obstacle. Then, we selected random points from this set of free voxels, and found the optimal solution between them with the assumption of 26-connected movement on the voxel grid. If there is no feasible solution (because one point is inside an obstacle) or if the voxel heuristic was perfect, the problem was discarded. This procedure was repeated until 10,000 unique paths were found on each map, although individual start and goal locations are not necessarily unique.

```
version 1
Complex.3dmap
101 104 104 143 65 71 71.66073800 1.044
103 106 93 118 58 98 58.73059289 1.052
117 65 71 120 90 96 44.21927406 1.218
...
```

Figure 2: Map instances

The resulting problem instances are demonstrated in Figure 2. The first line is the version number for the problem instance file format. The second line is the name of the map. The following lines contain the starting x/y/z coordinates of the start and the goal followed by the optimal distance between them and the ratio between the optimal distance and the heuristic distance. The first problem is from $\{101, 104, 104\}$ to $\{143, 65, 71\}$ and has length 71.66073800 while the shortest voxel distance, also the heuristic distance, would be 68.64295804, with a ratio of 1.044. The ratio is a rough proxy for the difficulty of a problem, although difficulty is also impacted by other factors such as path length and the number of reachable states.

Research on Path Planning in Voxel Spaces

In this section we point out the work we are aware of that studies approaches for simple 3D pathfinding in voxel spaces – where there are no constraints other than the obstacles in the environment. First, Carsten, Ferguson, and Stentz (2006) showed how to adapt Field D* to 3D maps. Next, Nash, Koenig, and Tovey (2010) analyzed the suboptimal-

ity of 3D grid movement and showed that Theta* variants could decrease the suboptimality of 3D planning. Finally, Wardhana, Johan, and Seah (2013) applied abstraction and refinement techniques to three dimensional free space.

Conclusions

There has been relatively little work on 3D path planning in simple voxel spaces. Part of this can be attributed to the lack of good test data on which to test algorithms; a problem which this data set remedies. Some areas for future work include but are not limited to:

- **Research on map representations**. This includes work on representations that are both memory efficient and query efficient for the queries performed in a shortest path request.
- **Research on adapting known grid techniques**. This work would analyze existing approaches and see how they apply or can be adapted to three dimensional space.
- **Research on shortest path techniques**. While the discretized 3D pathfinding problem is polynomial and the problem of finding true shortest paths is NP-hard, it is unclear how hard it is to find true shortest paths on the Warframe instances. Although the map representation is discretized, this doesn't prevent a path planning approach from finding true shortest paths that deviate from the voxel grid.
- **Research on new techniques**. It is often possible to make more specific assumptions about a fixed data set than in general. Thus, there may be new optimizations that can be discovered for this data set that drive the field forward.

Clearly there is significant room for new research in this area. We hope that this data set spurs new innovate research on the problem of 3D pathfinding.

References

Bajaj, C. 1988. The algebraic degree of geometric optimization problems. *Discrete and Computational Geometry* 3:177–191.

Bast, H.; Delling, D.; Goldberg, A.; Müller-Hannemann, M.; Pajor, T.; Sanders, P.; Wagner, D.; and Werneck, R. F. 2016. *Route Planning in Transportation Networks*. Cham: Springer International Publishing. 19–80.

Bortoff, S. A. 2000. Path planning for uavs. In *Proceedings of the 2000 American Control Conference. ACC (IEEE Cat. No.00CH36334)*, volume 1, 364–368 vol.1.

Brewer, D. 2017. 3d flight navigation using sparse voxel octrees. In *Game AI Pro 3: Collected Wisdom of Game AI Professionals*. CRC Press.

Canny, J. F., and Reif, J. H. 1987. New lower bound techniques for robot motion planning problems. In *28th Annual Symposium on Foundations of Computer Science, Los Angeles, California, USA, 27-29 October 1987*, 49–60.

Carsten, J.; Ferguson, D.; and Stentz, A. 2006. 3d field D: improved path planning and replanning in three dimensions. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2006, October 9-15, 2006, Beijing, China*, 3381–3386.

Colas, F.; Mahesh, S.; Pomerleau, F.; Liu, M.; and Siegwart, R. 2013. 3d path planning and execution for search and rescue ground robots. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, November 3-7, 2013*, 722–727.

Morton, G. M. 1966. *A computer oriented geodetic data base and a new technique in file sequencing*. International Business Machines Company.

Nash, A.; Koenig, S.; and Tovey, C. A. 2010. Lazy theta*: Any-angle path planning and path length analysis in 3d. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*.

Petres, C.; Pailhas, Y.; Patron, P.; Petillot, Y.; Evans, J.; and Lane, D. 2007. Path planning for autonomous underwater vehicles. *IEEE Transactions on Robotics* 23(2):331–341.

Schwarz, M., and Seidel, H.-P. 2010. Fast parallel surface and solid voxelization on GPUs. *ACM Transactions on Graphics* 29(6 (Proceedings of SIGGRAPH Asia 2010)):179:1–179:9.

Sturtevant, N., and Jansen, M. 2007. An analysis of map-based abstraction and refinement. In *SARA*, 344–358.

Sturtevant, N. R.; Traish, J.; Tulip, J.; Uras, T.; Koenig, S.; Strasser, B.; Botea, A.; Harabor, D.; and Rabin, S. 2015. The grid-based path planning competition: 2014 entries and results. In *Eighth Annual Symposium on Combinatorial Search*, 241–251.

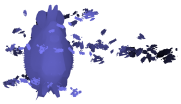
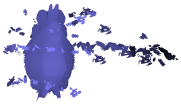
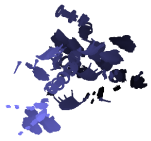
Wardhana, N. M.; Johan, H.; and Seah, H. S. 2013. Enhanced waypoint graph for surface and volumetric path planning in virtual worlds. *The Visual Computer* 29(10):1051–1062.

Welzl, E. 1985. Constructing the visibility graph for n-line segments in o(n2) time. *Information Processing Letters* 20(4):167 – 171.

Yahja, A.; Stentz, A.; Singh, S.; and Brumitt, B. L. 1998. Framed-quadtree path planning for mobile robots operating in sparse environments. In *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*, volume 1, 650–655 vol.1.

Yang, K., and Sukkarieh, S. 2008. 3d smooth path planning for a UAV in cluttered natural environments. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, September 22-26, 2008, Acropolis Convention Center, Nice, France*, 794–800.

Table 2: Maps in the Warframe Test Set

 A1 896 × 390 × 255	 A2 896 × 390 × 255	 A3 896 × 390 × 255	 A4 896 × 390 × 255	 A5 896 × 390 × 255	 BA1 745 × 527 × 615
 BA2 756 × 527 × 617	 BA3 788 × 527 × 604	 BB1 662 × 549 × 662	 BB2 662 × 549 × 680	 BC1 853 × 245 × 321	 BC2 853 × 245 × 342
 C1 1108 × 647 × 723	 C2 1093 × 647 × 725	 C3 1021 × 647 × 633	 C4 1029 × 647 × 645	 Complex 246 × 154 × 205	 DA1 389 × 313 × 511
 DA2 377 × 314 × 535	 DA3 386 × 331 × 517	 DB1 409 × 261 × 462	 DB2 418 × 260 × 466	 DC1 396 × 344 × 533	 DC2 375 × 354 × 554
 DC3 375 × 351 × 563	 EB1 445 × 306 × 617	 EB2 465 × 321 × 617	 EC1 500 × 312 × 562	 EC2 468 × 314 × 565	 EC3 480 × 306 × 562
 ED1 485 × 294 × 616	 EE1 541 × 288 × 591	 EE2 476 × 288 × 591	 FA1 770 × 466 × 564	 FA2 826 × 493 × 574	 FB1 812 × 358 × 578
 FB2 831 × 364 × 525	 FC1 568 × 331 × 868	 FC2 497 × 342 × 868	 Full1 659 × 549 × 624	Full2 851 × 288 × 326	Full3 659 × 569 × 675
Full4 834 × 360 × 553	Simple 105 × 132 × 105				