Pacific Graphics (2015)
N. J. Mitra, J. Stam, and K. Xu (Editors)

Short Paper

Accelerating Graph-based Path Planning Through Waypoint Clustering

N. M. Wardhana[1,2], H. Johan[3], and H. S. Seah[1,2]

[1]Multi-plAtform Game Innovation Centre, Nanyang Technological University, Singapore
[2]School of Computer Engineering, Nanyang Technological University, Singapore
[3]Fraunhofer IDM@NTU, Nanyang Technological University, Singapore

Abstract

Modern Computer Graphics applications commonly feature very large virtual environments and diverse characters which perform different kinds of motions. To accelerate path planning in such scenario, we propose subregion graph data structure. It consists of subregions, which are clusters of locally connected waypoints inside a region, as well as their connectivities. We also present a fast algorithm to automatically generate subregion graph from enhanced waypoint graph map representation, which also supports various motion types and can be created from large virtual environments. Nevertheless, subregion graph can also be generated from any graph-based map representation. Our experiments showed that subregion graph is very compact relative to the input waypoint graph. By firstly planning subregion path, and then limiting waypoint-level planning to the subregion path, up to 8 times average speedup can be achieved, while average length ratios are maintained at as low as 102.5%.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Geometric algorithms, languages, and systems I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

1. Introduction

Since the turn of this century, various Computer Graphics (CG) applications have increasingly featured very large virtual environments, which are occupied by diverse characters with different motion types. This situation poses a challenge how to perform path planning in a timely manner. Solving path planning problems usually involves creating a graph-based structure called *map representation* to approximate the virtual environment. Recently, Wardhana et al.'s proposed *enhanced waypoint graph* [WJS13], which supports various motion types that can be automatically generated given a very large virtual environment. As will be briefly introduced in Section 3, different from a navigation mesh, the use of waypoint graph makes it possible for characters to perform "free-flying" motion, that is not restricted to be on surfaces. However, the authors make use of A* algorithm directly on the very large generated waypoint graph, which is consequently impractical.

In Section 4, we present *subregion graph*, a path planning acceleration structure that consists of *subregions*, which are clusters of locally connected waypoints inside one region, and *subregion connectivities*. By planning *subregion path* between two points, only waypoints inside the subregion path need to be considered, thus reducing the number of visited waypoints and consequently accelerating path planning. We present an algorithm to automatically generate the subregion graph from an enhanced waypoint graph, taking into account different motion types, in Section 5. We will nevertheless show that our algorithm works for any kind of graph-based map representation. A two-step path planning algorithm to use the subregion graph for path planning is presented in Section 6. In Section 7, we report the outcomes of some experiments to evaluate the performance and the result of both our generation and path planning algorithms.

Our contributions can be summarised as follows.

- Extension to enhanced waypoint graph by adding support for adhesive motion type.
- Subregion graph, a compact data structure to accelerate path planning process with various motion types in very large virtual environments.

DOI: 10.2312/pg.20151282

- A fast method to automatically generate a subregion graph, given an enhanced waypoint graph and a set of motion types.
- A method to generate subregion graph from any graph-based map representation.

2. Related Work

There are three main categories of abstraction-based techniques to accelerate path planning, namely Node-centred, Subdivision-based, and Node/Edge-Importance-based techniques. Node-centred abstraction techniques select some nodes from the graph, and then expands from those nodes by following certain rules to create abstractions. In STAR abstraction [HMZM96], vertices are grouped within a certain distance from a particular vertex into one abstraction. On the other hand, Partial-Refinement A* (PRA*) [SB05] constraints the groups to a maximum of four vertices as a hierarchical structure is being built.

In subdivision-based abstraction category, various structures are explicitly determined to subdivide the original graph and separate the abstractions. In *multiway separator* [Fre87], the constraints are pre-selected *boundary nodes*. *Geometric containers* [WW03] and *arc-flag* [Lau04] use an overlaid 2D grid on the input graph to limit the subdivision, whereas HTAP [MH04] uses Voronoi subdivision.

Node/edge-importance-based abstraction techniques modify the Dijkstra/A* algorithm by filtering the nodes or edges it considers based on their importance with respect to shortest paths between every pair of nodes. Gutman uses the concept of *reach* of a vertex, which is high if the vertex is in the shortest paths that are long, and low otherwise. During the node expansion step, only vertices with high reach values are expanded [Gut04]. In *highway hierarchies* technique [SS05], only "highways" or important edges need to be considered outside a certain radius from the starting and goal nodes.

One example of non-abstraction class of acceleration techniques is the look-up table preprocessing. Included in this category are techniques such as Floyd-Warshall algorithm [Flo62, War62], A*-Landmark-Triangle-inequality (ALT) [GH05], and True-Distance Heuristics (TDH) [FBSS09]. In the preprocessing step, information tables are created as guidance in either directly determining paths, or as heuristic values, thus reducing the computation of the actual path planning step.

3. Review of Enhanced Waypoint Graph and Our Extension

A regular waypoint graph, also known as *roadmap*, contains *waypoints*, which describe important features in the environment such as corners and openings, and the waypoint connectivities. Enhanced waypoint graph [WJS13] augments

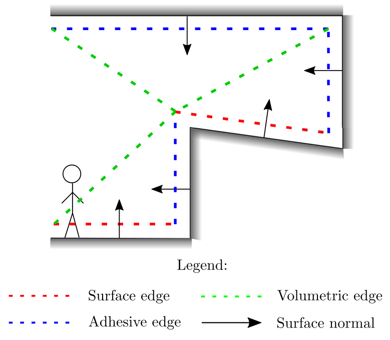

Figure 1: *Edges that support different motion types*

this structure via its capability to handle characters with different sizes and motion types. Every edge in the enhanced waypoint graph is labelled with a motion type, so that during path planning, only edges that support the character's movement are traversed. Wardhana et al.'s paper defines two motion types, namely surface and volumetric motions. They respectively correspond to movement on ground-like surfaces, and free-flying movements.

Enhanced waypoint graph's generation algorithm makes use of a 3D uniform grid containing axis-aligned boxes called *regions* to process very large environments. Waypoints are accordingly labelled as a *local waypoint*, if a waypoint is inside a region, or a *border waypoint*, if it is located at the boundary between two regions.

Our extension: In this paper, we add another motion type called *adhesive motion*. It still needs to be performed on a surface, but the surface can be of any orientation, excluding nearly horizontal surfaces. An edge is labelled with adhesive motion if it is entirely located near a surface whose normal neither points upward nor nearly upward. Animals like spiders and lizards are examples of characters that can perform adhesive motion. Fig. 1 illustrates the various motion types.

4. Proposed Subregion Graph and Terminologies

The idea behind our subregion graph, as depicted in Fig. 2, is to cluster waypoints in every region of enhanced waypoint graph based on their connectivity, and represent them as one abstraction node. Given a motion type set $M_C \subseteq M$, where M denotes a set containing all motion types (in our case, $M \equiv \{surface, adhesive, volumetric\}$), a group of waypoints in one abstraction must be visitable from each other without going to another abstraction by only considering edges that support any motion $m \in M_C$. One region may contain multiple disconnected waypoint sets, so it can have multiple subregions. Subregion graph also explains the abstraction connectivities, and will then be used to filter the waypoints so that not too many waypoints are visited during path planning, thus accelerating the process. A subre-

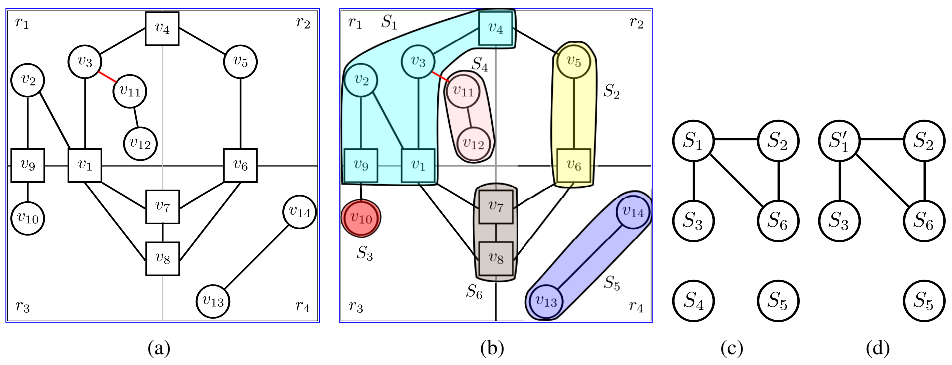

Figure 2: *(a) An example of a 2D enhanced waypoint graph, consisting of 6 regions, 15 waypoints, 15 edges, and 2 motion types. Circles and squares are respectively local and border waypoints, whereas black and red edges respectively represent surface and adhesive motions. (b) The subregions, generated with respect to only surface motion, are overlaid with different colours on the waypoints. (c) The subregion graph, also generated with respect to only surface motion, describes the corresponding subregions with respect to the waypoint graph, and the connection between the subregions. Circles and edges respectively denote subregions and subregion connectivities. (d) A different subregion graph generated with respect to both surface and adhesive motions. Notice that instead of S_1, it has $S_1' = S_1 \cup S_4$, and the original S_4 is non-existent.*

gion graph will be constructed for every motion type set in the application.

To define a subregion graph, some terminologies need to be introduced. Firstly, a waypoint v is *locally visitable* from another waypoint w with respect to a motion type set $M_C \subseteq M$ and region r if w can be visited from v by traversing only waypoints in r and edges that support motion types in M_C. Then, a *subregion* S is a set of waypoints such that two unique waypoints in S are locally visitable from each other with respect to motion type set M_C. Two subregions S_1 and S_2 on subregion graph G_S are connected if there is an edge $\langle v, w\rangle$ such that waypoint v is in S_1, waypoint w is in S_2, and the edge supports a motion type in M_C. To every waypoint v, we associate a *label* $L(v)$, indicating the subregion containing v. A *subregion graph* can finally be defined as a graph $G_S = \langle V_S, E_S\rangle$ whose vertex set V_S contains all subregions, and edge set E_S describes the connectivity among the subregions with respect to motion type set M_C.

5. Automatic Generation of Subregion Graph

Our algorithm to automatically generate a subregion graph, takes two input data, namely an enhanced waypoint graph G_W and a character's motion type set $M_C \subseteq M$. The algorithm will return a subregion graph G_S and, for every waypoint v, a label $L(v)$, which is the subregion containing the waypoint. All waypoints are initially unlabelled. The generation algorithm itself consists of two steps, namely *Subregion Detection* and *Subregion Connection*.

In the Subregion Detection step, we firstly initialise a variable $index = 0$. The algorithm visits an unlabelled starting waypoint (chosen randomly, with all local waypoints chosen first before border waypoints), and performs Breadth First Search to visit unlabelled waypoints without going out of the *active region*, and only traversing edges which support motion type in M_C. An active region is the region containing the waypoint, if it is a local waypoint, or one of the two regions the waypoint borders, if it is a border waypoint. Whenever a waypoint v is visited, we set the label $L(v) = index$. Once no more waypoint can be visited, *index* is incremented, a new unlabelled waypoint is selected, and the process is repeated until all the waypoints are labelled.

In the Subregion Connection step, the algorithm visits every edge which supports any motion in M_C, and check the two waypoints v and w at its end. Their respective labels $L(v)$ and $L(w)$ are then examined. If they are different and not yet connected, the connection is established.

To assign costs to subregion connectivities, we compared a few schemes in the experiment. Two most outstanding cost schemes are Fixed Cost (FC), in which the cost is fixed to value 1, and Centroid Distance (CD), which uses the distance between the centroids of two subregions as a cost. A centroid is the average of the waypoint locations in the subregion.

If the input waypoint graph is generated using another technique, the regional subdivision structure can be trivially constructed and associated to the waypoint, and we can continue with rest of the algorithm. It should be noted that if there is an edge that spans across multiple regions, no boundary waypoints need to be generated, as the subregion graph generation algorithm only needs to check the waypoints at both edge ends to connect the containing subregions. Subregion graph can also be created for a navigation mesh by firstly constructing the dual graph from the navigation mesh, which can then be treated as an input. However, it should be noted that in spite of recent development in multi-layered environment, the navigation mesh structure can only handle surface motion.

6. Path Planning Using Subregion Graph

Given two arbitrary points, we firstly look for their respective nearest traversable waypoints from them, using traversability test similar to the method Wardhana et al. proposed [WJS13]. Then, in the subregion graph level we perform the Dijkstra's algorithm on the subregion graph that supports the character's movement between the subregions that contain those nearest traversable waypoints. This results in *subregion path*, which contains a list of subregions. If the subregion path is empty, there is no path between the two points, and the algorithm stops here. Otherwise, we find the path between the two waypoints using filtered A* algorithm, which only expands waypoints inside the subregions in the subregion path. The original arbitrary points are appended to the result of this step, and the final path is returned. This path can be further smoothened in a post-processing step, for example using a technique akin to Pinter's algorithm [Pin01].

If there is a path between two waypoints, there is also a subregion path in subregion graph which connects the subregions containing them. Likewise, if there is a subregion path between two subregions, there is also a path between every waypoint in both subregions. Planning a path between the same pair of waypoints without and with subregion graph will either both return paths or both not return paths.

7. Experimental Results

We performed a few experiments to evaluate the subregion graph generation and path planning algorithms on a workstation with Intel Xeon E5-1650 @ 3.20 GHz processor and 16 GB of memory. We used the graph data structure, Dijkstra's, and filtered A* search implementations in Boost Graph Library [SLL14]. Ogre3D library [The11] was used as the rendering engine. We used two virtual environments, namely Kampong Glam and Marina Bay, which are two areas in Singapore. The Kampong Glam environment has an area of 488.65m $\times$510.9m with 79,875 triangles, whereas the Marina Bay environment has an area of $2,338.9$m $\times 2,832.48$m, with 555,238 triangles.

Subregion graph is compact. The number of subregions is not more than 3% of the number of waypoints, whereas the number of subregion connectivities is not more than 0.2% of the number of edges. The generation mechanism itself takes only around 20 seconds to process a graph with millions of waypoints and tens of millions of edges.

We also performed an experiment to evaluate the quality of path planning result using subregion graph. It involved

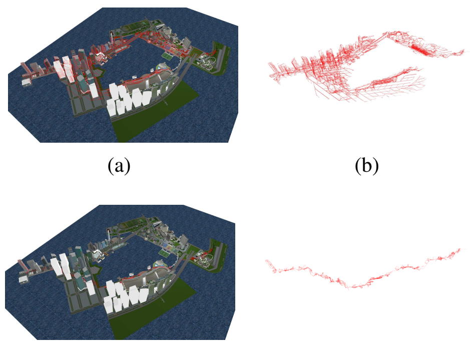

Figure 3: *(a) Without subregion graph, 120,878 edges were traversed (red lines) when planning a path between two distant waypoints. (b) The same set of edges with the environment hidden. (c) With subregion graph, the number of traversed edges is reduced to 13,483, which implies faster path planning. (d) The same set of edges with the environment hidden.*

sampling waypoints from the waypoint graph, and performing path planning repeatedly for every pair of these waypoints without subregion graph and with subregion graph, considering different cost schemes. These schemes were evaluated using two measurements, namely average length ratio and average speedup. The average length ratio describes the average deviations of the path lengths planned using subregion graph with a specific cost scheme, compared to the path lengths planned without a subregion graph. The average speedup explains how much acceleration was achieved on average for a particular cost scheme.

These experiments and calculations show that path planning using subregion graph with any cost scheme results in paths that are not significantly longer than paths planned without subregion graph, as their average length ratios are close to 100%. Subregion graph can accelerate path planning, via graph traversal restriction to the subregion path, as visualised in the example in Fig. 3. With subregion graph, up to more than 8 times acceleration can be achieved, while average path length ratios are maintained to as little as only 102.5%. CD cost scheme consistently had the lowest average length ratios, whereas in most cases where paths are found, FC cost scheme has the highest average speedup. If paths do not exist, higher speedups of over 200 times can be achieved.

8. Conclusion and Future Work

We have presented subregion graph, a compact graph-based data structure which can be used to accelerate path planning in very large virtual environments involving diverse characters with various motion types without significantly sacrificing path length. Subregion graph acts as an abstraction on top of a waypoint graph, and its primary role is to reduce the number of visited waypoints during path planning. We also presented a fast algorithm to automatically generate subregion graph, as well as a path planning method using subregion graph.

For future work, we will provide more discussion on the influence of the subregion graph size with respect to the input enhanced waypoint graph size, as well as comparison with existing techniques. Exact path planning algorithm using subregion graph as well as using adaptive subdivision (e.g. octree [GKB14]) to build a subregion graph are also a few interesting directions to explore. It might also be possible distribute path planning process to different frames, to allow other processes, such as rendering and the actual movement, to be performed. Another research direction is towards efficient multi-character path planning, for example planning crowd paths in a very large urban scene. Finally, due to the fast generation algorithm, we foresee that we can perform real-time update of subregion graph in a dynamic environment.

Acknowledgment

The models of Kampong Glam and Marina Bay were created by Media Development Authority (MDA) Singapore, and are used in this research with permissions. This research is supported by the National Research Foundation Singapore under its Interactive Digital Media (IDM) Strategic Research Programme. Henry Johan is supported by Fraunhofer IDM@NTU, which is funded by the National Research Foundation (NRF) and managed through the multiagency Interactive & Digital Media Programme Office (IDMPO) hosted by the Media Development Authority of Singapore (MDA).

References

[FBSS09] FELNER A., BARER M., STURTEVANT N., SCHAEFFER J.: Abstraction-based heuristics with true distance computations. In *Proceedings of the Eighth Symposium on Abstraction, Reformulation, and Approximation* (Lake Arrowhead, CA, USA, Aug. 2009). 2

[Flo62] FLOYD R. W.: Algorithm 97: Shortest path. *Communications of the ACM 5*, 6 (June 1962), 345. 2

[Fre87] FREDERICKSON G. N.: Fast algorithms for shortest paths in planar graphs, with applications. *SIAM Journal of Computing 16*, 6 (Dec. 1987), 1004–1022. 2

[GH05] GOLDBERG A. V., HARRELSON C.: Computing the shortest path: A* search meets graph theory. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms* (Philadelphia, PA, USA, 2005), SODA '05, Society for Industrial and Applied Mathematics, pp. 156–165. 2

[GKB14] GARĆIA F. M., KAPADIA M., BADLER N. I.: GPU-based dynamic search on adaptive resolution grids. In *Proceedings of 2014 IEEE International Conference on Robotics and Automation (ICRA)* (May 2014), pp. 1631–1638. 5

[Gut04] GUTMAN R. J.: Reach-based routing: A new approach to shortest path algorithms optimized for road networks. In *Proceedings of the 6th Workshop on Algorithm Engineering and Experiments (ALENEX) and the First Workshop on Analytic Algorithmics and Combinatorics (ANALCO)* (New Orleans, LA, USA, 2004), Arge L., Italiano G. F., Sedgewick R., (Eds.), SIAM, pp. 100–111. 2

[HMZM96] HOLTE R. C., MKADMI T., ZIMMER R. M., MACDONALD A. J.: Speeding up problem solving by abstraction: A graph oriented approach. *Artificial Intelligence 85*, 1-2 (Aug. 1996), 321–361. 2

[Lau04] LAUTHER U.: An extremely fast, exact algorithm for finding shortest paths in static networks with geographical background. In *Geoinformation und Mobilität - von der Forschung zur praktischen Anwendung*, Raubal M., Sliwinski A., Kuhn W., (Eds.), vol. 22 of *IfGI prints*. Institut für Geoinformatik, Westfälische Wilhelms-Universität, Münster, Germany, 2004, pp. 219–230. 2

[MH04] MOULD D., HORSCH M.: An hierarchical terrain representation for approximately shortest paths. In *PRICAI 2004: Trends in Artificial Intelligence*, Zhang C., W. Guesgen H., Yeap W.-K., (Eds.), vol. 3157 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2004, pp. 104–113. 2

[Pin01] PINTER M.: *Toward more realistic pathfinding* [Online]. Available: http://www.gamasutra.com/features/20010314/pinter_01.htm, March 2001. 4

[SB05] STURTEVANT N., BURO M.: Partial pathfinding using map abstraction and refinement. In *Proceedings of the 20th National Conference on Artificial Intelligence - Volume 3* (2005), AAAI'05, AAAI Press, pp. 1392–1397. 2

[SLL14] SIEK J., LEE L.-Q., LUMSDAINE A.: Boost Graph Library (version 1.57). http://www.boost.org/libs/graph/, 2014. 4

[SS05] SANDERS P., SCHULTES D.: Highway hierarchies hasten exact shortest path queries. In *Proceedings of the 13th Annual European Conference on Algorithms* (Berlin, Heidelberg, 2005), ESA'05, Springer-Verlag, pp. 568–579. 2

[The11] THE OGRE TEAM: Ogre – Object-oriented Graphics Rendering Engine (version 1.7.3). http://www.ogre3d.org/, May 2011. 4

[War62] WARSHALL S.: A theorem on boolean matrices. *Journal of the ACM 9*, 1 (Jan. 1962), 11–12. 2

[WJS13] WARDHANA N. M., JOHAN H., SEAH H.: Enhanced waypoint graph for surface and volumetric path planning in virtual worlds. *The Visual Computer 29*, 10 (2013), 1051–1062. 1, 2, 4

[WW03] WAGNER D., WILLHALM T.: Geometric speed-up techniques for finding shortest paths in large sparse graphs. In *Algorithms - ESA 2003*, Battista G., Zwick U., (Eds.), vol. 2832 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2003, pp. 776–787. 2