A Comparative Study of Navigation Meshes

Wouter van Toll[1] **Roy Triesscheijn**[1] **Marcelo Kallmann**[2] **Nuria Pelechano**[3]
Ramon Oliva[4] **Julien Pettré**[5] **Roland Geraerts**[1]

Keywords: navigation mesh, path planning, virtual environments, comparative study, geometry processing.
Affiliations: [1] Utrecht University. [2] University of California, Merced. [3] Universitat Politècnica de Catalunya. [4] Universitat de Barcelona. [5] INRIA.
Corresponding author: Wouter van Toll. E-mail: W.G.vanToll@uu.nl. Work address: Utrecht University, Department of Information and Computing Sciences, Princetonplein 5, 3584 CC Utrecht.

1 Introduction

Path planning for autonomous characters in 2D or 3D virtual environments is a fundamental task in simulations and games. A *navigation mesh* is a representation of a virtual environment that facilitates this. Various state-of-the-art navigation meshes exist [1–4,6], but there is no standardized way of evaluating or comparing them. Each implementation is in a different state of maturity, has been tested on different hardware, uses different example environments to show its (dis)advantages, and may have been designed with a different application in mind.

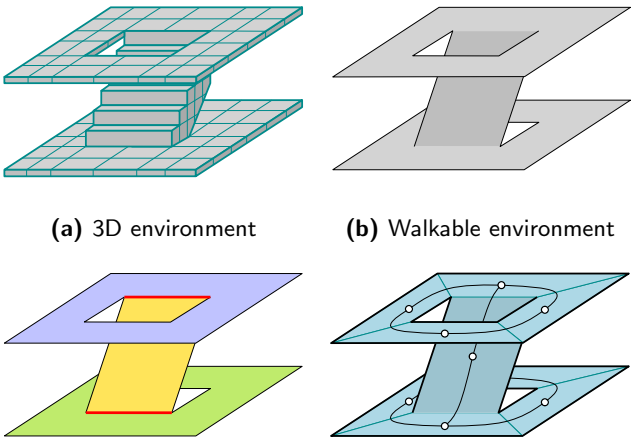
This abstract summarizes recent work [5] in which we have conducted the first *comparative study* of navigation meshes. The goal of this study was not to find 'the best' navigation mesh, but to develop a way of comparing navigation meshes based on theoretical properties and quantitative metrics. We expect that this study will set a new standard for the evaluation and development of navigation meshes, and that it will steer future research into interesting directions.

We summarize our study as follows. In Section 2, we introduce general definitions of environments and navigation meshes. In Section 3, we list theoretical properties by which navigation meshes can be classified. In Section 4, we give metrics for objectively measuring the quality of a navigation mesh for a given input environment. In Section 5, we use these metrics to compare 6 state-of-the-art navigation meshes in a range of environments. We analyse our results to identify important topics for future research.

2 Definitions

Each paper on navigation meshes uses slightly different definitions for the input (a virtual environment) and output (a navigation mesh). To enable an objective comparison, we first give general definitions of these concepts.

A **3D environment** (3DE, Figure 1a) is a collection of planar polygons in $\mathbb{R}^3$. These polygons may include floors, ceilings, walls, et cetera. To define the *free space* $\mathcal{E}_{free}$ of a 3DE, we need to describe on which surfaces our characters may walk. This is determined by parameters such as the maximum slope of a surface, the maximum step height of a staircase, and the required vertical distance between a floor and a ceiling.



(a) 3D environment **(b)** Walkable environment **(c)** Multi-layered environment **(d)** Navigation mesh

Figure 1: Different representations of an environment, and an example of a navigation mesh. These images correspond to the definitions in Section 2.

A **walkable environment** (WE, Figure 1b) is a set of interior-disjoint polygons in $\mathbb{R}^3$ on which characters can stand and walk. Thus, a WE is a clean representation of the free space $\mathcal{E}_{free}$ of a 3DE. Note: it is possible that not all surfaces of a WE are visible from a single top view.

A **multi-layered environment** (MLE, Figure 1c) is a subdivision of a WE into 2D components (*layers*) such that the polygons of each individual layer are non-overlapping when projected onto the ground plane. The layers are connected by line segments (*connections*).

A **2D environment** (2DE) can be seen as a special case of a WE that is entirely visible from above, or (equivalently) as an MLE with only one layer. This allows us to use the same metrics in both 2D and 3D.

Finally, a **navigation mesh** (Figure 1d) describes how characters can navigate through an environment $\mathcal{E}$. It is given by a set of regions $\mathcal{R}$ that should represent the free space $\mathcal{E}_{free}$, plus a graph $\mathcal{G}$ that describes how these regions are connected.

Some algorithms compute a navigation mesh from a 'clean' 2DE or MLE. Others take a raw 3DE as input and compute their own *approximation* of the free space.

3 Properties of Navigation Meshes

We propose a set of theoretical properties that describe a navigation mesh's data structure, algorithms, features, and limitations. These properties do not depend on a specific implementation or input environment. They are useful for broadly categorizing current navigation mesh research, and they can serve as a checklist for choosing an appropriate mesh for a particular application.

Region type. The type of regions in $\mathcal{R}$, e.g. triangles or disks.

Graph type. A description of the graph $\mathcal{G}$, e.g. 'the dual graph of $\mathcal{R}$' or 'the medial axis of $\mathcal{E}_{free}$'.

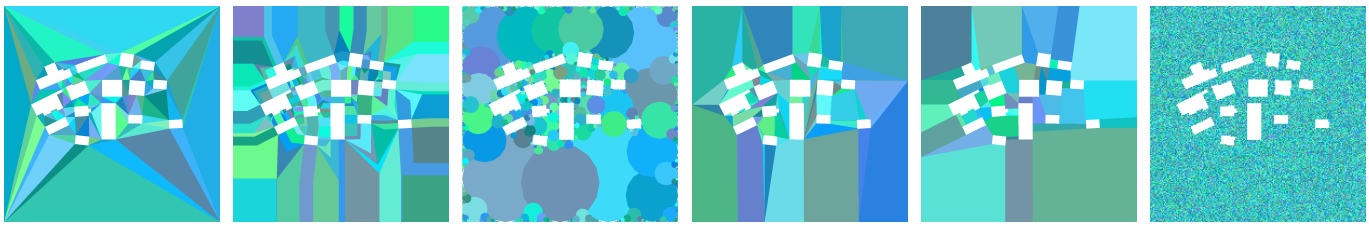


Figure 2: Navigation meshes computed for a simple 2D environment. Regions are shown in different colors. From left to right: the Local Clearance Triangulation [1], the Explicit Corridor Map [6], the Clearance Disk Graph [4], Recast [2], NEOGEN [3], and a grid.

Overlap. Whether or not the regions in $\mathcal{R}$ are allowed to overlap.

Pipeline. The conversion steps performed by the construction algorithm, e.g. 'from 2D environment to navigation mesh' or 'from 3DE to MLE to navigation mesh'.

Parameters. The parameters that the user needs to set for the construction algorithm of the navigation mesh. Having fewer parameters implies a more automated process for computing the mesh.

Computational complexity. The asymptotic construction time of the navigation mesh, often expressed in terms of the environment complexity or a grid resolution.

Storage complexity. The asymptotic size of the navigation mesh data structure.

Clearance. Whether or not the navigation mesh supports the computation of paths with an arbitrary clearance from obstacles, i.e. paths for disks with any radius.

Dynamic updates. Whether or not the navigation mesh supports dynamic insertions/deletions of obstacles.

In the full version of our paper, we use these properties to categorize several state-of-the-art navigation meshes.

4 Quality Metrics for Navigation Meshes

For a navigation mesh $\mathcal{M} = (\mathcal{R}, \mathcal{G})$ that has been constructed for an environment using a particular implementation, we want to objectively measure the quality. To this end, our paper [5] defines metrics in four categories:

Coverage. How accurately do the regions of $\mathcal{R}$ cover the geometry of $\mathcal{E}_{free}$? This determines how useful the mesh is for computing paths through the environment. Coverage is easy to define in 2D, but much less so in 3D. One of our main contributions is the definition of proper coverage metrics for 2D *and* 3D environments.

Connectivity. How accurately does the graph $\mathcal{G}$ represent the connectivity of $\mathcal{E}_{free}$? This question also determines whether or not appropriate paths can be found, but it concerns topology rather than geometry.

Complexity. How compactly does $\mathcal{M}$ represent $\mathcal{E}_{free}$? This can refer to the size of the graph $\mathcal{G}$ or to the complexity of the regions in $\mathcal{R}$. It depends on the application which property is more desirable.

Performance. How efficiently is $\mathcal{M}$ computed in terms of time and memory? An efficient algorithm allows the construction of navigation meshes in interactive applications such as level editors.

5 Experiments and Results

We use our definitions and metrics to compare state-of-the-art navigation mesh implementations. The Local Clearance Triangulation (LCT) [1] and the Explicit Corridor Map (ECM) [6] are *exact*: given a 'clean' 2DE or MLE, they produce a navigation mesh with perfect coverage within a provable time bound. The Clearance Disk Graph (CDG) [4], Recast [2], and NEOGEN [3] are *voxel-based*: given a 'raw' 3DE, they first use a 3D grid of 'voxels' to compute an approximation of the WE, which is then converted to a navigation mesh. For comparison, we have also added a grid method in which the walkable voxels are immediately used as navigation mesh regions. Figure 2 shows an example of the output of each method.

In our full paper, we report the results for 19 environments, ranging from simple 2D mazes to a complex 3D city. Although this set is not yet exhaustive, we can already draw careful conclusions from our results.

Voxel-based methods have parameters that can be tuned to yield good coverage, but they do not always preserve connectivity, and their construction time does not seem to scale well to large environments. These limitations highlight an important topic for future work: developing *exact* algorithms that automatically extract the walkable space from arbitrary 3D input in real-time.

We would like to add metrics that measure the efficiency of a navigation mesh for path planning: how much time does it take to compute paths in $\mathcal{G}$, and how efficiently can these be converted to geometric routes using the regions of $\mathcal{R}$? We have currently excluded such metrics because they are heavily implementation-dependent.

Another option is to look at the *quality* of the routes that are computed: how short are they, and how well do they correspond to real-life behavior? Measuring the 'realism' of a path is a difficult topic of ongoing research.

In conclusion, we have defined how to compare navigation meshes via generic definitions and metrics, and we have performed such a comparison to identify useful areas for future work.

References

[1] M. Kallmann. Dynamic and robust Local Clearance Triangulations. *ACM Transactions on Graphics*, 33(5), 2014.

[2] M. Mononen. Recast Navigation. `https://github.com/memononen/recastnavigation/`, 2016.

[3] R. Oliva and N. Pelechano. NEOGEN: Near optimal generator of navigation meshes for 3D multi-layered environments. *Computers & Graphics*, 37(5):403–412, 2013.

[4] J. Pettré, J.-P. Laumond, and D. Thalmann. A navigation graph for real-time crowd animation on multilayered and uneven terrain. In *Proc. 1st Int. Workshop on Crowd Simulation*, pages 81–89, 2005.

[5] W. van Toll, R. Triesscheijn, M. Kallmann, R. Oliva, N. Pelechano, J. Pettré, and R. Geraerts. A comparative study of navigation meshes. In *Proc. 9th ACM SIGGRAPH Int. Conf. on Motion in Games*, pages 91–100, 2016.

[6] W.G. van Toll, A.F. Cook IV, and R. Geraerts. Navigation meshes for realistic multi-layered environments. In *Proc. 24th IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 3526–3532, 2011.