FEDERAL UNIVERSITY OF RIO GRANDE DO SUL
ENGINEERING SCHOOL
ELECTRICAL ENGINEERING GRADUATE PROGRAM

JULIANA RUBENICH BRONDANI

PATHFINDING IN HIERARCHICAL REPRESENTATION OF LARGE REALISTIC VIRTUAL TERRAINS

Porto Alegre
2018

JULIANA RUBENICH BRONDANI

PATHFINDING IN HIERARCHICAL REPRESENTATION OF LARGE REALISTIC VIRTUAL TERRAINS

Master degree Thesis presented to the Graduate Program of Electrical Engineering (PPGEE) from the Federal University of Rio Grande do Sul as requirement to obtain the title of Master in Electrical Engineering.
Focus area: Automation and Control

Supervisor: Prof. Dr. Edison Pignaton de Freitas

Porto Alegre

2018

JULIANA RUBENICH BRONDANI

PATHFINDING IN HIERARCHICAL REPRESENTATION OF LARGE REALISTIC VIRTUAL TERRAINS

This thesis was judged adequate to obtain the title of Master in Electrical Engineering and was approved in its final form by the Supervisor and Examination Board.

Supervisor: Prof. Dr. Edison Pignaton de Freitas, UFRGS
PhD at Halmstad University - Halmstad, Sweden and Federal University of Rio Grande do Sul – Porto Alegre, Brazil

Examination Board:

Prof. Dr. Carlos Eduardo Pereira, UFRGS
PhD at University of Stuttgart – Stuttgart, Germany

Prof. Dr. Walter Fetter Lages, UFRGS
PhD at Aeronautics Institute of Technology – São José dos Campos, Brazil

Prof. Dr. Rafael Heitor Bordini, PUCRS
PhD at University College London – London, UK

PPGEE coordinator: Prof. Dr. Valner João Brusamarello

Porto Alegre, September, 2018.

PATHFINDING EM REPRESENTAÇÕES HIERÁRQUICAS DE GRANDES TERRENOS VIRTUAIS REALISTAS

A capacidade de executar uma busca de caminho (*pathfinding*) é de extrema importância em diversas áreas. Dentre elas, podemos destacar a área de simulações virtuais voltadas para treinamento militar. Na área militar, simulações são comumente utilizadas para auxiliar o treinamento de militares nas habilidades e conhecimentos necessários para executar tarefas de acordo com a doutrina militar. Além disso, elas também visam diminuir custos enquanto aumentam a segurança desse treinamento, a visibilidade de eventos e a reprodutibilidade de ações (Fletcher, 2009).

Porém, para que esses sistemas de simulações realistas sejam efetivos, é necessário que eles apresentem algumas características. Dentre elas podemos destacar: a necessidade de uma representação realista do ambiente virtual; a possibilidade de interação entre o usuário e o sistema e seus agentes; a possibilidade de os agentes navegarem de forma (semi) autônoma no ambiente virtual; e a necessidade de obedecer um limite de tempo de resposta nas interações para que o usuário não perca a imersão com o sistema de simulação.

Essas características geram requisitos para o algoritmo de *pathfinding* que são desafiadores de serem tratados. Os algoritmos de *pathfinding* nesses sistemas de simulação precisam encontrar um caminho em um terreno realista de grandes dimensões dentro de um tempo de resposta limitado e ainda precisam evitar colisões com obstáculos de navegação e outros agentes. Para atacar esses problemas, técnicas hierárquicas que buscam a otimização da navegação global (e.g. (Botea, Müller, & Schaeffer, 2004) (Fuentes, 2015)) surgiram como uma forma de melhorar a performance temporal dos algoritmos de busca de caminho. Essas técnicas buscam deixar o processo de *pathfinding* mais rápido mitigando a memória e o tempo consumido para execução através de abstrações do terreno virtual. Porém, à nosso conhecimento, essas técnicas ainda não foram bem exploradas para serem aplicadas em terrenos massivos como os de simulações virtuais. Além disso, essas técnicas ainda precisam manter em memória diversas cópias das representações do terreno virtual em diferentes resoluções.

Sendo assim, esse trabalho está inserido nessa linha de pesquisa que utiliza técnicas hierárquicas para representação das estruturas de navegação em grandes terrenos virtuais. Visando obter resultados mais rápidos dos algoritmos de busca de caminho, esse trabalho propõe a implementação de uma solução de navegação hierárquica que executa em dois passos. Essa solução também utiliza um mapa de navegação baseado em uma estrutura de *quadtree* hierárquica combinado com algoritmos de busca de caminho. No primeiro passo, a solução proposta representa todo o terreno virtual utilizando uma estrutura de *quadtree* construída *offline* e executa um algoritmo de *pathfinding* sobre ela para encontrar um caminho que conecta dois pontos. A estrutura de *quadtree* permite a redução do tamanho do espaço de busca dos algoritmos de busca de caminho em áreas esparsas do terreno virtual enquanto mantém a possibilidade de representar obstáculos de navegação com um alto nível de detalhamento em áreas densas. No segundo passo, a solução proposta utiliza o caminho resultante do primeiro passo para criar, em tempo de execução, uma representação mais refinada da estrutura de *quadtree* sobre esse caminho. Após esse refinamento, o algoritmo de busca de caminho é executado novamente e utiliza apenas o refinamento da estrutura de *quadtree* como espaço de busca para encontrar um caminho mais refinado como solução.

A motivação para o desenvolvimento desse trabalho é a necessidade de otimizar a navegação global em terrenos virtuais grandes e realistas e a necessidade dessa solução ser aplicável um sistema de simulação militar real. Particularmente, esse trabalho é motivado pela pesquisa e desenvolvimento de um sistema militar de simulação real: o sistema de simulação

virtual tático SIS-ASTROS (ASTROS, 2018). Nesse sistema de simulação real, navegação global e prevenção de colisão precisam ser realizadas em um terreno realista de dimensões massivas que é construído a partir de informação GIS de um terreno real (Frasson, Engel, & Pozzer, 2018) (Backes, Frasson, Engel, & Pozzer, 2017). Para simular a navegação dos agentes nesse ambiente virtual, o sistema se simulação SIS-ASTROS depende de um framework de navegação híbrido (Brondani, de Freitas, & Silva, 2017) no qual a solução hierárquica de navegação global está integrada com técnicas de navegação local.

Para avaliar as técnicas hierárquicas propostas nesse trabalho, experimentos são executados em diferentes tipos de estruturas de representação de terreno regulares e irregulares. Esses experimentos mostram que o algoritmo proposto que utiliza a estrutura de *quadtree* hierárquica em conjunto com um algoritmo de busca de caminho é capaz de deixar o processo de busca de caminho mais rápido em ambientes de simulação realísticos. Os resultados demonstram que essa otimização no tempo de execução é mantida à medida que o tamanho do terreno virtual de simulação aumenta mas tem como custo uma pequena perda de qualidade de caminho.

Palavras chave: Terreno hierárquico; Representação de mapa de navegação; Grandes terrenos virtuais; Pathfinding hierárquico; Sistemas de simulação.

AKCNOWLEDGMENTS

To the Graduate Program in Electrical Engineering, PPGEE, for the opportunity to develop works in my research area.

We also thank the Brazilian Army for the financial support through the SIS-ASTROS Project (813782/2014), developed in the context of the PEE ASTROS 2020.

ABSTRACT

Pathfinding is critical to virtual simulation applications. One of the most prominent pathfinding challenges is the fast computation of path plans in large and realistic virtual terrain environments. To tackle this problem, this work proposes the exploration of a quadtree structure in the navigation map representation of large real-world virtual terrains. Exploring a hierarchical approach for virtual terrain representation, we detail how a global hierarchical pathfinding algorithm searches for a path in a coarse initial navigation map representation. Then, during execution time, the pathfinding algorithm refines regions of interest in this terrain representation in order to compute paths with a higher quality in areas where a large amount of navigation obstacles is found. The computational time of such hierarchical pathfinding algorithm is systematically measured in different hierarchical and non-hierarchical terrain representation structures that are instantiated in the modeling of a small real-world terrain scenario. Then, similar experiments are developed in a large real-world virtual terrain that is inserted in a real-life simulation system for the development of military tactical training exercises. The results show that the computational time required to generate pathfinding answers can be optimized when the proposed hierarchical pathfinding algorithm along with the easy and reliable implementation of the quadtree-based navigation map representation of the large virtual terrain are explored in the development of simulation systems.

Keywords: Hierarchical terrain; Navigation map representation; Large virtual terrain; Hierarchical pathfinding; Simulation systems.

SUMMARY

FIGURE LIST

TABLE LIST

ALGORITHM LIST

1 INTRODUCTION

Pathfinding in large realistic terrains is of paramount importance in robotics (Kruse, Pandey, Alami, & Kirsch, 2013) (Algfoor, Sunar, & Kolivand, 2015) (Bayat, Najafinia, & Aliyari, 2018), computer games (Souissi et al., 2013) (Cui & Shi, 2011) and simulation systems (Wang, Li, Rezgui, Bradley, & Ong, 2014) (Cil & Mala, 2010), to name a few applications. In the military setting, simulation systems are mostly used in the training of military personnel, aiming to "reduce costs while increasing safety, visibility of events, and reproducibility of actions" (Fletcher, 2009). To a large variety of educational goals, however, these realistic simulation systems must present some characteristics so the training exercises are effective. These simulation systems should allow users to interact with a virtual environment and the different kinds of computer-generated forces properly represented in the system. Moreover, such agents representing military forces in these simulation training tools should be well able to navigate (semi) autonomously in large realistic virtual environment while maintaining a certain response time. These characteristics result in challenging problems for the exploration of different techniques in the construction of navigation structures and algorithms in such large virtual scenarios.

A challenging problem to be handled is to find ways of optimizing the computational time required to obtaining time-constrained path planning results in such large search spaces while avoiding collision with different navigation obstacles. To approach this issue, alternative hierarchical techniques aiming the optimization of virtual terrain navigation maps have emerged (Botea, Müller, & Schaeffer, 2004) (Fuentes, 2015) as a way of enhancing the computational time performance of pathfinding algorithms.

In this context, this work contributes to this line of research that relies on the hierarchical representation of the navigation map structure of large realistic virtual terrains.

Aiming at obtaining path planning results from hierarchical pathfinding algorithms over such large virtual environments, this work proposes the implementation of a two-step hierarchical navigation solution that uses a quadtree-based navigation map structure combined with a global pathfinding algorithm. In the first step, the proposed solution represent the entire large realistic terrain using a quadtree structure and executes a pathfinding algorithm upon it. The quadtree structure allows the reduction of the search space in sparse virtual terrain areas while maintaining the possibility of representing the navigation obstacles of dense terrains areas with a high level of detail. In the second step, the proposed solution uses the resultant path from the first step to create a more refined representation of the quadtree-based structure during execution time. Then, the pathfinding algorithm is executed again using only the refined representation as search space and finding a refined path as solution.

Particularly, this work is motivated on the research and development of a real-life military simulation system – the SIS-ASTROS simulation system for the systematic development of virtual tactical simulation-based training exercises (ASTROS, 2018). In such virtual tactical simulations, effective global navigation and obstacle avoidance are achieved in a large and realistic virtual terrain setting that is constructed from GIS-based map data regarding real-world terrains (Frasson, Engel, & Pozzer, 2018) (Backes, Frasson, Engel, & Pozzer, 2017). To simulate prompt agent navigation behaviors in this virtual environment, the SIS-ASTROS simulation system relies on a hybrid semi-autonomous navigation framework (Brondani, de Freitas, & Silva, 2017) in which the proposed hierarchical global pathfinding solution along with force-based local navigation techniques are integrated. To evaluate the hierarchical techniques proposed in this work, experiments developed in different kinds of regular and irregular terrain representation structures show that the hierarchical quadtree approach along with a global pathfinding algorithm composes a new navigation approach that is capable of speeding up the required pathfinding computations in realistic simulation environments. The

results show that the optimized execution time of such pathfinding searches is maintained as the size of the search space representing the virtual simulation environments is increased with the cost of a slight reduction in path quality

The thesis is organized as follows. Chapter 2 reviews navigation strategies and terrain topologies. Chapter 3 highlights relevant works to the current proposal. Chapter 4 describes the real-world simulation application in which this work is inserted and the construction process of the structure used to represent the large virtual terrain. Chapter 5 describes the proposed hierarchical virtual terrain representation and pathfinding solution for the global navigation in large realistic virtual terrain. Chapter 6 summarizes the experiments' setup in different simulation scenarios. Chapter 7 presents the results of these different experiments and discusses the experimental results. Finally, Chapter 8 concludes the work and suggests directions for future work.

2 BACKGROUND

This chapter reviews some essential concepts related to navigation and pathfinding algorithms as many of them are mentioned along with the text. The concepts discussed in this chapter are related not only to navigation strategies and reasoning, but also to the representation of the virtual environment where the navigation component is inserted. The virtual environment can be represented using different structures that are suitable for different types of applications. A discussion about their differences is presented for a better understanding of the choices made for the solution proposed in this work.

2.1 STATIC X DYNAMIC REASONING

In static reasoning, the navigation algorithms consider that the structure of the virtual scenario and the position and geometry of the terrain features and agents on it does not change. In a system with a static approach, the data about the virtual terrain can be pre-computed to save memory consumption and execution time, as it will remain the same throughout the execution of the system.

In contrast, in dynamic reasoning, the virtual terrain, terrain features, and other agents can move and be deformed at runtime, and the navigation technique must consider this. For example, in a military simulation, consider that a bridge connects two sides of a river and allows the passage of military vehicles. The pathfinding algorithm can use this bridge as a valid path for the agents to navigate across the river. However, during the simulation, this bridge can be destroyed by an enemy attack. In this case, the virtual description must change at runtime to set that this bridge is not traversable anymore and that the pathfinding algorithm cannot use it as a valid path in its future executions.

2.2 NAVIGATION STRATEGIES

2.2.1 Global Navigation

In a global navigation strategy, the solution has the form of a plan, as it is a sequence of actions. This plan leads the agent from an initial to a goal state, and it is calculated before the beginning of the navigation task or whenever a new situation arises (Khatoon & Ibraheem, 2012). For this reason, systems that require this type of navigation usually keep a symbolic representation of the virtual terrain and the key goals of the system.

However, doing the necessary approximations and calculations during the navigation planning creates a high computational cost. The ability to consider every detail of the navigation map yields a more intelligent response, but it can often result in the system not being able to produce real-time responses.

2.2.2 Local Navigation

Local navigation strategies intend to produce prompt and robust navigation actions in response to a dynamic environment (Shen and Zhou, 2006). The modeling of a local or low-level navigation behavior is commonly used in these systems, as there is no need for a global view and representation of the virtual environment where the agent is navigating. To realistically model actions regarding navigation, local navigation systems maintain a continuous relationship with the local surroundings of their environment. Even though these systems have the possibility of dealing with dynamic situations, they may not scale up well since the calculations to keep track of the surroundings become more complicated as the number of agents and dynamic obstacles increases in the virtual environment.

2.2.3 Hybrid Navigation

Hybrid navigation is a combination of the global and local navigation methods (Shen and Zhou, 2006). This approach typically leads to layered architectures reflecting the natural pattern of a human's decision-making process. In these processes, long-term navigation goals are decided, and then, while trying to achieve them, agents have to deal with dynamic/real-time decisions from non-expected situations.

2.3 TERRAIN TOPOLOGIES

The terrain representation is a critical component for global pathfinding. To search a path in a virtual environment, a global pathfinding algorithm has to deal with relevant requirements: the search algorithm needs to be able to access and interpret data about the virtual environment description, and this data about this environment needs to be stored in a structure that allows simple and fast access during the search (Anguelov, 2011). In addition, the choice of which pathfinding solution to use depends on the developers' goals, and on the virtual environment under concern. For example, some systems may require real-time responses from the navigation module dealing with a large number of agents in the virtual terrain. However, they may not require that the returned path have a high quality, which means that the path does not need to be similar to the actual shortest path between two points. In this situation, the terrain representation would not require a very accurate description of the virtual terrain features, as long as this description allows a fast calculation of such path. In summary, the virtual terrain representation can have a direct impact on the performance of the search algorithm, and in the consequent precision of the collision avoidance strategy obtained when such algorithms are used.

There are different virtual terrain representation methods as reported in the literature ((Souissi et al., 2013), (Algfoor et al., 2015), (Kapadia & Badler, 2013)). This chapter highlights

the most common virtual terrain representations structures used nowadays for navigation systems (Sturtevant, 2013): grids, visibility graphs, navigation meshes, and quadtrees.

2.3.1 Regular Grids

Regular grids are simple to implement and update as they always have the same number of cells and edges independently of the number of terrain features they have to represent in the virtual scenario (Souissi et al., 2013). Each of these cells can be marked as either traversable or blocked according to the presence of terrain features, among other criteria. This topology divides the virtual terrain into cells in the form of regular polygons (like squares, triangles or hexagons) of the same size.

The regular grid terrain representation has some drawbacks. The first is that the modeling of terrain features is likely to be less accurate as the shape and size of the cells may not allow their representation to be realistic. For example, a regular grid with square cells may not allow the representation of circular obstacles accurately. A second drawback is the memory consumption and associated time overhead for searches in its structure. Regular grids can result in substantial search spaces when representing virtual terrains requiring cells with fine granularity. For virtual terrains with large dimensions, the execution of a search algorithm has a negative effect on the performance and memory consumption of the system (Anguelov, 2011).

2.3.2 Irregular Grids

Irregular grids are characterized by grid cells that can have different shapes and sizes. When explored by search algorithms, irregular grids may require less memory to execute the search than if such kind of search was being executed on regular grids. This type of terrain representation speeds up the exploration of the search space because an irregular grid can subdivide only particular regions of interest. This cannot be done in regular grids. In an irregular

grid, for example, an area that does not have any navigation obstacles will not be divided into smaller cells because there is no need to refine it further in the resulting virtual terrain. The agent can navigate through a whole area without having to avoid collision with the navigation obstacles in that area. In virtual terrains represented by irregular grids, large cells that represent large traversable areas are often present. According to (Kapadia & Badler, 2013), topologies considered as irregular grids are waypoint graphs, visibility graphs, and navigation meshes. In addition to these, a quadtree representation of the virtual terrain is considered as a type of irregular grid as described in (Algfoor et al., 2015), although a quadtree representation can also be understood as a hierarchical technique. Table 1 was created to present a comparison between these irregular grid techniques. This comparison highlights how these techniques are constructed and the pros and cons of using them in a real-world application.

Table 1- A comparison of irregular grid representation techniques.

Irregular Grid Techniques				
	Waypoint Graphs	**Visibility Graphs**	**Navigation Mesh**	**Quadtree**
Construction	● Waypoints are placed across the virtual terrain and are linked to create a graph. ● Waypoints construction can be done manually or autonomously.	● Models the topology of the terrain by linking the terrain navigation obstacles' vertices between themselves and with the start and goal positions of the path (Souissi *et al.*, 2013).	● Represents walkable areas of a map by a convex polygon mesh.	● Repeatedly divides square grid cells into four smaller ones when there is a terrain feature of interest inside its bounds.
Pros	● Compared to regular grids, waypoint graphs are sparser, which makes them cheaper to store and inexpensive to answer pathfinding requests.	● The shortest path between two positions is either a straight line or a tangential to the terrain navigation obstacles. ● The path found will be the actual shortest path.	● Low number of edges per node and linear number of nodes that are created according to the presence of terrain navigation obstacles. ● Polygons can describe the terrain navigation obstacles more accurately. ● Coarser terrain description than other techniques.	● Reduction of the memory usage and the size of search space.
Cons	● Does not guarantee full coverage of the virtual terrain. ● Can generate redundant nodes that unnecessarily increase the search space (Anguelov, 2011).	● If start and goal positions of pathfinding are not already on the graph, they must be inserted and connected to the other vertices before computing the path. ● The time for this connection may become higher than the search time. (Koefoed-Hansen e Brodal, 2012).	● The subdivision process of the virtual terrain in a polygonal mesh has a high computational cost. ● Usually it is created offline. ● Computational cost for creation is a disadvantage in dynamic environments.	● Loss of path quality. ● Does not present advantages when the virtual terrain has a large density of terrain obstacles that are spread across its entire surface.

2.3.3 Discussion

Regular grids are a relevant choice to represent virtual terrains when there is no need for a high level of detail in the representation of navigation obstacles. In this situation, even a large terrain can be represented by large cells and create a small search space. However, regular grids are not good solutions for the representation of large virtual terrains when it is necessary

to describe the terrain navigation obstacles with details. That is because a high level of details forces the grid representation to create a large number of small cells, which results in the construction of a substantially large search space to be accessed by the pathfinding algorithm.

The navigation mesh and the quadtree stand out for usage in large terrains in which the density of navigation obstacles is variable. With a more accurate representation of navigation obstacles and, at the same time, a coarser representation of navigation space with polygons, a navigation mesh can provide fast pathfinding without compromising the quality of the path. However, the construction of a navigation mesh is a complex process when compared to the other techniques. Among other reasons, this construction depends on geometry data from the features that are represented in the virtual terrain. If the application where the navigation mesh is inserted does not contain this type of data, this technique may not be explored. In contrast, a quadtree is a suitable solution for global pathfinding in large terrains with a large number of navigation obstacles due to its hierarchical nature. In this case, a quadtree cell is divided into four smaller cells either until there are no more navigation obstacles represented in this cell or until the quadtree reaches a maximum level of division. Cells that do not need to be subdivided and, therefore, do not have children cells are called “leaf cells.”

To assist the pathfinding process, this quadtree structure can be understood as an irregular grid. That is because the search algorithm only needs to search the “leaf cells” to find a path to navigate the virtual terrain. From a bottom-up perspective, the “leaf cells” of the quadtree are organized as an irregular grid that is formed by square cells of different sizes. As a square cell is only subdivided when it contains features of interest, the reduction of search space occurs. That is because sparse areas (which do not contain these features) will only be represented by single large cells. So, the size of the resulting search space is reduced which is a fact that can speed up the process of searching for a path. Although the use of large cells for representing terrain areas tends to reduce de size of the search space, some loss of path quality

may occur. That is because this path does not necessarily represent the optimal movement that agents may need to make to travel in minimum time and distance to their destinations.

In the context of this work, the quadtree is still a suitable solution to be explored in the real-world military simulations. Among other reasons, the usage of this topology in a hierarchical pathfinding approach can present a solution for the problem of low quality of the planned path. That is because this hierarchical structure allows refining the representation detail of navigation obstacles so it can describe these features more accurately. As a result of this more accurate representation, this hierarchical structure can allow creating a resultant path with more quality. To be constructed, a quadtree structure also requires simpler input data regarding terrain navigation obstacles, which is a positive aspect for the simulation applications we are dealing with in the project that this work is inserted in.

3 RELATED WORK

This chapter presents a literature review about global pathfinding algorithms and how these algorithms can be optimized according to the application they are inserted into. In addition, pathfinding techniques that explore the hierarchical representation of the virtual terrain to speed up the search are further discussed as they are promising solutions to be utilized in applications in which large virtual terrains are required.

3.1 GLOBAL PATHFINDING

AI pathfinding techniques allow agents to move across either virtual environments or real-world environments, or both. In many applications, the computational time of pathfinding algorithms is a crucial aspect to be investigated. In applications like computer games, robotics and simulation systems, these paths have to be computed in real-time, sometimes with limited CPU and memory resources (Cowling et al., 2013). Even though the available processing capacity for these applications is increasing due to the constant development of computer hardware technology, many virtual computer games, and simulation systems only allocate around 20% of the total processing time per frame to the computation of these AI algorithms (Rabin, 2005). Due to these constraints, agent navigation problems have been handled by a variety of techniques that are often aimed at providing a high-performance search.

For global pathfinding, which is the focus of this work, the A* (Nilsson, 1998) and its variations are so far the most explored algorithms. According to (Dooms, 2013), A* is optimally efficient, and no other optimal algorithm can guarantee to expand fewer nodes of a navigation graph than the A*. Although the A* algorithm is widely explored in the implementation of navigation functions for computer game characters, its computational cost may become prohibitive with the increase of the application problem scale. In essence, such computation cost sharply increases as the size of the search space and the number of agents in the system scale up.

3.2 A* AND ITS VARIATIONS

There has been much research regarding the optimization of the A* search in the AI literature. A* variants differ according to the type of application they are being applied to. In a virtual environment with dynamic obstacles, for instance, A* variants worth mentioning are the Dynamic A* (D*) (Stentz, 1995) and the D* Lite (Koenig & Likhachev, 2002). These algorithms allow the modification of the already calculated path when faced with an unforeseen situation.

Variants of the A* search algorithm are also concerned with the improvement of path quality. It means that these algorithms focus on creating true and realistic shortest paths. An example of this type of algorithm is the Theta* (Daniel et al., 2010), a path planning algorithm for virtual environments represented by a grid. The Theta* algorithm propagates search information along the grid edges, but the resultant path is not constrained to movements only along these edges. It is possible to cross from one grid cell to another at any angle of movement.

There are also A* variants focusing on the real-time execution requirements. The Real-Time A* (RTA) and the Learning Real-Time A* (LRTA*) (Korf, 1990) are examples of this focus. These search algorithms do not wait for the whole path to be planned before allowing the agents to initiate their navigation. They plan a prefix of the path in a local region so the agents can immediately start moving. If this current path becomes blocked or if the agent reaches the end of the local search area, a new partial path is computed for the agents.

In this work, optimizations for the search algorithm are focused on exploring the representation of the virtual terrain to generate improvements in memory and computation time. Many techniques are aiming at the same target. In the Relaxed Dijkstra (RD) and the Relaxed A* (RA*) proposed in (Ammar et al., 2016) the authors focus on finding a solution for global path planning for mobile robot navigation in large-scale size problems. The objective is to find the shortest obstacle-free path for the robot to navigate between two locations in the large terrain

in the available time. For this, algorithms proposed by the authors explore a grid representation of the terrain and estimate an optimal path between two points without visiting any cell more than once. The results show that both the RD and RA* solutions provided a better trade-off between quality of the solution and execution time in a large virtual terrain representation compared to the basic Dijkstra and A* algorithms.

Another popular solution is the Anytime Dynamic A* (AD*) (Likhachev et al., 2005). This solution combines aspects of solutions focused on dynamic virtual environments and real-time execution as the authors want to solve problems of path planning in dynamic, relatively high-dimensional state spaces. The AD* continually improves a solution during deliberation time and reuses previous searches' information as much as possible.

Many prominent techniques that also explore the representation of the virtual terrain are based on creating a hierarchy of terrain structures or searches to speed up the pathfinding process. These hierarchical solutions are discussed in more details in the sequence.

3.3 HIERARCHICAL SOLUTIONS

A hierarchical pathfinding technique has as objective to mitigate the time consumption of the pathfinding process. To do so, these techniques commonly use abstractions of the virtual terrain representation to simplify the search process. These abstractions create different representations of the same virtual terrain with different levels of detail. A more detailed representation of the virtual terrain can represent the terrain features more accurately. In doing so, this more detailed representation also creates a large search space with small cells and, therefore, it has a fine granularity or a high resolution. In contrast, a less detailed representation of the virtual terrain creates a simpler search space with larger cells that does not describe the terrain features in the virtual terrain accurately. However, the simpler search space allows the search algorithm to execute faster than in the detailed representation. The less detailed representation of the virtual terrain has a coarse granularity or a low resolution.

Botea et al. (Botea, Müller, & Schaeffer, 2004) propose the Hierarchical Path-Finding A* (HPA*) which aims to reduce the pathfinding complexity in terrains that are represented by grid structures. They do this by abstracting the terrain map into pre-computed clusters, where such clusters are then grouped to form larger clusters. Having an abstracted path found by the HPA*, this path is then refined and returned as the result of the search process. HPA* experiments show that this technique reduces the search effort and produces resultant paths that are within 1% of the optimal path. In addition, this HPA* approach can be explored when different terrain topologies are considered. To do so, however, the pre-computation of clusters is required. The cost of inserting the start and goal points into the abstract terrain representation also increases significantly when additional levels of abstractions are represented in the hierarchy that describes the terrain characteristics.

Sturtevant and Buro (Demyen & Buro, 2006) describe the Partial-Refinement A* (PRA*) to speed up the pathfinding process at the cost of a slight reduction of the path quality. In the PRA*, the virtual environment is discretized into tiles. Adjacent tiles form cliques, which are combined to create more abstract terrain representation levels. This process repeats until a single tile represents the whole environment. The PRA* algorithm performs a search on an abstract terrain representation level until a solution is found. Then, this coarse path solution is projected down and refined on lower levels of the terrain representation back to the original discretization of the virtual terrain.

The above-mentioned hierarchical solutions are focused on 2D grid terrain representations. In contrast, (Pelechano & Fuentes, 2016) focuses on 3D representations for large terrains. Particularly, it explores the navigation mesh polygon-based representation of the terrain (discussed in Chapter 2.3.2). In the end, Pelechano and Fuentes propose the Hierarchical Pathfinding for Navigation Meshes (HNA*). They present a bottom-up method, which is used to create a hierarchical representation, where this structure is based on a k-way partitioning

algorithm. This hierarchical representation is annotated with sub-paths that can be accessed by their Hierarchical NavMesh Pathfinding Algorithm (HNA*) (Pelechano & Fuentes, 2016). In essence, the authors explore a hierarchy of navigation meshes. As a result, the HNA* improves the execution time of the pathfinding process when compared to a non-hierarchical navigation mesh solution. Besides, the benefits of the HNA* solution become more noticeable as the environment dimensions and terrain features increase. Similar to the limitations of the HPA*, the limitation of this HNA* is the step that connects the initial and goal point in the terrain representation, which becomes costlier as the number of abstraction levels of the terrain representation increases.

3.4 Applications with Hierarchical Solutions

There are examples of applications that explored a hierarchical pathfinding solution in large virtual terrains in the literature. One of them is the simulation tool SymSG Border Tactics (Chmielewski, Kędzior, Stąpor, & Kukiełka, 2017), which is a distributed simulation software for tactical training. This system requires a resolution of cells with 10 x 10m² for scenarios as large as 50 x 20km², which is a massive terrain when compared with virtual terrain representations often presented in standard computer games. A hierarchy of regular grid representations with different resolutions is used to model this large virtual environment. In it, the pathfinding algorithm searches for a path in large-scale tiles first. Then it uses the large-scale resultant path to calculate a refined high-resolution result. In addition, a long distance pathfinding algorithm for marine environments is introduced in (Shah & Gupta, 2016). Based on an A* search, the authors proposed to create a hierarchy of the terrain representation in areas of interest. In doing so, they explore visibility graphs which are defined over quadtrees data structures. This means that the virtual terrain is represented in a coarse way when this quadtree is used. Then, a visibility graph is created to describe the obstacles inside each cell of the quadtree. As the free space may change in real-world marine environments, it is essential to use

a structure that allows dynamically updating the representation structure whenever a simulation is being executed. Primarily, the authors tested their hierarchical terrain representation and pathfinding solution in an artificial virtual environment. Then, they applied it to a real-world scenario that was properly generated from nautical chart data. Both the artificial and real-world virtual terrain scenarios represent regions with 100 x 100km² where features with a resolution of 10m in each dimension can be represented.

3.5 DISCUSSION

To perform a global pathfinding search in a virtual terrain, the A* algorithm and its variants are the most explored algorithms due to their optimality and completion (Souissi et al., 2013). Moreover, different A* algorithms are explored in different application problems. Many of these works propose techniques that are focused on finding the fastest path with optimal collision avoidance.

Hierarchical terrain representation and pathfinding techniques are commonly used when dealing with large and complex terrains in real-world applications. Even though navigation meshes representations of the terrain present many benefits and can be used to create a hierarchical representation of the virtual terrain (Pelechano & Fuentes, 2016), to the best of our knowledge, this representation is still not well explored to the modeling of large terrains. Grid-based techniques are more often explored in application scenarios where large terrains are needed (Chmielewski et al., August, 2017) (Shah & Gupta, 2016) due to their construction and data access benefits when compared to navigation meshes (Sturtevant, 2013).

To sum up, it is important to highlight that the terrain representation format have a direct impact on the performance of pathfinding algorithms. To approach this issue, a global pathfinding algorithm for large and realistic real-world terrains is proposed in this work. This algorithm is then applied to a real-world application: the SIS-ASTROS simulation system.

Along with the proposed solution, hierarchical terrain representation techniques are further explored in this work.

Table 2 presents a comparison of our proposed solution with other solutions reviewed in this work and summarize the different characteristics between them.

Table 2 - Comparison of the proposed Solution with different reviewes techniques for global navigation in large virtual terrains.

Global Navigation Technique	Applied to Real Application	Terrain Representation Technique	Search Algorithm Used	Refined Representation Construction
(Botea, Müller, & Schaeffer, 2004)	No	Homogeneous Grids	HPA*	Offline
(Pelechano & Fuentes, 2016)	No	Navigation Meshes	A*	Offline
(Demyen & Buro, 2006)	No	Cliques	PRA*	Offline
(Shah & Gupta, 2016)	Yes	Quadtree / Visibility Graphs	A*	Offline
(Chmielewski et al., 2017)	Yes	Homogeneous Grid	A*	Offline
Proposed Solution	Yes	Quadtree	A*	Execution Time

In addition, different from the hierarchical techniques reviewed ((Botea, Müller, & Schaeffer, 2004), (Demyen & Buro, 2006) and (Pelechano & Fuentes, 2016)), this work proposes the construction of the abstraction in different resolutions of areas of the virtual terrain in execution time. This characteristic give the possibility of changing the resolution of the virtual terrain according to the scenario the simulation system is executing.

In (Chmielewski et al., 2017) the authors proposes the usage of a hierarchical structure of homogeneous grid in a massive terrain inserted in a similar military simulation system. However, they do not present data about the performance of this algorithm. As a result, the quadtree structure is used to represent the virtual terrain. In (Chmielewski et al., 2017) the authors proposes the usage of a hierarchical structure of homogeneous grid in a massive terrain

inserted in a similar military simulation system. However, they do not present data about the performance of this algorithm. The usage of a quadtree structure is similar to the work of (Shah & Gupta, 2016). However, the difference is that visibility graphs are not explored due to the characteristics of the virtual terrain where our proposed solution is inserted. In the work of (Shah & Gupta, 2016) the authors must solve navigation problem in a virtual environment with the presence of large navigation obstacles. In the simulation system where our solution must be applied, there is a presence of a large number of small navigation obstacles. Using a visibility graph to represent them would generate too many redundant connections between navigation obstacles and a large time overhead to insert and connect the initial and goal position of the pathfinding algorithm in this structure.

4 HIERARCHICAL GLOBAL PATHFINDING ALGORITHM: APPLICATION AND VIRTUAL TERRAIN REPRESENTATION

It is important to describe the real simulation system where the proposed Hierarchical Global Pathfinding Algorithm is inserted into for a better understanding of the techniques used in the development of the proposed algorithm. For this reason, relevant characteristics, constraints and requirements of the application in hand are briefly discussed in this chapter. In addition, the structure used to represent the large virtual terrain is also described. As the large dimensions of the virtual terrain is one of the main motivations for this work, it is also important to understand how (and why) the quadtree structure is used to represent the large virtual terrain and how its structure is connected to create a search space for the global pathfinding algorithm.

4.1 APPLICATION

The hierarchical structure for representing the navigation map of large real-world virtual terrain scenarios along with global pathfinding algorithms approached in this work were motivated by the navigation requirements of virtual simulation-based military training exercises that are developed in the SIS-ASTROS virtual tactical simulation system (ASTROS, 2018). In this simulation setting, this training occurs when military agents are involved in large real-world terrain scenarios representing battle situations. To better simulate such virtual training exercises, the SIS-ASTROS simulation system has characteristics of both constructive and virtual simulations (Hodson & Hill, 2014), leading to a new kind of virtual tactical simulation environment as detailed in (Brondani et al., 2018). In effect, the simulations are concerned with tactical aspects of a military operation, mixing both strategic and operational concerns. A simple example of this strategic and operational combination in such navigation framework is the task of determining the best path for a convoy of military vehicles while avoiding a possible retaliation from enemy forces as such forces are represented in a virtual battle terrain. In this situation, there is a strategic aspect behind the selection of routes that are more likely to avoid

agent detection by enemy forces, although such route choices also require the consideration of constraints imposed by the terrain characteristics. In summary, the simulations are concerned with higher-level aspects of a military operation, such as when autonomous agents are explored in constructive simulation scenarios. As in virtual simulations, however, this simulation system also allows users to interact with the virtual environment during the military exercises. As an answer to such constructive and virtual simulation requirements, a (semi) autonomous navigation framework is explored in the SIS-ASTROS simulation system as presented in (Brondani et al., 2017) and (Brondani et al., 2018).

The SIS-ASTROS simulation system has three main architectural components as illustrated in Figure 1: a simulation table in which tactical aspects of the simulation exercises are examined in a real-world terrain representation which is captured in a 2D map view, a wall in which such virtual tactical simulation situations in the virtual terrain scenario is presented in a 3D view, and an instructor terminal in which the full training setup of the simulation exercises is developed by instructors. The global navigation algorithms proposed in this work are inserted in the simulation table through which the users can interact with the simulator, using it as a tool to help the decision making process during the simulation exercise.

During military simulation exercises in the SIS-ASTROS virtual tactical simulator, agents navigate across virtual terrain scenarios (the 2D and 3D ones, simultaneously) in order to execute different application-specific tasks. In this kind of simulation, as presented in (Brondani et al., 2017) and (Brondani et al., 2018), a global hierarchical pathfinding algorithm is responsible for planning routes for these agents, where the resulting routes consider the military doctrine and the tactical actions required by the current battle situation. In the end, the fast computation of these routes while the military exercises are executed in the simulation system requires a global vision of the large real-world virtual terrain scenario, which is the focus of this work.

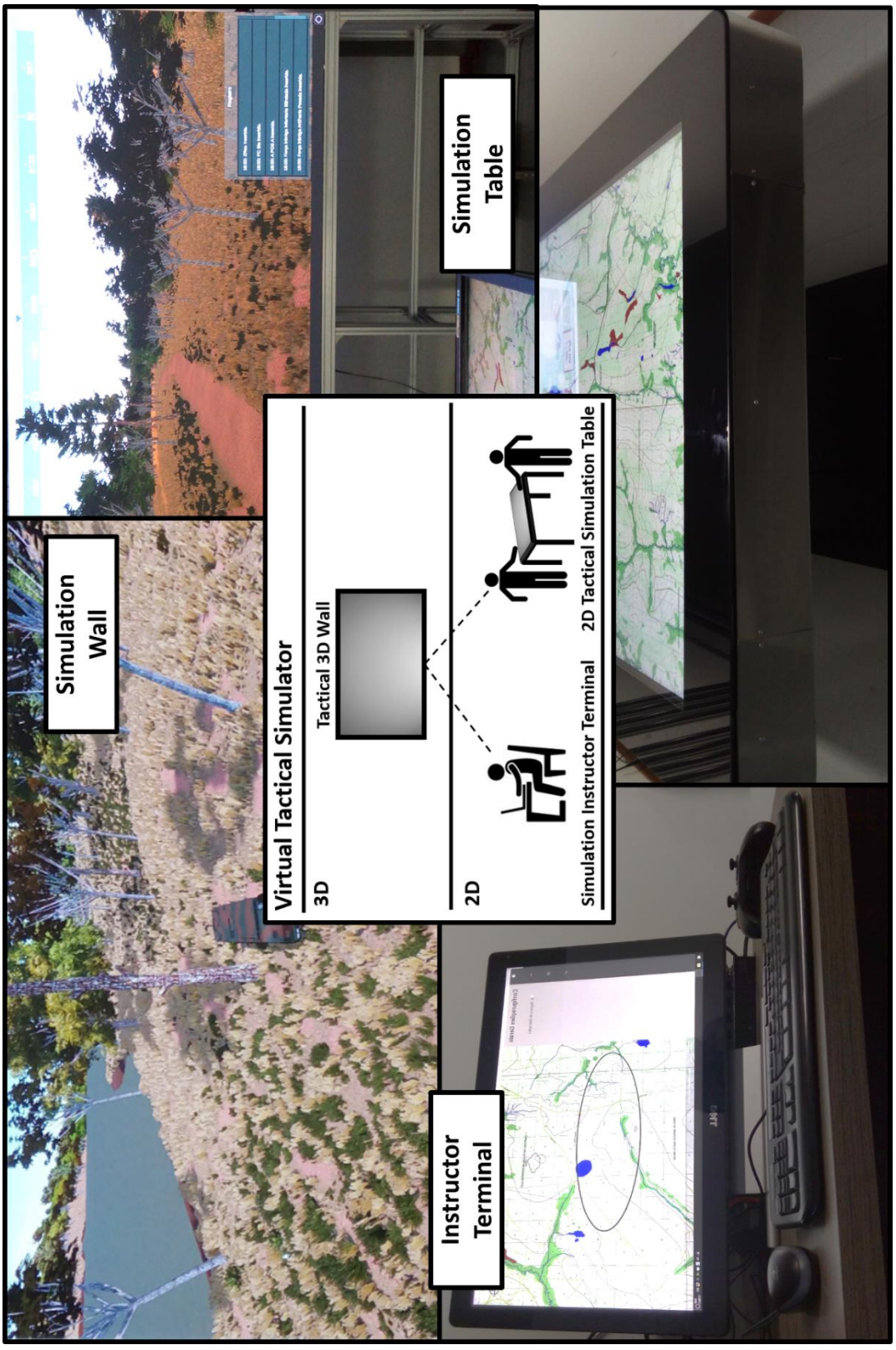

Figure 1- The setup of the SIS-ASTROS (Sis-Astros, 2018)Virtual Tactical Simulation System.

4.2 GLOBAL NAVIGATION SOLUTION: REQUIREMENTS AND CONSTRAINTS

Requirements and constraints of the SIS-ASTROS virtual tactical simulator are considered in the hierarchical global pathfinding algorithm and the navigation map structure proposed in this work. Requirements regarding the representation of large real-world terrains refer to the need of implementing simulation environments with dimensions around 55 × 55 km². While computing a path in such large real-world virtual terrains, which may involve agents to move through both very dense forest areas (in which a high density of navigation obstacles is present) and sparse open fields, it is important to keep a prompt and realistic agent behavior in the resulting simulation system. In this case, if pathfinding algorithms take too long to compute a route for agents to move in such virtual scenarios, the user experience with the virtual simulation environment is affected negatively. Therefore, it is essential to use pathfinding algorithms that are able to find a global path in these large virtual terrains quickly. In addition, constraints regarding the simulation engine used in the SIS-ASTROS project refer to the fact that its simulation system is being projected and implemented on top of the Unity 3D Engine (Technologies, 2018). In general, this is a free and open cross-platform engine that is popularly explored in the development of computer games. For the implementation of navigation algorithms, this engine contains native navigation and physics components. However, due to many graphical requirements regarding the 3D visualization of large simulation terrains used by the SIS-ASTROS simulation system (see (Frasson, Engel, & Pozzer, 2016), (Frasson et al., 2018) and (Backes et al., 2017) for new computer graphic techniques developed to deal with the realistic and real-time visualization of such large terrains), in-house agent navigation implementations inserted in the Unity 3D Engine were developed.

The representation of navigation maps for virtual terrain scenarios with large dimensions is fundamental to the work presented in this work. As the size of terrain environment has a high impact in the performance of navigation algorithms executed on it, this

work proposes the use of a hierarchical approach to both represent navigation obstacles in the virtual terrain and execute a global pathfinding algorithm on this real-world terrain setting. It does not mean that regular grid structures cannot be explored in the representation of navigation maps in large real-world terrain scenarios used in simulation systems. When doing so, however, the use of a regular grid is likely to result in large search spaces that need to be access by pathfinding algorithms. When a high resolution for the terrain representation is used, a regular grid relies on small grid cells to represent the whole area of the virtual terrain even when navigation obstacles are not present in the grid cell bounds. Due to this reason, this quadtree structure was selected to represent the navigation map of the virtual terrain environment in the SIS-ASTROS simulation system. Among other reasons, this well-known data structure allows reducing the search space of pathfinding algorithms (as experiments developed in this work show). In practice, the hierarchical nature of the quadtree structure allows the subdivision of terrain regions when it is necessary to represent the presence of navigation obstacles in them. It means that such hierarchical approach also permits to represent terrain features of interest and navigation obstacles for the simulations in a high degree of detail. The quadtree representation, also have a variant structure called octree (Samet, 1988) that can make a 3D representation of the virtual environment. However, for the global navigation in the considered simulation system, it is not necessary to employ this technique. As this work is focused on the control of agents that move in 2 dimensions only, it is not necessary to consider the possibility of navigation in the virtual terrain in more than these two dimensions. An abstraction of the height data of the virtual terrain in a flat 2D representation is enough to meet the requirements of the navigation in the real application.

From the pathfinding perspective, it is possible to quickly find a coarse path from start to goal using the quadtree navigation map representation due to the optimization of the search

space. To do so, a hierarchical A* pathfinding algorithm combined with a hierarchical representation of the virtual terrain is explored in this work.

4.3 QUADTREE CONSTRUCTION

For the virtual terrain M of $dimension(x, y)$, it is constructed a quadtree-based navigation map representation $Q = (N, E)$ where N is the set of quadtree nodes and E is the set of edges.

A quadtree node is represented by $n_i^{lv} \in N$ where i is the index of the node and lv is the level of this node in the quadtree structure. The maximum depth allowed in the quadtree structure is represented by $maxCoarseLevel$ where $0 \leq lv \leq maxCoarseLevel$. The first level of the quadtree ($lv = 0$) have only one node (n_1^0), which is the first node created and the root of the tree structure. This root node is denoted by n^R.

The set E represent the set of edges that connect the quadtree nodes to create the tree structure. An edge $e = (n_i^{lv}, n_j^{lv-1}) \in E$ connects a quadtree node n of index i in the level lv to another quadtree node n of index j in the level $lv - 1$. Therefore, the node n_i^{lv} is considered a child of node n_j^{lv-1} . This node is always one level below its parent.

From this description, a quadtree node n_i^{lv} has the following properties represented by $n_i^{lv}.property$:

a) A parent node $n_i^{lv}.parent$, which is a quadtree node in the previous level of depth of quadtree structure. This node is connected to its parent through an edge $e = (n_i^{lv}, n_j^{lv-1})$. A parent node does not exist if the considered quadtree node is the root of the structure.

$$n_i^{lv}.parent = n_j^{lv-1}: \; n_i^{lv}.parent = null \; \rightarrow n = \; n^R$$

b) A *bound* $n_i^{lv}.bound$ that represents the area of the virtual terrain covered by the quadtree node n_i^{lv}. The size of the $bound$ of a quadtree node n_i^{lv} is one quarter the size the $bound$ from its parent node n_j^{lv-1}.

$$size\left(n_i^{lv}.bound\right) = (1/4)\ size\left(n_j^{lv-1}.bound\right)$$

c) A set of children nodes C identified by their position regarding their parent node. For $n_i^{lv}.C$ to be a valid set, it must have either four or zero elements.

$$n_i^{lv}.C = \begin{cases} \left\{n_{NW}^{lv+1}, n_{NE}^{lv+1}, n_{SE}^{lv+1}, n_{SW}^{lv+1}\right\} & if\ n_i^{lv}\ is\ not\ a\ "leaf\ node" \\ \emptyset & if\ n_i^{lv}\ is\ a\ "leaf\ node" \end{cases}$$

d) A boolean property named $isLeaf$ that is true if the current quadtree node n_i^{lv} is a "$leaf\ node$". If the set of children is empty ($n_i^{lv}.C = \emptyset$) than the node n_i^{lv} is a "$leaf\ node$".

$$n_i^{lv}.isLeaf = \begin{cases} true & if\ n_i^{lv}.C = \emptyset \\ false & if\ n_i^{lv}.C \neq \emptyset \end{cases}$$

e) A set of neighbor nodes: $Ne = \left\{ne_1^{lv}, ne_2^{lv}, \ldots, ne_i^{lv}\right\}$ where i is the number of neighbor nodes. These neighbor nodes represent connections between the quadtree nodes and they can be of any quadtree level (lv). However, neighbor nodes must always be "$leaf\ nodes$".

$$ne_i^{lv}.C = \emptyset$$

f) A boolean property named $walkable$ that indicates if the quadtree node allows agents to navigate throught it. There are different methods to determine if an agent can navigate through a quadtree node. These methods can be as simple as to consider the presence of any navigation obstacles inside the $bounds$ of a quadtree node or as complex as to consider the density of navigation obstacles inside the $bounds$ of a quadtree node. In this definition, the method that determines if a quadtree node allows the agents to navigate throught it is represented as $isBlocked()$. In addition, only the "$leaf\ nodes$" of the quadtree are considered when determining the value of this property as they are the nodes that form the search space of the global pathfinding algorithm.

$$n_i^{lv}.walkable = \begin{cases} true & if\ isBlocked\left(n_i^{lv}\right) = false \wedge\ n_i^{lv}.isLeaf\ = true \\ false & if\ isBlocked\left(n_i^{lv}\right) = true \vee\ n_i^{lv}.isLeaf\ = false \end{cases}$$

The construction of the quadtree representation structure (Q) starts by representing the whole virtual terrain using a single quadtree node (n^R) and then recursively dividing it into smaller nodes according to a condition function. In this top-down subdivision process involving the analysis of the terrain features of interest, the condition that guides the subdivision of the quadtree nodes is based on the presence of navigation obstacles in virtual terrain areas that are delimited by the bounds of the quadtree nodes. As defined in the SIS-ASTROS project, the static navigation obstacles in the virtual terrain are represented in a hash table structure for quick access. This hash table is represented as $H = (K; O)$ where *K* is the set of keys to access different indexes in the hash structure and *O* is the set of static obstacles present in the virtual terrain. A key K_i is used to find a set of static obstacles $O_i = \{ o_{i1}, o_{i2}, \dots, o_{ij}\}$, where *j* is the number of static obstacles that can be accessed using the key K_i. Therefore, the bound of a

quadtree node ($n_i^{lv}.bound$) is used to determine a key K_i to access the hash table H and to retrieve information about the obstacles O_i present in the area covered by this node bound.

In summary, the core of the quadtree construction algorithm is the function that controls the subdivision of the quadtree nodes (*PartCond*) of the terrain navigation map. As defined in this project, a partitioning condition states that a subdivision in the quadtree structure occurs as long as there are navigation obstacles within the bounds of the quadtree node ($n_i^{lv}.bound$). Alternatively, this partitioning occurs until a defined maximum subdivision level ($maxCoarseLevel$) is reached. It is possible to represent the partitioning condition given a quadtree node n_i^{lv} as:

$$PartCond(n_i^{lv}) = true \qquad if\ O_i \neq \emptyset \wedge lv < maxCoarseLevel$$

Where O_i represents the static obstacles contained in the area covered by $n_i^{lv}.bound$. These obstacles O_i are retrieved from the hash $H = (K; O)$ using a key $K_i = (n_i^{lv}.bound)$.

When this partitioning condition is fulfilled, the quadtree node is subdivided into four children nodes where children nodes have one quarter the size of their parent node. These children nodes are then recursively subdivided to create the quadtree structure (Algorithm 1, from line 5 to line 11).

Algorithm 1 Method that subdivides a quadtree node.

```
1: n_i^lv: the quadtree node to be subdivided
2: procedure SUBDIVIDE(n_i^lv)
3:     if PartCond is true then
4:         if n_i^lv.isLeaf is true then
5:             for i = 1 : 4 do
6:                 cn_i^(lv+1) ← new quadtree node
7:                 cn_i^(lv+1).bound ← new bound(1/4 size(n_i^lv.bound))
8:                 cn_i^(lv+1).parent ← n_i^lv
9:                 Subdivide(cn_i^(lv+1))
10:            end for
11:        end if
12:    end if
13: end procedure
```

Algorithm 1- Pseudocode of the method that subdivides a quadtree node.

In the SIS-ASTROS simulation system, the partitioning condition uses the type and number of navigation obstacles that are present in the used virtual terrain. In practice, navigation obstacles are terrain features of interest for the simulation purposes that block the movement of the agents. Therefore, a quadtree node that represents an area with a large quantity of navigation obstacles is considered blocked ($n_i^{lv}.walkable = false$) and the agents will not be able to navigate across this node. For example, when the presence of a river is detected in the terrain area that is covered by a quadtree node, this type of obstacle alone is enough to indicate that a subdivision of the node should be made. When forests are detected in the virtual terrain, the quadtree construction algorithm considers not only the presence of this type of navigation obstacle, but also the density of vegetation in such forests (i.e. navigability inside this forest region) in order to determine if a subdivision of the quadtree node is necessary. If a large area of the virtual terrain used contains only some sparse trees, for instance, a local navigation algorithm (Brondani, de Freitas, & Silva, 2017) (Brondani et al., 2018) (Menezes & Pozzer, 2018) can safely treat collision avoidance with these sparse obstacles (in the SIS-ASTROS simulation system, such local navigation is based on steering behavior algorithms (Reynolds,

1999)). It is important to notice that the process of constructing the initial navigation map representation using the quadtree structure occurs offline in the SIS-ASTROS simulation system. According to (De Berg, Van Kreveld, Overmars, & Schwarzkopf, 1997), considering N the set of all quadtree nodes, the time complexity for the construction of the quadtree is $O(N)$ time.

4.4 CONNECTION AMONG NEIGHBOR NODES OF THE QUADTREE REPRESENTATION

Pathfinding algorithms explore the search space by analyzing the links among terrain nodes represented in the navigation map of the virtual terrain environment. To structure this search space, the leaf nodes ($n_i^{lv}.isLeaf = true$) of the quadtree are connected to each other. Figure 2 illustrates this connection between the leaf nodes with their four-adjacency. Figure 2.A illustrates the connection between the leaf nodes of the quadtree from a bottom-up perspective, presenting the quadtree structure as an irregular grid. The darker quadtree node is the one being analyzed and the arrows indicate which quadtree nodes are connected to it. In Figure 2.B, the quadtree is presented in its tree structure representation, in which the dashed lines represent the relation between child-parent nodes. The darker node and the darker arrows respectively show the same node and its links as illustrated in Figure 2.A. In Figure 2.B, it is possible to observe the connection between leaf nodes across different levels of the tree structure.

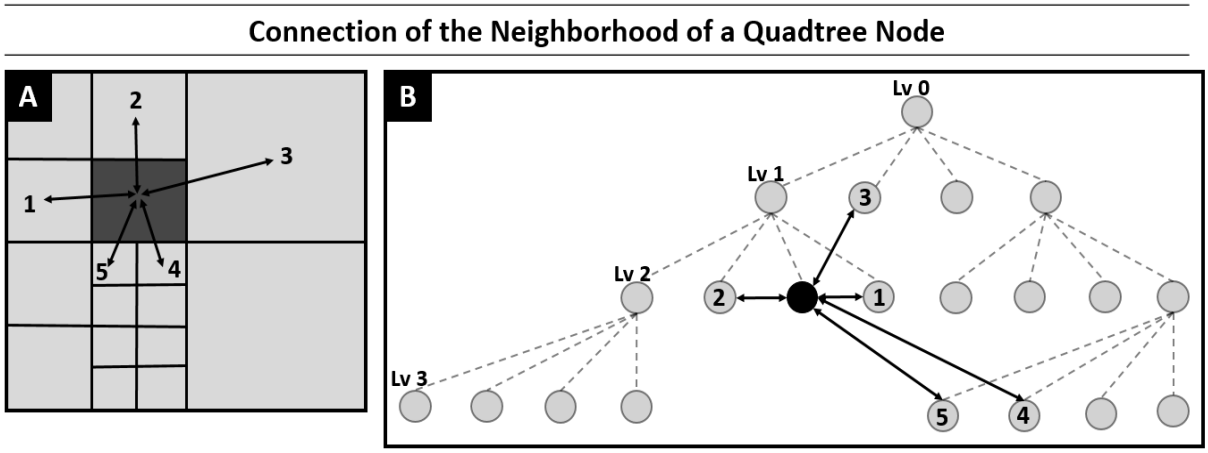

Figure 2 –Illustration of neighborhood connection of a quadtree node. (A) Represents the connection from a bottom-up perspective of the quadtree structure. (B) Represents the connection along a tree perspective.

The neighbors of a quadtree node are determined during the construction of the quadtree-based navigation structure. During the execution of the pathfinding algorithm, the information about the neighborhood of a node is retrieved from this navigation structure. In general, either four-adjacency or eight-adjacency can be implemented according to the terrain representation needs of the simulation system in which this navigation structure is inserted into. In the SIS-ASTROS simulation system, a four-adjacency is used in order to avoid connections between nodes in only one point (diagonal connections), since these diagonal node connections may cause undesired agent navigation behaviors. For example, when there is only one point of connection between two nodes, agents may not be able to traverse between these nodes due to their size and steering restrictions. In practice, it happens when there is a large amount of navigation obstacles around such connection points in the terrain representation structure or when the nodes around these points are blocked ($n_i^{lv}.walkable = false$), for instance.

As an example of the process of connecting nodes, the process of finding the north neighbor nodes from a quadtree node can be illustrated by the pseudocode presented in Algorithm 2 and Algorithm 3. The first step to retrieve a north neighbor of a node is to find a mirrored position (*mPos*) from the center of this node in the north direction of the neighbor that is being retrieved (Algorithm 2, line 8). With this mirrored position (*mPos*), it is possible to

determine which quadtree node contains this position by recursively searching the quadtree nodes from root (n^r) (Algorithm 2, line 18). This node is called mirrored node ($mirrorNode_i^{lv}$). If this mirrored node does not have any children (is a "leaf node") than it is the only neighbor node in the north direction (Algorithm 2, from line 22 to line 23). Otherwise, there are more nodes in the neighborhood in the north direction. So it is necessary to analyze the children of the mirrored node (Algorithm 2, line 25) to find the north neighbors. The method to analyze the mirror node is described in the Algorithm 3. This method finds all the children nodes from the mirror node that are in the north frontier of the node being analyzed (south of $mirrorNode_i^{lv}$) recursively. Then, it returns north neighbors in the form of a list (Algorithm 3, from line 9 to line 11). As exemplified using the north direction, the process described by these two algorithms is repeated when finding south, east and west neighbors of a given quadtree node.

Algorithm 2 Method that finds the neighbors of a node.

```
1: n_i^lv: the quadtree node being analyzed
2: dir: the direction the neighbor is being retrieved
3: procedure GetNeighbors(n_i^lv, dir)
4:     if n_i^lv is n^R then
5:         return null
6:     end if
7:     if dir is north then
8:         mPos ← mirrored position from the center of n_i^lv.bound in the north direction
9:     else if dir is south then
10:        mPos ← mirrored position from the center of n_i^lv.bound in the south direction
11:    else if dir is east then
12:        mPos ← mirrored position from the center of n_i^lv.bound in the east direction
13:    else if dir is west then
14:        mPos ← mirrored position from the center of n_i^lv.bound in the west direction
15:    else
16:        return null
17:    end if
18:    mirrorNode_i^lv ← FindNodeWithPos(mPos, n^R)
19:    if mirrorNode_i^lv is null then
20:        return null
21:    end if
22:    if (n_i^lv.isLeaf) and (mirrorNode_i^lv.walkable is true) then
23:        return mirrorNode_i^lv
24:    else
25:        return FindFrontier(mirrorNode_i^lv, dir)
26:    end if
27: end procedure
```

Algorithm 2 – Pseudocode of the method that finds the neighbors of a node.

Algorithm 3 Method that finds all the neighbor nodes in a frontier of a quadtree node.

```
1: mirrorNode_i^lv: the mirror node from the one being analyzed
2: dir: direction in which the neighbors of the node being analyzed must be retrieved
3: procedure FindFrontier(mirrorNode_i^lv, dir)
4:     frontierNe ← list with the neighbors in the frontier of the node being analyzed
5:     if mirrorNode_i^lv.walkable is true then
6:         if mirrorNode_i^lv.isLeaf is true then
7:             return null
8:         else
9:             if dir is north then
10:                Add FindFrontier(mirrorNode_i^lv.C(n_SE^lv+1),dir) into frontierNe
11:                Add FindFrontier(mirrorNode_i^lv.C(n_SW^lv+1),dir) into frontierNe
12:            else if dir is south then
13:                ...
14:            else if dir is east then
15:                ...
16:            else if dir is west then
17:                ...
18:            end if
19:        end if
20:    end if
21:    return frontierNe
22: end procedure
```

Algorithm 3 – Pseudocode of the method that finds all the neighbor nodes in a frontier of a quadtree node.

The neighborhood of a quadtree node is found during the construction of the quadtree (an offline process). Therefore, if there are no changes in the quadtree during execution, it is only necessary to retrieve this information in the pathfinding algorithm.

5 HIERARCHICAL GLOBAL PATHFINDING ALGORITHM: THE PROPOSED SOLUTION

This work proposes the use of a hierarchical pathfinding algorithm and a quadtree structure for the computation of global navigation paths in the virtual environment. In doing so, it is considered that:

a) The $initialQuadtree$ is the terrain representation structure that is constructed before the simulation exercises initiate. It is constructed (loaded in memory) during the initial setup of simulation.

b) The $initialQuadtree$ has a coarse terrain representation resolution that is determined by a $maxLevel$ parameter. In this resolution, the quadtree structure does not represent all the terrain navigation obstacles in details.

c) The $maxRefinedLevel$ is a parameter that indicates the highest level of resolution that the initial quadtree can reach when the terrain representation refinements are required.

d) The set NR is the set of refined nodes created during the refinement of the $initialQuadtree$. In this set, $nr_i^{rlv} \in NR$ represents a quadtree node nr of index i and maximum depth of rlv where $0 \leq rlv \leq maxRefinedLevel$.

e) The set NeR is the set of neighbor nodes from the refined nodes.

$$NeR = \{ ner_1^{rlv}, ner_2^{rlv}, \dots, ner_i^{rlv} \}, \text{ where } i \text{ is the number of neighbors.}$$

f) The variable $coarsePath$ is a list of leaf nodes from the $initialQuadtree$ representation.

$coarsePath = [\ n_1^{lv},\ n_2^{lv}, \dots ,\ n_i^{lv}\ ]$, where i is the number of nodes in the $coarsePath$.

g) The variable $refinedPath$ is a list of leaf nodes from the refined representation of the $initialQuadtree$.

$refinedPath = [\ nr_1^{rlv}, nr_2^{rlv}, \dots, nr_j^{rlv}\ ]$, where j is the number of nodes in the $refinedPath$.

With these above defined concepts, it is possible to divide the hierarchical global pathfinding algorithm into two phases: the coarse and the refined phases.

5.1 THE COARSE PATHFINDING PHASE

In the coarse phase of the hierarchical global pathfinding, the algorithm performs a search for a path between the start and the goal positions using the $initialQuadtree$ representation of the virtual terrain as the search space. As stated before, this search space is a coarse representation of the whole virtual terrain. From a bottom-up perspective, this representation can be interpreted as an irregular grid since it is formed by the leaf nodes ($n_i^{lv}.isLeaf = true$) of the initial quadtree representation. The result of this pathfinding is the list of quadtree nodes $coarsePath$ that can be of a maximum depth of $maxCoarseLevel$. As presented in section 4.2, the $maxCoarseLevel$ is the maximum depth of the $initialQuadtree$.

Figure 3 illustrates an example of this pathfinding phase using an A* algorithm. Figure 3-A shows that an agent needs to move from A to B in a terrain that is represented by the

quadtree-based navigation map structure. Given a four-adjacency in the execution of the pathfinding algorithm, the resultant $coarsePath$ between these positions is illustrated by the dark nodes in Figure 3-B. In this first phase, the $initialQuadtree$ contains a coarse representation of the terrain only. It means that single obstacles (such as trees, for instance) sparsely distributed in the realistic terrain are not represented in this $initialQuadtree$.

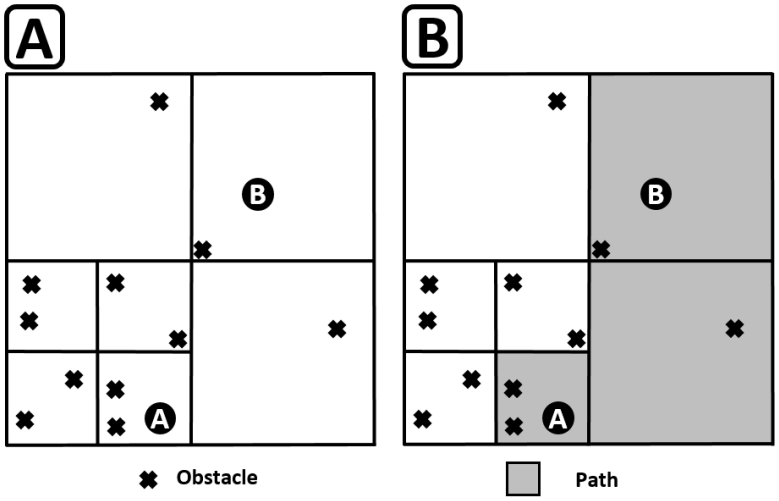

Figure 3- The first phase of the Hierarchical Quadtree Pathfinding: a coarse path is computed.

If a $coarsePath$ can be found during the pathfinding in the coarse phase, it is possible to advance to the next phase of the hierarchical global pathfinding. Otherwise, the algorithm is interrupted and the user is informed that it was not possible to find a path between the start and the goal positions used as input in the simulations.

5.2 THE REFINED PATHFINDING PHASE

In the refined phase of the hierarchical global pathfinding, the nodes belonging to the $coarsePath$ are used as input in the construction of a refined quadtree-based navigation map representation (Algorithm 4, from line 5 to 7). Developed during the simulation execution time, this refinement occurs by subdividing the nodes belonging to the $coarsePath$ into more levels of depth in the quadtree structure. Similar to what happens when the $initialQuadtree$

representation is constructed (section 4.3), the subdivision of the quadtree nodes in the previously computed $coarsePath$ occurs until there are no more terrain obstacles within the bounds of these nodes or until a maximum level of refinement ($maxRefinedLevel$) is reached. It is important to observe that nodes outside the $coarsePath$ are not in the search space of the second pathfinding phase, hence they are not refined. After this refinement step, the refined nodes are connected to their neighbors (Algorithm 4, from line 8 to 10) in the quadtree-based navigation map structure. The method used to connect the neighborhood of these refined nodes is described with more details in Algorithm 5. In this algorithm, the neighbors are retrieved using the methods already explained in section 4.4 and they are stored in the list NeR of the node considering that the algorithm is in its second phase of the hierarchical global pathfinding.

With the refined nodes created and connected to the $initialQuadtree$ structure, the next step is to execute the global pathfinding algorithm again to search for a $refinedPath$. To do this, the refined nodes that contain the start and the goal positions (Algorithm 4, from line 11 to 12) and the coarse path computed in the first pathfinding phase are used as input for the pathfinding algorithm (Algorithm 4, line 13). Different from the first phase, the search space for the pathfinding algorithm in the second phase is not the whole navigation map structure of the virtual terrain since this search space is formed only by the coarse nodes returned in the first phase, along with their refined subnodes. Therefore, the global pathfinding algorithm uses the $coarsePath$ to keep track of the nodes that are part of such reduced search space. At the end of the refined execution of the pathfinding algorithm, a $refinedPath$ is returned as a result allowing agents to move from the start to the goal positions in the virtual terrain. Then, the refinement of the quadtree is destroyed (removed from memory) and the quadtree-based navigation map structure returns to its initial terrain representation state (i.e. the representation of navigation obstacles that is constructed offline, before the simulation exercises started). If it is not possible to find a $refinedPath$, the $coarsePath$ is returned as the result instead.

Algorithm 4 Method that returns the refined path.

```
1: coarsePath: the coarse path from the initial quadtree representation
2: startPos: initial position for the pathfinding
3: targetPos: target position for the pathfinding
4: procedure MakeRefinedPath(coarsePath, initialPos, targetPos)
5:     for each quadtree node n_i^lv in coarsePath do
6:         Subdivide(n_i^lv)
7:     end for
8:     for each quadtree node n_i^lv in coarsePath do
9:         ConnectNeighbours(n_i^lv)
10:    end for
11:    startNode_i^lv ← FindNodeWithPosition(startPos)
12:    goalNode_i^lv ← FindNodeWithPosition (goalPos)
13:    refinedPath ← FindRefinedPath(coarsePath, startNode_i^lv, goalNode_i^lv)
14:    return refinedPath
15: end procedure
```

Algorithm 4 – Pseudocode of the method that returns the refined path.

Algorithm 5 Method that finds and connects the neighbor nodes of a quadtree node.

```
1: n_i^lv: the quadtree node being analyzed
2: refined: true if the node being analyzed is in the refined phase of the pathfinding and false
   otherwise.
3: procedure ConnectNeighbors(n_i^lv, refined)
4:     if n.C is empty then
5:         if refined is false then
6:             n_i^lv.Ne ← range of GetNeighbors(n_i^lv)
7:         else
8:             n_i^lv.NeR ← range of GetNeighbors(n_i^lv)
9:         end if
10:    else
11:        for each child cn_i^lv+1 in n_i^lv.C do
12:            ConnectNeighbors(cn_i^lv+1, refined)
13:        end for
14:    end if
15: end procedure
```

Algorithm 5- Pseudocode that finds and connects the neighbor nodes of a quadtree node.

Figure 4 illustrates the execution of this hierarchical pathfinding algorithm described above. Figure 4-A shows a solid outline around the nodes belonging to the coarse path computed. Figure 4-B illustrates the nodes of the *coarsePath* after refinement, where it is

possible to observe that some terrain obstacles that were not detailed in the coarse node are now represented by refined nodes. The dark nodes in Figure 4-C represent the $refinedPath$ that is found by the global pathfinding algorithm after its second execution finishes. The $refinedPath$ is the path that the agents use to move from A to B in the virtual terrain.

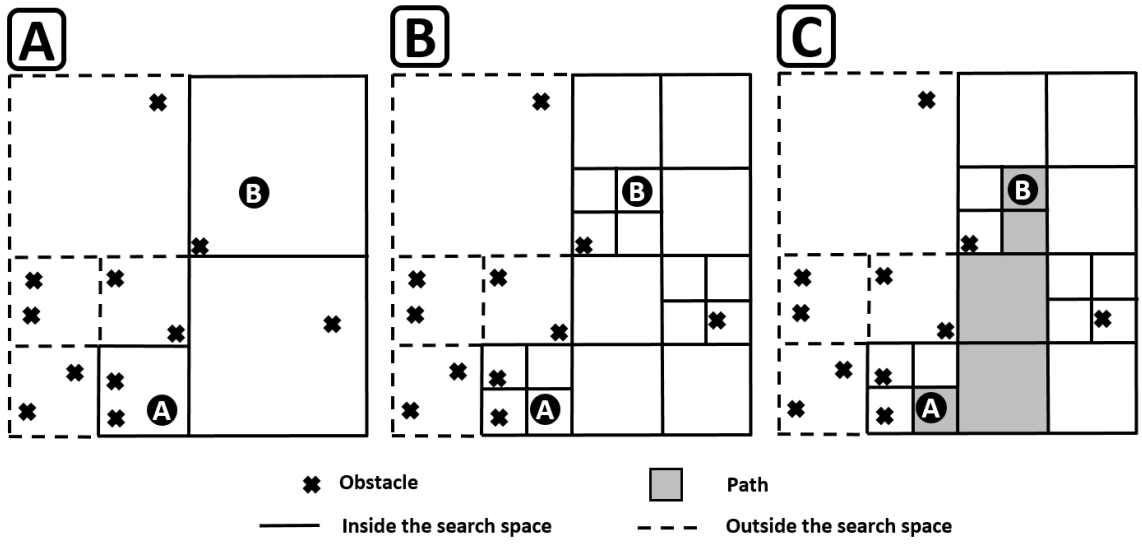

Figure 4 - The second phase of the Hierarchical Quadtree Pathfinding: a fine path is computed.

5.3 DISCUSSION

When this global hierarchical pathfinding algorithm is used, the final path returned reaches a degree of detail that is similar to what can be generated when a non-hierarchical navigation map representation for the same terrain resolution is used. However, the hierarchical pathfinding algorithm requires the analysis of a smaller search space as observed by the execution time evaluations developed in the experiments of this work. Because the refinement created in the initial quadtree representation is destroyed at the end of the second pathfinding phase, only the reduced search space that is explored in the beginning of the pathfinding process is maintained in memory. Different from other hierarchical navigation techniques ((Botea, Müller, & Schaeffer, 2004), (Demyen & Buro, July, 2006)), the possibility of creating refined navigation map terrain representations only when necessary during execution time avoids the

storage of more refined copies of the whole navigation representation structure of the virtual terrain. In many senses, this is relevant to decrease the overall memory footprint of the simulation system in which these navigation techniques are integrated. In addition, it allows the adaptation of the level of detail of the virtual terrain in the refined navigation map representation according to the requirements of the simulation system in which such virtual terrains are inserted in (but also according to the availability or not of detailed GIS-based map data which is used in the construction of such virtual terrains).

6 EXPERIMENTAL METHODOLOGY

The different experiments developed in this work aim to evaluate the impact that the proposed hierarchical approach for global pathfinding and navigation map representation has on the computation of agent routes in large realistic virtual terrains explored in simulation systems. The hypothesis is that the hierarchical pathfinding algorithm and the quadtree-based navigation structure representation of such large virtual terrain environments allow speeding up the computational time required to search for such agent routes.

The experiments were executed in two different scenarios. One of these scenarios is the actual massive terrain (around (55 × 55) km²) used in the SIS-ASTROS simulations system. Figure 5-A shows a part of this massive virtual terrain represented in 2D using the quadtree structure. The quadtree for the realistic terrain is constructed using the GIS data from the real-world terrain. The other test scenario contains a reduced size excerpt (around (3.5 × 3.5) km²) of the virtual terrain used in the SIS-ASTROS simulation system. This reduced scenario is illustrated in Figure 5–B. It is constructed using information about the navigation obstacles from the virtual terrain of the real-life simulation system. This information is used to construct a simpler scenario for the pathfinding algorithm that does not need to contain graphic representations of the virtual terrain and other data used by other components of the simulation system. In Figure 4–B, the dark nodes indicate areas that are blocked for navigation purposes of the simulation-based training exercises. The lighter nodes indicate the quadtree nodes that are *walkable.*

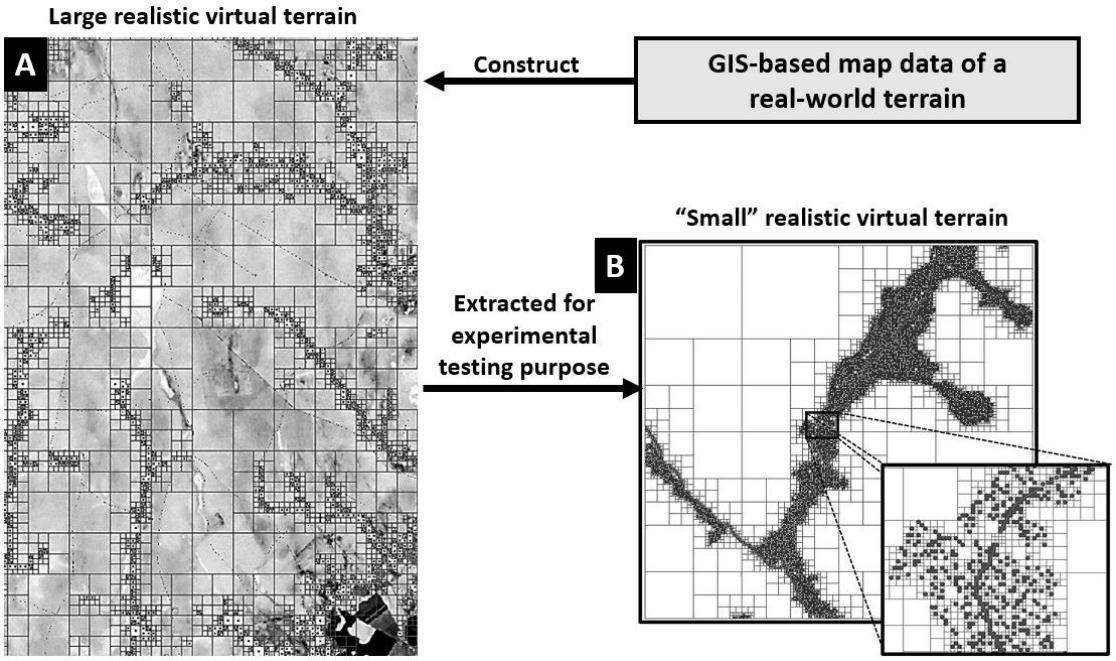

Figure 5 - The irregular (quadtree) representation of a virtual terrain used in the experiments.

In all the experiments developed in this work, the execution time of the global pathfinding algorithm was measured. In effect, this pathfinding algorithm computed agent routes in the "small" and the large virtual terrain scenarios. According to (Nielsen, 1994), there are three important limits of 0.1 seconds, 1.0 second and 10 seconds for the response time in human-computer interaction, as it is the case of users interacting with simulation systems. At 0.1 second, users feel that the system is reacting instantaneously. From this value to about one second, it is the "limit for the users" flow of thought to stay uninterrupted even though they notice the delay." In values between one and ten seconds, users will increasingly feel more disconnected from the system execution. At around ten seconds, users feel that an error has occurred if the system does not provide some form of feedback to inform that it is executing the user requests. In this work, these limits are relevant in the analysis of the execution time of the proposed pathfinding algorithm.

It was defined that the "ideal" execution time for the global pathfinding algorithm in the SIS-ASTROS simulation system ought to be lower than the 1.0 second limit. The percentage

of executions that exceed this response time is measured in order to determine whether the global pathfinding algorithm is able to maintain a consistent response time throughout the tests. The percentage of executions that exceed the time limit of 10.0 seconds is also measured since it allows identifying situations that users may think a problem has occurred in the simulations. So, this 10.0 seconds threshold value regarding pathfinding response time is used to highlight computational responses in which users would lose immersion and, consequently, realism when observing agent navigation behaviors in the simulation-based training exercises. It is also relevant to observe that the time required to construct the initial quadtree-based navigation map representation of these virtual terrains was not computed in the experiments. That is because this process was developed offline, i.e. before the simulations were initiated, which is a moment in which regular users are not involved in the simulation-based training exercises yet.

Different pathfinding approaches and terrain representation techniques are compared in the first experimental scenario in which the "small" terrain excerpt is used. In effect, this terrain scenario is explored to allow the comparison of pathfinding execution times obtained when different navigation map structures are used. The test dataset in this "small" terrain contains around 10,000 pairs of start and goal positions that are used as input in the pathfinding algorithm executions. The second experimental scenario involves the large realistic virtual terrain representation used in the SIS-ASTROS simulation system. In it, the test dataset has around 130,000 pairs of start and goal positions. These positions cover all the surface of this terrain, guaranteeing that most agent routes in such large virtual terrain are used as input in the pathfinding algorithm executed. Using these two virtual terrains with different dimensions, the pathfinding execution time results were measured according to the distances between these start and goal positions. This approach allows one to analyze whether the performance of the hierarchical global pathfinding algorithm proposed in this work can scale up well with the dimensions of the virtual terrain used.

Table 3 summarizes the setups/scenarios of the performed experiments. These experiments were executed in the Unity Engine using a computer with an Intel® Core™ i7-5820K CPU @3.30GHz processor with 16GB of memory and NVIDIA GeForce GTX 750 Ti GPU. It is important to highlight that the execution of the pathfinding algorithms in the Unity Engine is likely to include some time overhead during the execution of the tests and that the parallelization of the pathfinding in the CPU cores is not used.

Table 3 - A summary of the experiments developed in this work.

OVERVIEW OF EXPERIMENTS		
	"Small" realistic virtual terrain scenario	**Large realistic virtual terrain scenario**
Size of the terrain	3.5 x 3.5 km²	55 x 55 km²
Terrain representation techniques	Non-Hierarchical Homogeneous Grid (NHHG); Non-Hierarchical Quadtree (NHQ); Hierarchical Homogeneous Grid (HHG); Proposed Hierarchical Quadtree (PHQ)	Hierarchical Quadtree (PHQ)
Pathfinding measures	Pathfinding execution time; Percentage of pathfinding executions that exceed an ideal response time; Percentage of pathfinding executions that amount to errors in terms of response time; Number of nodes on the resultant path; Average level of the nodes in the resultant path; Sum of the distance between the centers of the nodes in the resultant path;	Pathfinding execution time; Percentage of pathfinding executions that exceed an ideal response time; Percentage of pathfinding executions that amount to errors in terms of response time;
Pathfinding algorithm	Non-hierarchical A* algorithm in non-hierarchical terrain representations; Hierarchical A* algorithm in hierarchical terrain representations;	Hierarchical A* algorithm;
Test dataset	~ 10,000 pairs of start and goal in the virtual terrain representation separated according to different distance intervals	~130,000 pairs of start and goal in the virtual terrain representation separated according to different distance intervals

All the different terrain representation techniques used in the experiments were implemented in our project. In the simplest form of representing the navigation structure of the virtual terrain used in the first experimental scenario, the Non-Hierarchical Homogeneous Grid (NHHG) is based on a flat and square regular grid. For comparison purposes, the cells of this NHHG have the same dimensions as the nodes in the last level of the quadtree representation

structure used in the Non-Hierarchical Quadtree (NHQ), which is seven meters. In effect, seven meters refers to the dimension of regular agents (i.e. military vehicles) appearing in the SIS-ASTROS simulation system. In contrast, the NHQ represents the navigation structure of the virtual terrain scenario in nine levels of resolution (*maxCoarseLevel*). In the last level of this quadtree, the bounds of the quadtree nodes also have around seven meters. In this case, this terrain representation structure, with its nine representation levels, is enough to capture movement actions involving agents in the SIS-ASTROS simulation system. Moreover, the results of the NHQ experiments can be compared to the NHHG experiments.

In the Hierarchical Homogeneous Grid (HHG) used in the first experimental scenario, a coarse regular grid that has cells with around twenty-eight meters is initially created. Then, during the execution of pathfinding algorithms, this grid is refined in a task that is similar to what happens when the PHQ representation is used. Each coarse grid cell is refined to be represented by a homogeneous grid with smaller grid cells. These refined cells have around seven meters, which allows comparing experimental results obtained in these HHG and PHQ different navigation map representations. The homogenous grid refinement is also destroyed after the search, so the terrain representation structure returns to its initial coarse state that contains larger grid cells.

In the Proposed Hierarchical Quadtree (PHQ) representation used in the first experimental scenario, the refinement that occurs in the second phase of the hierarchical global pathfinding algorithm (see section 5.2) subdivides the *coarsePath* found in the initial quadtree structure into nine levels of resolution (*maxRefinedLevel*). Due to this resolution, the pathfinding execution time results over the PHQ can be compared to the ones obtained when the NHQ representation structure is used. It is also relevant to mention that the initial (before refinement) coarse representation of the PHQ structure has seven levels of resolution. This depth is able to represent areas with around twenty-eight meters in the last level

(*maxCoarseLevel*) of the PHQ *initial quadtree* terrain representation (allows comparison with the HHG). It means that the dimensions of the nodes in the last level of the PHQ are not as large as to not faithfully represent both blocked and traversable areas in the "small" virtual terrain environment. As explained before, navigation actions that consider individual obstacles inside these twenty-eight meters square areas are left to local navigation algorithms implemented in the SIS-ASTROS simulation system (Brondani et al., 2017) (Brondani et al., 2018).

The experiments in the second experimental scenario are similar to the ones executed in the "small" scenario. In this second scenario, however, only the PHQ is used to represent the large realistic virtual terrain environment used in the SIS-ASTROS simulation system. Two alternative experiments were performed in this second scenario. In the first experiment in the large terrain (A), the coarse representation of the PHQ has seven levels of depth. The size of the initial search space is presented in Table 3. In its refined representation, the quadtree has nine levels. As the number of levels is similar to the first experimental scenario, this representation allows a comparison with a terrain representation used in the "small" terrain scenario. In the second experiment in the large realistic virtual terrain (B), the quadtree representation of the virtual terrain has the depth used in the SIS-ASTROS simulation system. While the initial coarser quadtree have ten levels of depth, representing areas around (53 × 53) m^2, the refined quadtree representation has twelve levels of depth, representing areas around (13 × 13) m^2. The choice of these different levels of depth was empirically executed in our project using the SIS-ASTROS simulation system as their depths presented a good trade-off between representation accuracy of terrain features of interest and the execution time of the pathfinding algorithm used. In addition, the dimensions of the refined quadtree nodes are still close to: a) the dimensions of the simulated agents; and b) the dimensions of the nodes created in the experiments in the "small" experimental scenario.

In the experiments, each terrain representation configuration generates search spaces with different sizes. Table 4 summarizes theses search space sizes, where the term “leaf cell” is used to describe the cells that form them. These search space sizes have a significant impact in the execution time of the pathfinding algorithm.

Table 4 - Size of the search space (number of leaf cells) in each virtual terrain representation used in the experiments.

Search Space Size of Different Terrain Representation Configurations		
Test Scenario	**Terrain Representation Configuration**	**Number of Leaf Cells**
“Small” realistic virtual terrain	NHHG	262,144
	NHQ	27,694
	HHG	16,384
	PHQ	2,962
Large realistic virtual terrain	PHQ (7 maxLevel, 9 refinedMaxLevel)	34,441
	PHQ (10 maxLevel, 12 refinedMaxLevel)	101,154

Lastly, only a standard A* pathfinding approach is used in the performed experiments. Compared to other pathfinding techniques, such as Dijkstra (Dijkstra, 1959), the A* is a faster algorithm as it expand less nodes in the search space. For this reason, the A* algorithm is used in the experiments even though it is possible to use other search kinds of search algorithms over the navigation structure representation of the virtual terrain. Moreover, other different A* variants can also be used (as the ones cited in Chapter 3), as the proposed solution is generic and does not create constraints for these optimized A* algorithms. In general, it is reasonable to consider that the use of a standard A* algorithm version is sufficient for the purposes of these experiments.

7 EXPERIMENTAL RESULTS FOR THE PATHFINDING EXECUTION

The experiments developed in this work involved the execution of the global A* pathfinding algorithm in different navigation map representations of the "small" realistic virtual terrain scenario. In doing so, the pathfinding execution time was systematically measured. The results obtained are presented as a dispersion graphs in Figure 6, Figure 7, Figure 8 and Figure 9. In these graphs, the execution time results were plotted according to distances from the start and the goal positions in the virtual terrain scenario used as input on the pathfinding algorithm. After, the PHQ is experimented in the large virtual terrain used in the SIS-ASTROS simulation system. Two different resolutions of the quadtree representation of the virtual terrain are used to demonstrate how the PHQ works when the search space size is incremented.

7.1 EXPERIMENTAL PATHFINDING RESULTS IN THE "SMALL" VIRTUAL TERRAIN SCENARIO

Tacking the "small" virtual terrain scenario as reference (i.e. the (3.5 × 3.5) km² terrain scenario), the figures show that there is a significant variation in the pathfinding execution time due to the distribution of the start and goal positions in different areas of the virtual environment. These results demonstrate that the pathfinding algorithm in the NHHG representation (Figure 6) had the worst performance. Even when the results in small distance intervals (lower than 1 km, for instance) had a low execution time (around 40% of the results below 1 second), these execution time values sharply increased as the start and goal distances increased. This is due to not only the large search space that the NHHG representation have, but also to the overhead of refining this homogeneous grid during the execution time of the simulations. Besides, it is possible to observe as the start and the goal distances increased, the number of executions plotted in the graph decreased. This happened because very few pathfinding executions returned a path solution in less than 10 second, which is the "error" time limit defined in this work. The pathfinding execution time in the NHHG (Figure 7), the NHQ

(Figure 8) and the PHQ (Figure 9) representations increased linearly as start and goal distances increased. The HHG is the technique that presented the most abrupt execution time increment. Besides, the pathfinding algorithm in this HHG representation has execution time below the ideal response time of 1 second in very small distances (i.e. less than 500 meters) in the virtual terrain scenario. The pathfinding algorithm in the NHQ representation had some of the executions in larger distances (around 8% for more than 2.5 kilometers) in which the execution time results were below the ideal response time of 1 second. Among the techniques compared in the "small" virtual terrain scenario, however, the use of the NHQ representation was the one that presented the most scattered pathfinding execution time results. This is due to the irregular nature of the NHQ representation, since it has large cells in some terrain regions and very small cells in others. In practice, this NHQ representation creates large search spaces in terrain regions with a high density of navigation obstacles and small search spaces in areas with sparse obstacles. It is relevant to mention that the best pathfinding execution time results were obtained when the PHQ representation was tested. It presented a low and constant execution time behavior in which almost all executions of the pathfinding algorithm had execution time results that were below the ideal response time (99.9% of results below 1 second). Besides, this PHQ representation along with the hierarchical global pathfinding algorithm did not resulted in execution times as scattered as when the NHQ representation was used. That is because only the PHQ cells from the initial *coarsePath* calculated were refined, resulting into a larger search space. It means that the whole navigation map for the virtual terrain was not analyzed in the pathfinding refinement in order to generate a more accurate resulting path for agents.

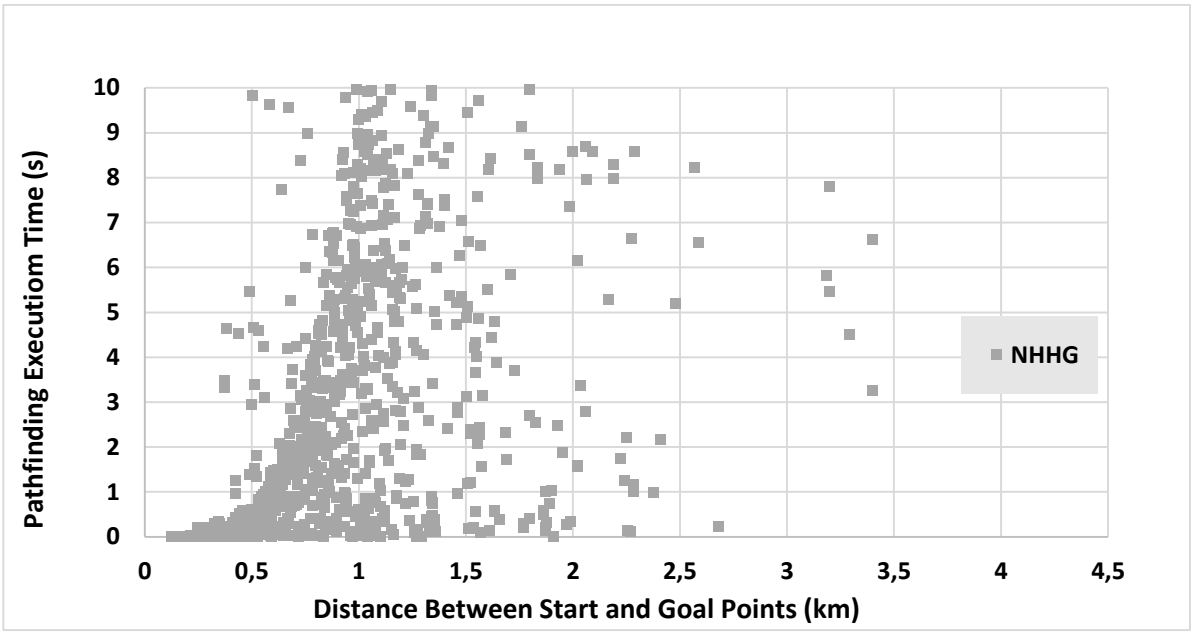

Figure 6 – Distribution of the pathfinding execution time computed using a Non-Hierarchical Homogeneous Grid to represent the virtual terrain in the small realistic test scenario.

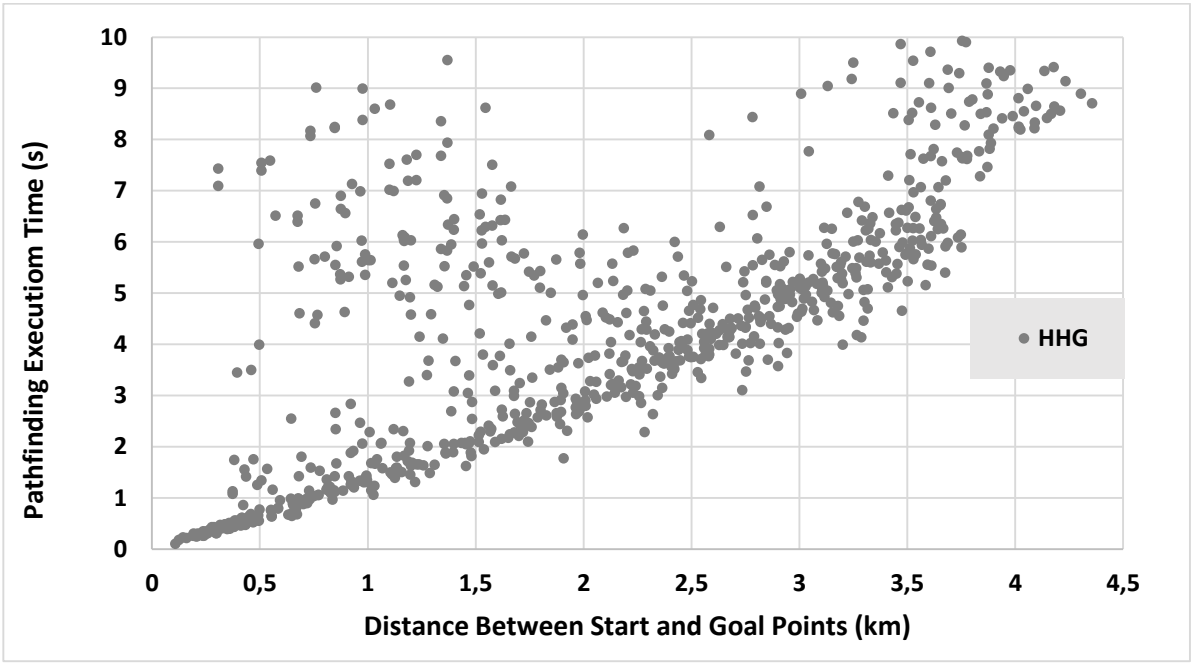

Figure 7 - Distribution of the pathfinding execution time computed using a Hierarchical Homogeneous Grid to represent the virtual terrain in the small realistic test scenario.

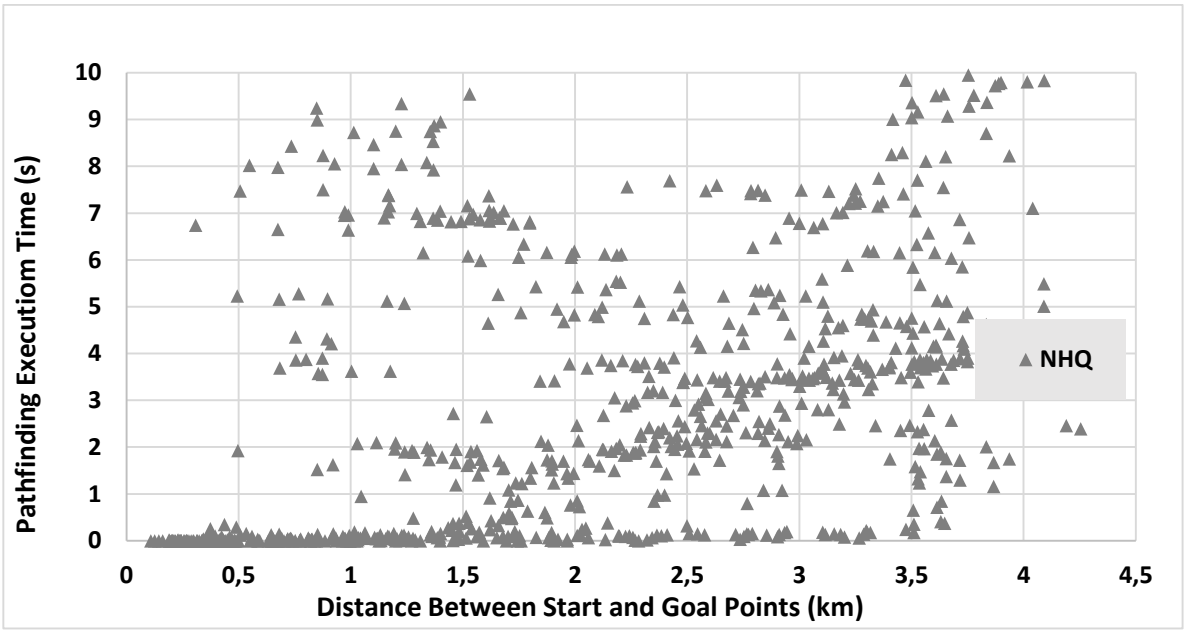

Figure 8 - Distribution of the pathfinding execution time computed using a Non-Hierarchical Quadtree to represent the virtual terrain in the small realistic test scenario.

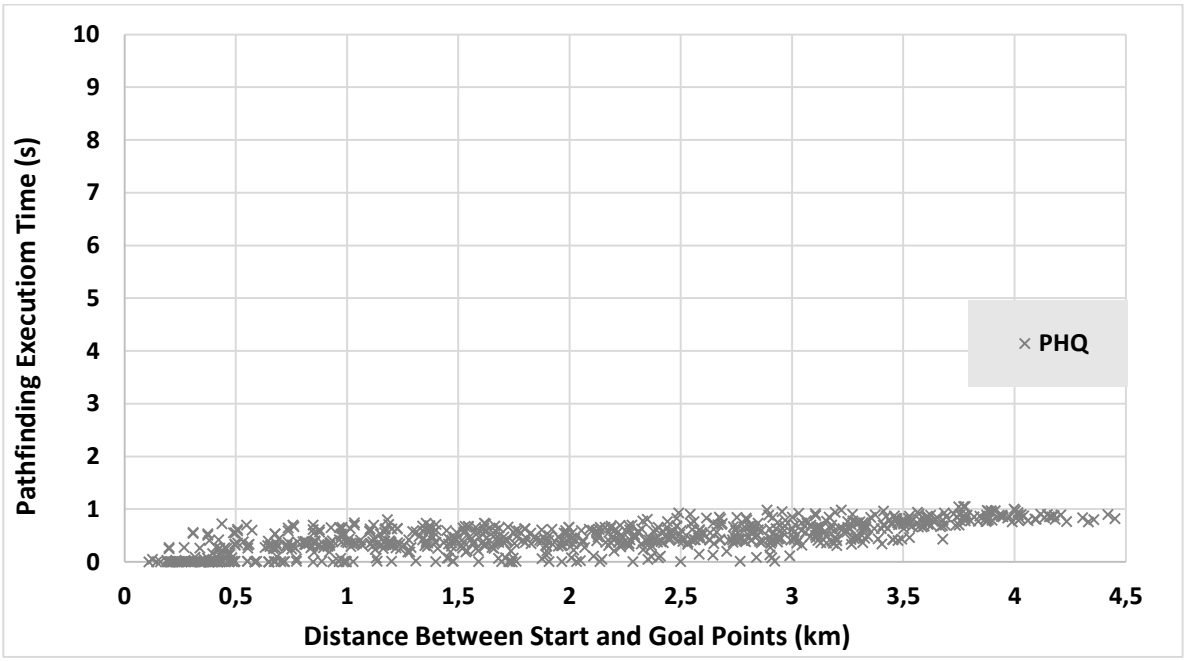

Figure 9 - Distribution of the pathfinding execution time computed using the Proposed Hierarchical Quadtree to represent the virtual terrain in the small realistic test scenario.

For a better understanding of these results, Figure 10 presents graphs that show the amount of pathfinding executions (from a total of 9,900) in different intervals of execution time. This way, it is possible to better visualize the frequency that the pathfinding algorithm is able

to maintain an ideal response time in each terrain representation technique. Figure 10-A shows that, as previous stated, the NHHG had the worst performance of the compared techniques. The distribution graph of the NHQ (Figure 8) shows that there are still many executions within the ideal response time but it is difficult to guarantee this ideal response time as the distance interval increases. This behavior is also illustrated by Figure 10-B where the NHQ still maintained more than half of its pathfinding executions under the ideal response time. Figure 10-C demonstrate that the HHG improves the results from the NHHG as there are more executions within the ideal response time and under the error response time. However, the majority of the executions are still above the error limit. Lastly, Figure 10-D present the results of the PHQ. The PHQ had the best results. Practically all the executions of the pathfinding are within the ideal response time limit.

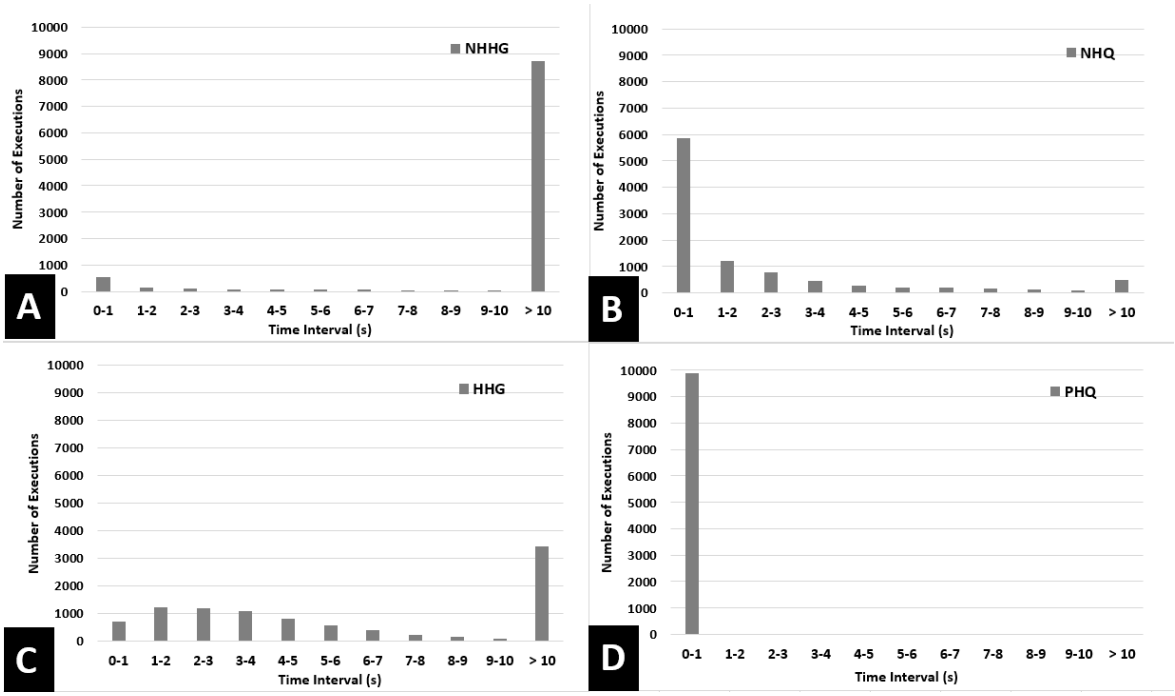

Figure 10 - Number of executions from the dataset with 9.900 pathfinding inputs in different intervals of execution time in (A) the NHHG (B) the NHQ (C) the HHG and (D) the PHQ representations.

In the experiments developed in the “small” virtual terrain scenario, the percentage of pathfinding executions that exceeded the desired response time of 1 second and the “error” time

limit of 10 seconds (explained in section 5.1) were also measured. Results from these experiments are presented in Table 5.

Table 5 - The percentage of pathfinding executions that exceeded the ideal and error responses time in the small realistic virtual terrain scenario.

Percentage of Pathfinding Executions that Exceeded Time Limits (total number of execution: 9,900)		
Virtual Terrain Representations	**Ideal Response Time (1s)**	**Error Response Time (10s)**
NHHG	12.59%	77.13%
NHQ	35.61%	5.09%
HHG	87.74%	1.28%
PHQ	0.10%	0,00%

As presented in Table 5, the percentage of pathfinding searches over the HHG, NHHG and NHQ representations that exceed the optimal response time limit of 1 second is very high. For the majority of the pairs of start and goal positions distributed across the virtual terrain scenario, the pathfinding algorithm was not able to maintain a satisfactory response time. In contrast, pathfinding over the PHQ representation kept a satisfactory response time independent of the start and goal positions in such virtual terrain. In addition, the execution time of the pathfinding algorithm over the PHQ representation did not have any execution that exceeded the time limit of 10 seconds. In contrast, the pathfinding algorithm over the non-hierarchical terrain representation techniques (NHHG and NHQ) had the worst performance in the “error” response time evaluations. For small distances in the virtual terrain scenario, these results demonstrate that a non-hierarchical representation can achieve better results for the pathfinding algorithm than its hierarchical counterpart. That is because they do not have the overhead of refining the representation structure at execution time. For large distance, however, the usage of a hierarchical terrain navigation map representation technique along with the hierarchical pathfinding algorithm brings benefits as they allow to quickly reducing the search space to be explored by the pathfinding algorithm used in the simulation exercises.

It is relevant to notice that the experiments above are focused on the execution time of the pathfinding algorithms. In addition to this, it is also necessary to evaluate whether the path returned has satisfactory quality when using the PHQ representation. This means that the resultant path will contain an accurate description of the navigation obstacles in the virtual terrain, in addition of being the smallest path between the start and goal positions selected. To do this, the PHQ representation is compared to the HHG representation. The HHG representation is used in this comparison because the homogeneous grid represents all regions of the virtual terrain with the same degree of resolution. Moreover, it is also a hierarchical technique as the PHQ. For this reason, the results of the HHG are expected to have a better path quality even when the pathfinding computation is more time consuming. For a satisfactory result, the resultant paths obtained when the PHQ representation is used should not differ too much from the ones obtained when the HHG representation is used. In this evaluation of path quality, a) the number of nodes in the resultant path, b) the average quadtree depth of these nodes and c) the sum of the distance between the centers of each node in the path are measured.

Figure 11 presents the graph comparing the number of nodes in the resultant paths that a obtained when the PHQ and the HHG representations are used. As a quadtree structure represents different regions of the virtual terrain with nodes of different sizes, the number of nodes in the resultant path obtained when the PHQ representation is used is significantly smaller than the ones obtained when the HHG is used. From this scenario, it is necessary to evaluate whether the resultant path obtained when the PHQ representation is used also does not loses quality since it has a small number of nodes.

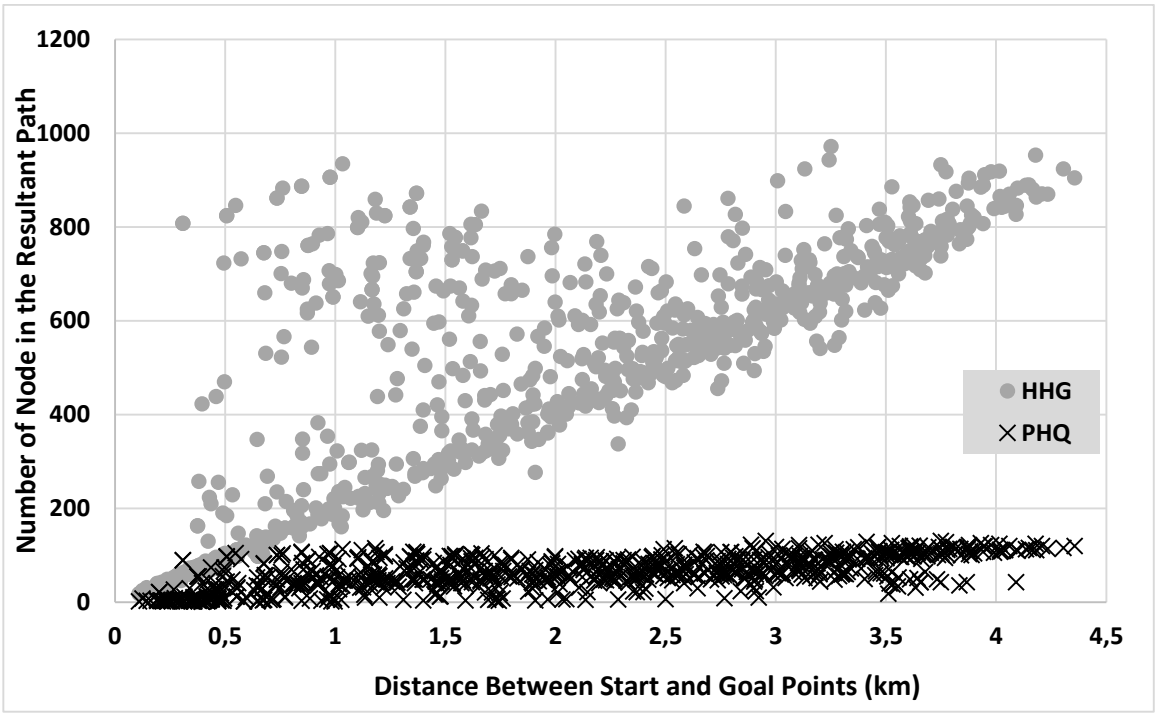

Figure 11 - Comparison between the number of nodes in the resultant path between the PHQ and the HHG.

Figure 12 presents the graph that compares the average depth of the nodes in the navigation map structure that represents the virtual terrain. As the nodes from the HHG representation always have the same resolution, all the results in the HHG graph have the same value. The PHQ samples demonstrate that a high percentage of results maintain an average value of more than seven levels. In these pathfinding searches, this indicates that the pathfinding algorithm needed to refine certain regions of the virtual terrain to better represent the navigation obstacles. At the same time, this pathfinding algorithm was able to detect areas that did not require this refinement. In the HHG representation, these areas without navigation obstacles must be represented with nodes in a high resolution, creating an unnecessary increment of the search space.

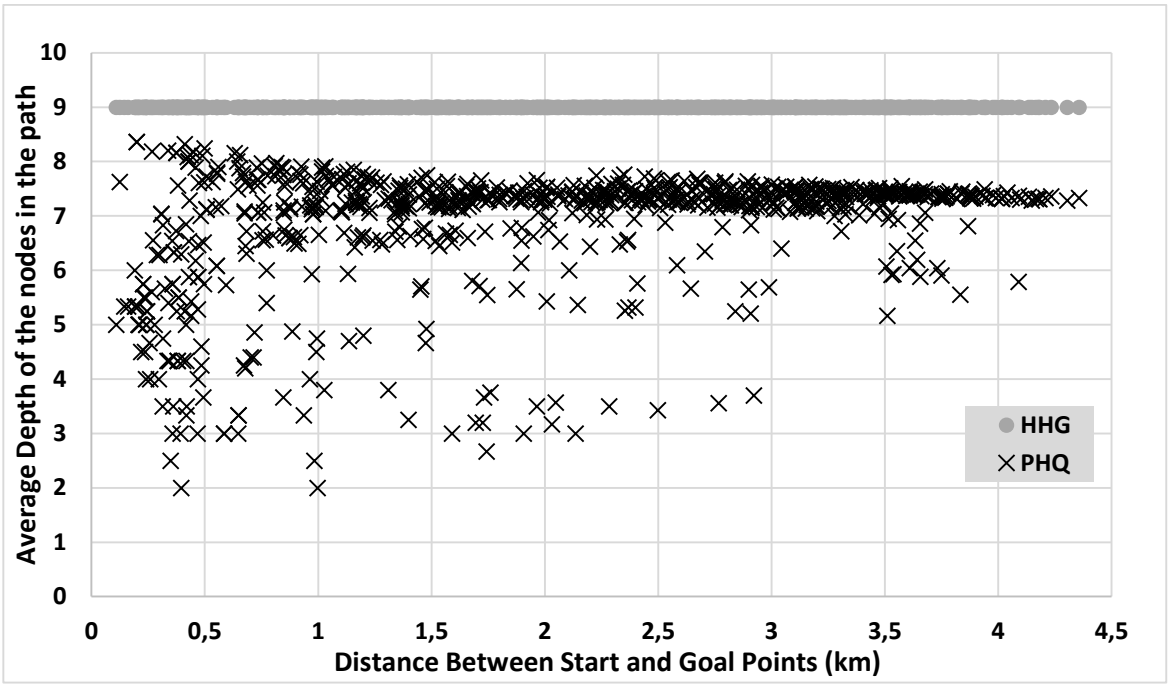

Figure 12 - Comparison between the average level of the nodes in the resultant path between the PHQ and the HHG.

Lastly, Figure 13 presents a graph comparing the difference between the sums of the distances between the centers of the nodes that are part of the resultant path returned by the pathfinding algorithm. The HHG representation has a higher fidelity with the actual pathfinding shortest distance between the start and the goal positions due to the homogeneous nature of its nodes. In contrast, the different size of the nodes in a quadtree representation can misrepresent the real distance between two positions in the virtual terrain scenario. For example, when the agent navigates across the center of each node in the resultant path, it may need to navigate additional unnecessary distances to reach the center of larger nodes. The hierarchical nature of the PHQ representation should minimize this distance error. From the graph in Figure 13, it is possible to observe that there is a difference between the paths obtained when the HHG and the PHQ representation were used. In the majority of the executions, the PHQ presented a smaller path between initial and goal points. However, these results are not necessarily better than the HHG. The large nodes present in the hierarchical quadtree representation may result in smaller paths but this path can be less accurate than the ones from the HHG. This happens when the

large nodes abstract information about sparse navigation obstacles in the virtual terrain or when it does not describe characteristics of the terrain precisely.

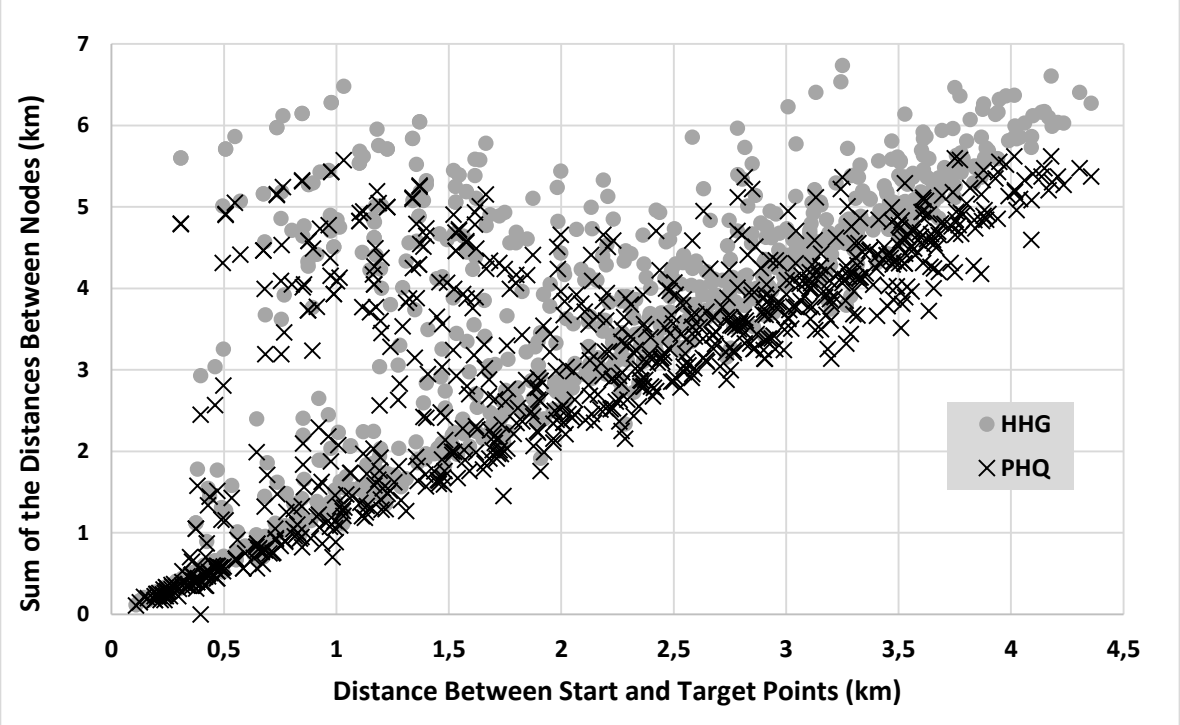

Figure 13- Difference between the sums of the distances between the centers of the nodes that are part of the resultant path from the pathfinding algorithm in the HHG and PHQ.

To illustrate the side effects of the irregular representation of the terrain, Table 6 presents the percentage of the pathfinding executions in which the difference between the computed PHQ and HHQ paths were less than a defined distance limit (defined as 10% of the maximum distance between the start and the target positions). These results are separated in different intervals, according to the distance between start and target positions used by the pathfinding algorithm. Each row in the table represents the acquired result from around 2500 runs for each case.

Table 6 - The percentage of pathfinding executions where the difference between the paths generated when the PHQ and HHG representations used were within an acceptable limit.

Distance (d) between start and target positions in the (3.5 × 3.5) km² terrain scenario	**Distance error between the computed PHQ and HHG paths**	**Percentage of executions in which the distance differences between computed PHQ and HHQ paths is below the error**
d < 1 km	100 m	68%
1 km ≥ d < 2 km	200 m	33%
2 km ≥ d < 3 km	300 m	24%
3 km ≥ d < 5 km	500 m	8%

Table 6 demonstrates that there is a loss in path quality (i.e. in the "shortest" distance between start and goal positions). As observed in Figure 12, this difference increases as the distance between start and goal positions increases. In small distances (around 1 km) both results are very similar. Despite these distance deviations from the actual shortest paths, the gains in speed for the global A* pathfinding algorithm computations greatly compensate the amount of loss in the quality of the resultant path when comparing to a technique that uses a homogeneous representation of the virtual terrain.

7.2 EXPERIMENTAL PATHFINDING RESULTS IN THE LARGE VIRTUAL TERRAIN SCENARIO

Tacking the large virtual terrain scenario as reference (i.e. the (55 × 55) km² terrain scenario), Figure 14 presents the distribution graph of the pathfinding execution time obtained when the PHQ representation was used in a first experimental setting. In this navigation structure representation, while the coarse quadtree representation for this large virtual terrain has 7 levels of depth, the refined quadtree representation has 9 levels of depth (i.e. the level of detail that a *coarsePath* is analyzed). For comparison purposes, these are the same depth levels used in the hierarchical representation in the "small" terrain test scenario.

Figure 14 presents a sample of the results of the pathfinding algorithm. These results were obtained when different distance intervals were used as input in the pathfinding executions. These distances refer to the start and the goal positions dispersed in all regions in

this large realistic virtual terrain. The graph in Figure 14 shows that the PHQ representation was able to maintain most of the pathfinding execution results below the optimal response time of 1 second. These optimal pathfinding response time results were obtained for distances lower than 30 kilometers between the start and the goal positions. For distances higher than this, some pathfinding executions (around 12%) still presented response time results that were lower than the 1 second limit, although the execution time results became more scattered in these long distances. The graph in Figure 14 also shows that the pathfinding algorithm is still able to maintain a quick execution time behavior as distances between the start and the goal positions increased. This fact demonstrates that the PHQ representation allows such pathfinding algorithm to scale the execution time well with the increment in the size of the virtual terrain scenario.

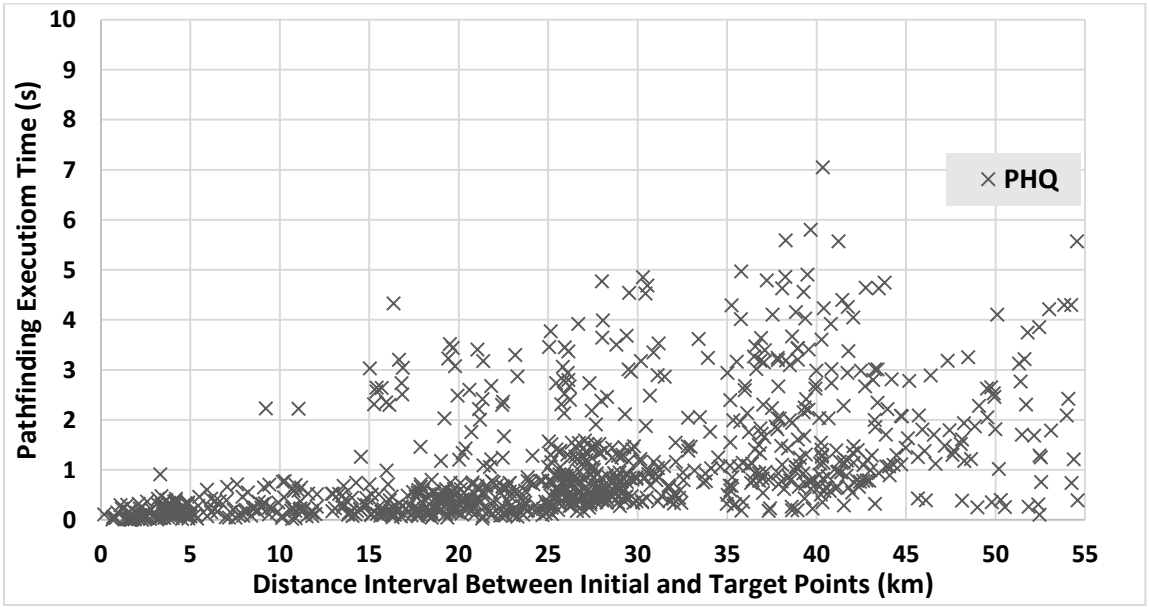

Figure 14 - The distribution of the execution time of the hierarchical pathfinding algorithm in the large realistic virtual terrain scenario represented by the PHQ where *maxLevel* = 7 and *maxRefinedLevel* = 9.

Figure 15 presents the distribution graph of the pathfinding execution time in the PHQ representation of the large terrain scenario. In this experimental setting, however, the setup of the quadtree is the same used when this terrain scenario is loaded in the SIS-ASTROS

simulation system. This means that the coarse quadtree representation has 10 levels of depth and the refined quadtree representation have 12 levels of depth. It is possible to observe that a high percentage of the results until around 30 km of distance between the start and the goal positions had an execution time below the optimal response time of 1 second (around 41%). After this point, the results of pathfinding execution time became much more scattered. In this case, it is not only the distance between the start and the goal positions that influence the execution time of the hierarchical pathfinding algorithm. The resolution of the obstacles' description in the virtual terrain navigation is also a factor that greatly influences the pathfinding performance. In addition, even with more scattered results in large distances, it is possible to observe that these results do not have a trend that indicates a sharp increment in the pathfinding execution time.

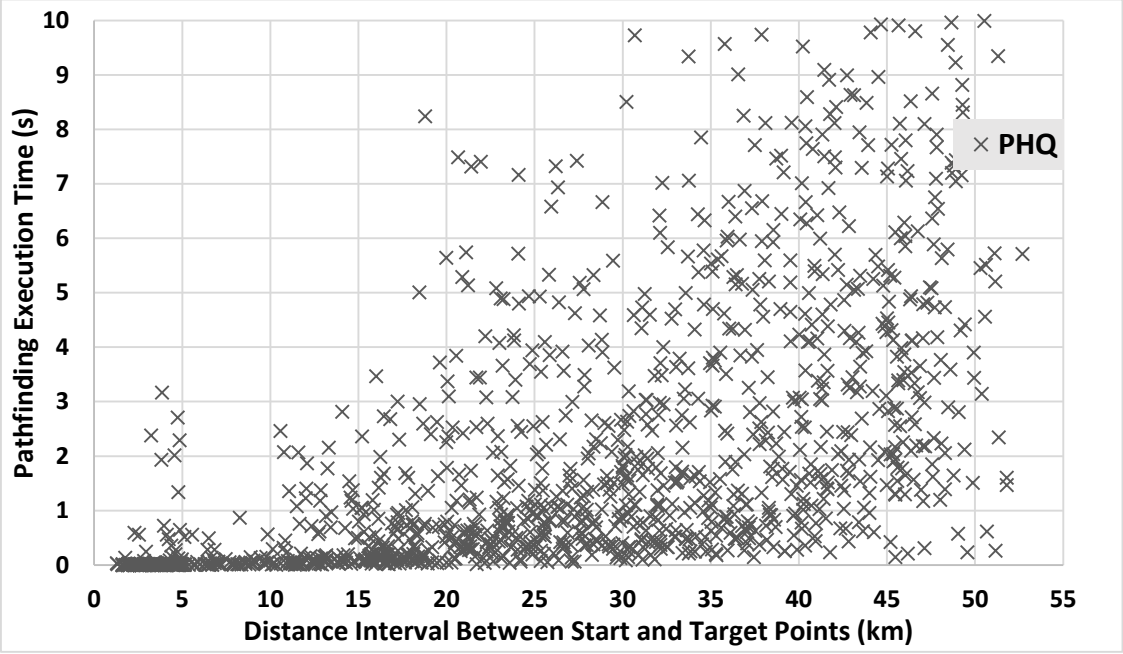

Figure 15- The distribution of the execution time of the hierarchical pathfinding algorithm in the large realistic virtual terrain scenario represented by the PHQ where *maxLevel* = 10 and *maxRefinedLevel* = 12.

Table 7 presents the percentage of pathfinding executions that exceeded the desired response time and the percentage of pathfinding executions that exceeded the "error" response

time limit. These results were obtained when the two PHQ configurations (i.e. 7 to 9 levels and 10 to 12 configuration levels) were used in the navigation map representation of the large virtual terrain. These pathfinding execution time results show that the PHQ representation and the hierarchical A* pathfinding algorithm can maintain a significant percentage of the executions (more than 60% and 80% in each PHQ configuration) within the optimal response time limit of 1 second. However, due to the large number of distance intervals in this massive virtual terrain and the presence of areas with a large density of navigation obstacles in it, the percentage of pathfinding executions that exceed the optimal response time threshold was not as low as it was expected after the tests in the "small" terrain scenario. Nevertheless, considering the dimensions of the large virtual terrain in the SIS-ASTROS simulation system, the small increment in the pathfinding execution time that was obtained in the experiments is a relevant result to be highlighted. In addition, only 1.24% and 7.23% of pathfinding executions in each configuration of the PHQ representation could be interpreted as an "error" since their response time were too high to be accepted by simulation system users.

Table 7 - The percentage of pathfinding executions that exceeded the ideal and error responses time in the large real-world virtual terrain scenario.

Percentage of Pathfinding Executions that Exceeded Time Limits		
Virtual Terrain Representations	**Ideal Response Time (1s)**	**Error Response Time (10s)**
PHQ (maxLevel = 7/ maxRefinedLevel = 9)	19%	1.24%
PHQ (maxLevel = 10/ maxRefinedLevel = 12)	39.19%	7.23%

In conclusion, similar to the HPA* technique detailed in (Botea *et al.*, 2004), this work details the exploration of a hierarchical grid-based navigation map structure for the virtual terrain representation. It also proposes the exploration of a hierarchical global A* pathfinding algorithm to speed up the search of agent routes in such virtual terrains. Different from the HPA*, which explores the use of homogeneous grids, this work relies on a quadtree structure

aiming to optimize pathfinding time executions in large virtual terrain environments. In contrast with the techniques proposed in (Botea *et al.*, 2004) and (Pelechano & Fuentes, 2016), this work describes how to explore refined abstractions of the virtual terrain during execution time. Despite a possible overhead due to this execution time refinement in the quadtree structure, the hierarchical pathfinding algorithm maintains a satisfactory response time when long distance paths have to be computed. Most importantly, the PHQ representation along with the hierarchical A* global pathfinding algorithm are able to mitigate the path planning time for agent routes in large virtual terrains. According to the experiments here reported, the techniques detailed in this work also delivery optimal and scalable global pathfinding response time results, which are significant results for large realistic terrain scenarios as required by simulation systems. The downside of this solution is that the resultant path loses quality in areas with sparse obstacles since these areas are represented by few large cells in the navigation map structure. For this reason, when agents are required to move through these large quadtree nodes, they may not navigate through the fastest and smallest path they could take. In addition, for a substantial interval of distances between the start and goal positions, the user of the simulation system may experience a small delay until the resultant path is calculated.

8 CONCLUDING REMARKS

The exploration of large realistic virtual terrain environments is fundamental to improve the training effectiveness of simulation systems. To promote the systematic use of such large virtual environments, this work approaches the challenge of designing and implementing time-framed navigation algorithms for agents involved in virtual simulation-based training exercises. In addition to reviewing important terrain representation concepts and pathfinding algorithms, this work is focused on pathfinding algorithms that explore the optimized representation of the navigation map structure of the virtual terrain scenario in order to detect and avoid collision with obstacles inserted into the scenario.

First of all, the work shows that a quadtree-based structure allows speeding up the global pathfinding search since it hierarchically describes the navigation map representation of the virtual terrain. This hierarchical representation structure permits representing terrain regions having different densities of navigation obstacles since representation cells having different resolutions (terrain sizes) are explored. In practice, the hierarchical global pathfinding algorithm discussed a) generates a coarse path using the quadtree representation and c) refines the nodes of this coarse path. During execution time, this refinement subdivides the coarse nodes from the initial path in further quadtree levels, resulting in a more detailed representation of the navigation obstacles in the region covered by these nodes. Then, this hierarchical pathfinding algorithm uses only these refined nodes to compute a route with a higher quality than the coarse initial path.

In this work, the proposed navigation techniques are compared to other hierarchical and non-hierarchical virtual terrain representation approaches. As distances from start to goal positions used by the pathfinding search algorithm were increased in the several performed experimental scenarios, the proposed pathfinding technique had execution time results within

the response time limit defined in the majority of the distance samples retrieved from the test datasets used. Most importantly, these techniques were also systematically tested in the large realistic virtual terrain used in the real-life SIS-ASTROS simulation system. Experimental results indicate that the resultant path may lose some quality and the simulation user may experience some delay during navigation interactions with the simulation system as the scale of the virtual terrain navigation map and distance between initial and target positions is increased. However, it also shows that the global A* pathfinding execution time over the Proposed Hierarchical Quadtree representation is not increasing abruptly. This pathfinding execution time increment indicates that such navigation techniques can be scaled to even larger virtual scenarios and, therefore, it is a reliable approach to the representation of the navigation map structure of large virtual terrains used in different kinds of simulation applications.

Similar to the HPA* technique detailed in (Botea et al., 2004), this work details the design, implementation and test of a hierarchical grid-based navigation map structure for virtual terrain representation. Different from the HPA*, which explores the use of homogeneous grids, this work relies on an irregular grid structure aiming to decrease the pathfinding execution time in virtual terrain environments. In contrast with the techniques proposed in (Botea et al., 2004) and (Pelechano & Fuentes, 2016), this work describes how to explore refined abstractions of the virtual terrain during execution time. Despite a possible overhead due to this execution time refinement in the quadtree structure, the hierarchical global A* pathfinding algorithm used maintains a satisfactory response time when long distance paths have to be computed. Most importantly, the Proposed Hierarchical Quadtree representation along with the hierarchical A* global pathfinding algorithm are able to mitigate the path planning time for agent routes in large virtual terrains. In summary, a consistent contribution to efficient path planning in simulation systems that consider large terrains is presented here. This contribution is consubstantiated by the proposal of the hierarchical quadtree terrain representation and the actual algorithms that

explores this structure to perform the pathfinding computations. These elements not only amount to a step-ahead compared to what is found in the literature in this area, but they also amount to a technical contribution that can be directly employed in any simulation system considering large terrains and similar requirements in terms of realism.

Directions for future work aim to adjust the proposed hierarchical pathfinding and terrain navigation map solution to allow dynamic changes in the navigation obstacles (e.g. modifications in the terrain structure by itself) during execution time. Ways of reducing the shortest distance errors in the hierarchical pathfinding computations without increasing too much the pathfinding execution time can also be investigate in the future. In addition, the hierarchical quadtree approach can be extended to the representation of large 3D virtual simulation environments. Having such kind of dynamic 3D virtual terrain navigation map representation, we aim to compute time-framed global pathfinding for both terrestrial (2D) and aerial (3D) agents inserted into virtual simulation scenarios.

REFERENCES

ABD ALGFOOR, Z.; SUNAR, M. S.; KOLIVAND, H. A Comprehensive Study on Pathfinding Techniques for Robotics and Video Games. International Journal of Computer Games Technology, v. 2015, p. 11, 2015. ISSN 1687-7047.

AMMAR, A. et al. Relaxed Dijkstra and A* with Linear Complexity for Robot Path Planning Problems in Large-scale Grid Environments. Soft Computing, v. 20, n. 10, p. 4149-4171, 2016. ISSN 1432-7643.

ANGUELOV, B. Video game pathfinding and improvements to discrete search on grid-based maps. 2011. (PhD Diss.). Faculty of Engineering, Built Environment and Information Technology, University of Pretoria, Pretoria, South Africa.

BACKES, G. C. et al. Rendering of Large Textures for Real-Time Visualization. Sociedade Brasileira de Computação (SBC) - Proceedings of Simpósio Brasileiro de Games e Entretenimento Digital (SBGames) 2017, Curitiba, Brazil.

BAYAT, F.; NAJAFINIA, S.; ALIYARI, M. Mobile robots path planning: Electrostatic potential field approach. Expert Systems with Applications, v. 100, p. 68-78, 2018. ISSN 0957-4174.

BOTEA, A.; MÜLLER, M.; SCHAEFFER, J. Near optimal hierarchical path-finding. Journal of Game Development, v. 1, n. 1, p. 7-28, 2004.

BRONDANI, J. R.; DE FREITAS, E. P.; SILVA, L. A. A task-oriented and parameterized (semi) autonomous navigation framework for the development of simulation systems. 21st International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES 2017). Marseille, France: Procedia Computer Science - Elsevier. 112: 534-543 p. 2017.

BRONDANI, J. R. et al. Semi-autonomous navigation for virtual tactical simulations In the military domain. 8th International Conference on Simulation and Modeling (SIMULTECH). Porto, Portugal. 2018: 443 - 450 p. 2018.

CHMIELEWSKI, M. et al. High detail terrain models and multiresolution path finding algorithms for Border Guard constructive simulator. A study of effective movement algorithms in high resolution simulation environment. Mathematics and Computers in Sciences and in Industry (MCSI), 2017 Fourth International Conference. Crete, Greece: IEEE: pp 270-280 p. 2017.

CIL, I.; MALA, M. A multi-agent architecture for modelling and simulation of small military unit combat in asymmetric warfare. Expert Systems with Applications, v. 37, n. 2, p. 1331-1343, 2010. ISSN 0957-4174.

COWLING, P. I. et al. Search in real-time video games. Dagstuhl Follow-Ups. Wadern, Germany: Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik. 6 2013.

CUI, X.; SHI, H. A*-based pathfinding in modern computer games. International Journal of Computer Science and Network Security, v. 11, n. 1, p. 125-130, 2011. ISSN 1738-7906.

DANIEL, K. et al. Theta*: Any-angle path planning on grids. Journal of Artificial Intelligence Research, v. 39, p. 533-579, 2010. ISSN 1076-9757.

DE BERG, M. et al. Computational geometry. Berlin, Heidelberg: Springer, 1997. 1-17.

DEMYEN, D.; BURO, M. Efficient triangulation-based pathfinding. AAAI'06 Proceedings of the 21st national conference on Artificial intelligence Boston, Massachusetts. 1: 942-947 p. 2006.

DIJKSTRA, E. W. A note on two problems in connexion with graphs. Numerische mathematik, v. 1, n. 1, p. 269-271, 1959. ISSN 0029-599X.

DOOMS, A. Parallel multi-agent path planning in dynamic environments for real-time applications. 2013. (MA Thesis). Faculty of Engineering and Architecture, Ghent University, Gent, Belgium.

ENGEL, T. A.; POZZER, C. T. Shape2River: a tool to generate river networks from vector data. Sociedade Brasileira de Computação (SBC) - Proceedings of Simpósio Brasileiro de Games e Entretenimento Digital (SBGames), 2016, São Paulo, Brazil.

FLETCHER, J. Education and training technology in the military. Science, v. 323, n. 5910, p. 72-75, 2009. ISSN 0036-8075.

FRASSON, A.; ENGEL, T. A.; POZZER, C. T. Improving Terrain Visualization Through Procedural Generation and Hardware Tessellation. Sociedade Brasileira de Computação (SBC) - Proceedings of Simpósio Brasileiro de Games e Entretenimento Digital (SBGames), 2016, São Paulo , Brazil.

FRASSON, A.; ENGEL, T. A.; POZZER, C. T. Efficient screen-space rendering of vector features on virtual terrains. Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games, 2018, Vancouver, Canada. ACM. p.7.

FUENTES, C. Hierarchical Path Finding to Speed up Crowd Simulation using Navigation Meshes. Latin American Journal of Computing Faculty of Systems Engineering Escuela Politécnica Nacional Quito-Ecuador v. 2, n. 1, 2015.

HODSON, D. D.; HILL, R. R. The art and science of live, virtual, and constructive simulation for test and analysis. The Journal of Defense Modeling and Simulation, v. 11, n. 2, p. 77-89, 2014. ISSN 1548-5129.

KAPADIA, M.; BADLER, N. I. Navigation and steering for autonomous virtual humans. Wiley Interdisciplinary Reviews: Cognitive Science, v. 4, n. 3, p. 263-272, 2013. ISSN 1939-5086.

KHATOON, S.; IBRAHEEM, I. Autonomous mobile robot navigation by combining local and global techniques. International Journal of Computer Applications, v. 37, n. 3, p. 1-10, 2012.

KOEFOED-HANSEN, A.; BRODAL, G. S. Representations for Path Finding in Planar

Environments. 2012. (PhD Diss). Aarhus Universitet, Datalogisk Institut, Aarhus, Denmark.

KOENIG, S.; LIKHACHEV, M. D* Lite. AAAI Conference on Artificial Intelligence - The Fourteenth Innovative Applications of Artificial Intelligence Conference on Artificial Intelligence (IAAI-02) Edmonton, Alberta, Canada: 476 - 483 p. 2002.

KORF, R. E. Real-time Heuristic Search. Artificial Intelligence, v. 42, n. 2-3, p. 189-211, 1990. ISSN 0004-3702.

KRUSE, T. et al. Human-aware robot navigation: A survey. Robotics and Autonomous Systems, v. 61, n. 12, p. 1726-1743, 2013. ISSN 0921-8890.

LIKHACHEV, M. et al. Anytime Dynamic A*: An anytime, replanning algorithm. International Conference on Automated Planning and Scheduling (ICAPS). Monterey, California, USA: 262-271 p. 2005.

MENEZES, V.; POZZER, C. Development of an Autonomous Vehicle Controller for Simulation Environments. Sociedade Brasileira de Computação (SBC) - Proceedings of Simpósio Brasileiro de Games e Entretenimento Digital (SBGames), 2018, Foz do Iguaçu, Paraná, Brazil.

NIELSEN, J. Usability engineering. San Diego, CA: Academic Press, 1994. ISBN 0080520294.

NILSSON, N. J. Artificial intelligence: a new synthesis. Morgan Kaufmann, 1998. ISBN 0080948340.

PELECHANO, N.; FUENTES, C. Hierarchical path-finding for Navigation Meshes (HNA∗). Computers & Graphics, v. 59, p. 68-78, 2016. ISSN 0097-8493.

RABIN, S. Artificial intelligence: Agents, architecture, and techniques. In: (Ed.). Introduction to Game Development. Hingham, MA: Charles River Media, 2005.

REYNOLDS, C. W. Steering behaviors for autonomous characters. Game Developers Conference, 1999, San Francisco, CA. Miller Freeman Game Group. p.763-782.

SAMET, H. An overview of quadtrees, octrees, and related hierarchical data structures. Earnshaw R.A. (eds) Theoretical Foundations of Computer Graphics and CAD. NATO ASI Series (Series F: Computer and Systems Sciences), v. 40, p. 51-68, 1988.

SHAH, B. C.; GUPTA, S. K. Speeding Up A* Search on Visibility Graphs Defined Over Quadtrees to Enable Long Distance Path Planning for Unmanned Surface Vehicles. International Conference on Automated Planning and Scheduling (ICAPS), 2016, London, UK. p.527-535.

SHEN, Z.; ZHOU, S. Behavior representation and simulation for military operations on urbanized terrain. Simulation, v. 82, n. 9, p. 593-607, 2006. ISSN 0037-5497.

SIS-ASTROS. Simulation Project ASTROS 2020 - Project 3.07.0065/Agreement 813782/2014. Federal University of Santa Maria, Brazil: Education Ministry 2018.

SOUISSI, O. et al. Path planning: A 2013 survey. Proceedings of 2013 International Conference on Industrial Engineering and Systems Management (IESM), 2013, Rabat, Morocco. IEEE. p.1-8.

STENTZ, A. The focussed D* algorithm for real-time replanning. International Joint Conference on AI (IJCAI). Quebec, Canada 95: 1652-1659 p. 1995.

STURTEVANT, N. R. A. Choosing a search space representation. Game AI Pro: Collected Wisdom of Game AI Professionals, v. 1, p. 253-258, 2013.

TECHNOLOGIES, U. Software Unity®. unity3d.com 2018.

WANG, B. et al. BIM based virtual environment for fire emergency evacuation. The Scientific World Journal, v. 2014, p. 22 p.,2014. ISSN 2356-6140.