Motion Planning of Autonomous Agents Situated in Informed Virtual Geographic Environments

Mehdi Mekni
Department of Computer Science
Sherbrooke University
Sherbrooke, Canada
Email: mmekni@gmail.com

***Abstract*—Multi-Agent Geo-Simulation (MAGS) aims to simulate phenomena involving a large number of autonomous situated actors (implemented as software agents) evolving and interacting within a Virtual representation of the Geographic Environment (VGE). Motion planning is a critical issue since it corresponds to one of the most important activities of agents moving in a complex and large-scale VGE. There is also a need for an accurate representation of the environment in order to support efficient path planning computation as well as reactive navigation for the detection and avoidance of obstacles and other agents. In this paper, we propose a semantically informed and geometrically precise virtual geographic environment method which allows to use Geographic Information System (GIS) data to automatically build an informed graph structure called Informed Virtual Geographic Environmnt (IVGE). Furthermore, we propose a topologic abstraction algorithm which builds a Hierarchical Topologic Graph (HTG) describing the IVGE and a Hierarchical Path Planning (HPP) algorithm which uses this graph. In addition, we propose a graph-based neighborhood structure in order to support motion planning of autonomous agents taking into account the characteristics of the IVGE.**

***Keywords*-Informed Virtual Geographic Environment (IVGE); Hierarchical Path Planning (HPP); Navigation and Collision Avoidance;**

I. INTRODUCTION

During the last decade, the Multi-Agent Geo-Simulation (MAGS) approach has attracted a growing interest from researchers and practitioners to simulate phenomena in a variety of domains including traffic simulation, crowd simulation, urban dynamics, and changes of land use and cover, to name a few [3]. Such approaches are used to study phenomena (i.e., car traffic, mobile robots, sensor deployment, crowd behaviours, etc.) involving a large number of simulated actors (implemented as software agents) of various kinds evolving in, and interacting with, an explicit description of the geographic environment called Virtual Geographic Environment (VGE). Nevertheless, simulating such autonomous situated agents remains a particularly difficult issue, since it involves several different research domains: geographic environment modelling, spatial cognition and reasoning, situation-based behaviours, etc. The autonomy of an agent is defined by its capacity to perceive, act and decide about its actions without external governance [23]. One of the most fundamental capacities of a situated autonomous agent is its ability to navigate inside a VGE while taking into account both the agent's and the environment's characteristics. When examining situated agents in a VGE, whether for gaming or simulation purposes, one of the first questions that must be answered is how to represent the world in which agents navigate [25]. Since a geographic environment may be complex and large-scale, the creation of a VGE is difficult and needs large quantities of geometrical data describing the environment characteristics (terrain elevation, location of objects and agents, etc.) [20] as well as semantic information that qualifies space such as buildings, roads, parks, as illustrated in Figure 2. Hence, a situated autonomous agent should consider the semantic information that qualifies the geographic environment in which and with which it interacts. Current approaches usually consider the environment as a monolithic structure, which considerably limits the way that large-scale, real world geographic environments and agent's spatial reasoning capabilities are handled [19].

Path planning is a typical spatial reasoning capability for situated agents in VGE [22]. The problem of path planning in MAGS involving complex and large-scale VGEs has to be solved in real time, often under constraints of limited memory and CPU resources [6]. Classic path planners provide agents with obstacle-free paths between two positions located in the VGE. Such paths do not take into account the environment's characteristics (topologic and semantic) nor different agent categories and capabilities [5]. For example, classic planners assume that all agents are equally capable to reach most areas in a given map, and that any terrain portion which is not traversable by one agent is also not traversable by the other agents. Such assumptions limit the applicability of these planners to solve a very narrow set of problems: path planning of homogeneous agents in a homogeneous environment. A path planning algorithm should take into account the semantic information that qualifies the geographic environment in which agents evolve and with which they interact. Moreover, in navigation applications

(local path planning) which involve several moving agents that do not know their respective mobility plans, a scheme for detection and resolution of collision conflicts between agents becomes mandatory. In this project, our goal is to address the issue of navigation and path planning for agents having different capabilities evolving in complex and large-scale geographic environments.

In order to achieve such a goal, a geographic environment model should precisely represent geographic features. It should also integrate several semantic notions characterising these geographic features. Since we deal with large-scale geographic environments, it would be appreciable to have a VGE organised hierarchically in order to reduce the search space for path planning. Indeed, hierarchical search is recognised as being an effective approach to reduce the complexity of such a problem [10]. There is also a need for autonomous situated agents which are able to plan paths, to detect and avoid both *static* and *dynamic* obstacles located in the VGE. Static obstacles correspond to areas that are not navigable for agents such as walls, fences, trees, rivers, etc. Static obstacles also include obstructions resulting from terrain elevation. Dynamic obstacles correspond to other moving agents which are navigating in the VGE.

In this paper, we propose a novel approach to simulate motion planning of autonomous situated agents in virtual geographic environments. This approach is composed of four parts: 1) a geometrically precise and semantically enhanced virtual geographic environment called *Informed VGE* (IVGE); 2) a topologic abstraction algorithm used to diminish path planning complexity; 3) a *Hierarchical Path Planning* (HPP) algorithm to support motion planning of autonomous agents situated in large-scale geographic environments; and 4) a graph-based structure called *Neighborhood Graph* (NG) to address collisions detection and avoidance between moving agents in 3D virtual environments.

The remainder of this paper starts with a discussion of related works on geographic environment representation using data provided by Geographic Information Systems (GIS) and path planning and navigation in virtual environments. In Section III, we present our approach to automatically create an Informed VGE. Section IV outlines a method to enhance the IVGE description using a topologic abstraction that reduces the size of the topologic graph and enables building a hierarchical topologic graph; Section V presents how we leverage the hierarchical graph structure of the IVGE model in order to support situated reasoning algorithms such as hierarchical path planning. Section VI introduces our model to support navigation in Informed VGE. Finally, we conclude with a discussion and present future works.

II. Related Works

In this section we provide a brief overview of prior works related to *environment representation*, and *path planning and navigation* in virtual environments.

A. Environment Representation

Virtual environments and spatial representations have been used in several application domains. For example, Thalmann *et al.* proposed a virtual scene for virtual humans representing a part of a city for graphic animation purposes [8]. Donikian *et al.* proposed a modelling system which is able to produce a multi-level data-base of virtual urban environments devoted to driving simulations [15]. Ali *et al.* used a multi-agent geo-simulation approach to simulate customers' behaviours in shopping malls [1]. More recently, Shao *et al.* proposed a virtual environment representing New York City's Pennsylvania Train Station populated by autonomous virtual pedestrians in order to simulate the movement of people [20]. Paris *et al.* also proposed a virtual environment representing a train station populated by autonomous virtual passengers, in order to characterise the levels of services inside exchange areas [17]. However, since the focus of these approaches is computer animation and virtual reality, the virtual environment usually plays the role of a simple background scene in which agents mainly deal with geometric characteristics. Indeed, the description of the virtual environment is often limited to the geometric level, though it should also contain topological and semantic information for other types of applications. Therefore, most interactions between agents and the environment are usually simple, only permitting to plan a path in a 2D or 3D world with respect to free space and obstacle regions [7].

B. Path Planning and Navigation

An extensive literature exists on agents' path planning in robot motion planning and virtual environments [13]. Roughly, these methods can be categorised as: *path planning* (global) and *navigation* (local).

Path Planning: The path planning issue, which consists of finding an obstacle-free path between two distinct positions located in a VGE, has been extensively studied. The computational effort required to plan a path, using a search algorithm such as A* [16] or Dijkstra [14], increases with the size of the search space [5]. Consequently, path planning on large-scale geographic environments can result in serious performance bottlenecks. However, representing the virtual environment using a hierarchical approach reduces the size of the search space as well as the complexity of path planning algorithms [10]. Two recent hierarchical triangulation-based path planning approaches are described in [6], namely *Triangulation A** and *Triangulation Reduction A**, which

are relevant to our work. *TA** makes use of the *Delaunay Triangulation* (DT) technique to build a polygonal representation of the environment without considering the semantic information. This results in an undirected graph connected by constrained and unconstrained edges, the former being traversable and the latter not. *TRA** is an extension of *TA** and abstracts the triangle mesh into a structure resembling a roadmap. Both TA* and TRA* are able to accurately answer path queries for agents since they make use of the DT technique. However, the abstraction technique used by TA* and TRA* aims at maximising triangle size, which does not reduce the size of the search space. Moreover, both *TA** and *TRA** assume a homogeneous flat environment, which considerably reduces the capacity to handle 3D environments enhanced with semantic information.

Navigation: An agent navigation behavior aims at predicting local collisions and avoiding the other navigating agents. Most current models are based on a particle approach proposed by Helbing [11]. However, this model suffers from several shortcomings. First, it cannot predict collisions since it waits for navigating agents to collide before adapting their behavior. In addition, it produces oscillations when adapting directions, which affects the quality of the agents' navigation behavior. Finally, Helbing's model manages very basic agents (particles) and it is difficult to adapt it to more complex simulated actors. Other reactive navigation models exist, including variants of potential fields [9]. They can handle dynamic environments, but suffer from "'local-minima"' problems and may not be able to find a collision-free path, when one exists [13]. Often, these models do not give any kind of guarantee on their behavior. Other navigation algorithms are based on path or roadmap modification, which allows a path to be deformed as a result of obstacle detection. These methods include Elastic Bands [18], Elastic Roadmaps [24], and adaptive roadmaps [21]. Alternatively, Lamarche and Donikian proposed to use a *Neighborhood Graph* (NG) based on a *Delaunay Triangulation* (DT) of the agents' positions filtered by visibility [12]. This structure offers a low computational cost, which enables the simulation of a large number of situated agents [12]. However, this NG does not take into account the terrain's elevation since it is based on a two-dimensional DT. In order to support moving agents in virtual environments, we claim that an NG should take into account terrain elevation. Indeed, a real environment is rarely flat and ignoring this information would distort the neighboring relationship between moving agents located in the virtual environment. In Section VI, we propose an approach which extends Lamarche and Donikian's method and enables us to create a 3D NG to support motion planning of autonomous agents situated in *Informed VGE*.

III. Computation of IVGE Data

In this section, we present our automated approach to computing the IVGE data directly from vector GIS data. Figure 1) depicts the four stages which compose our approach: *input data selection*, *spatial decomposition*, *maps unification*, and finally the *informed graph generation*.

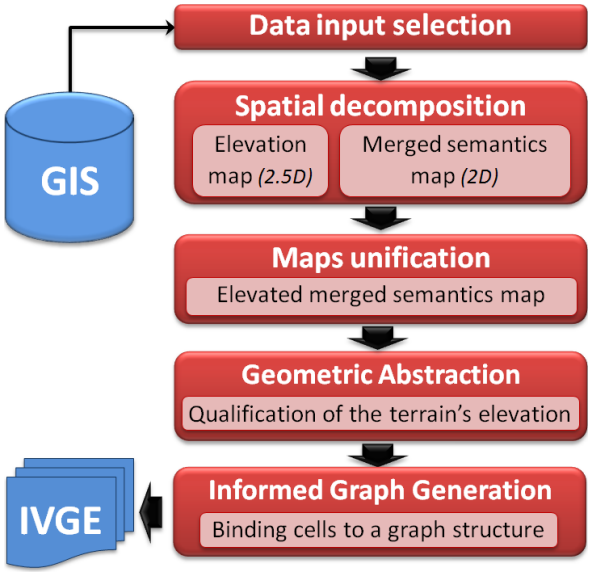
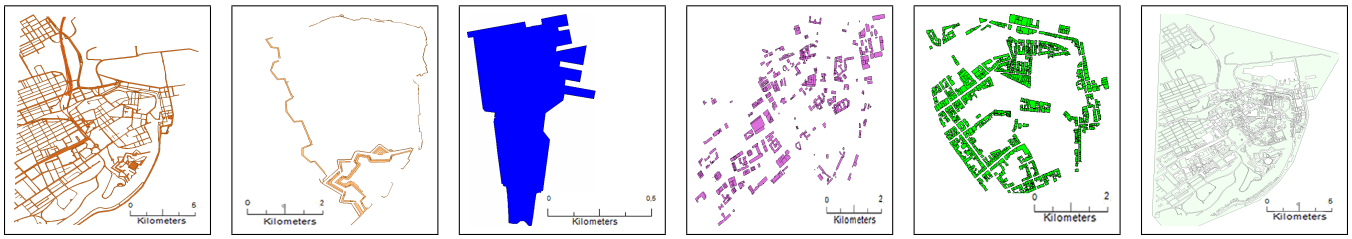


Figure 1: The five stages to obtain an IVGE from GIS data.

Input data selection: The first step of our approach is the only one requiring human intervention. It consists in selecting the different vector data sets which are used to build the IVGE. The input data can be organised into two categories. First, *elevation layers* containing the geographical marks that indicate absolute terrain elevations. Second, *semantic layers* are used to qualify various types of data in space. Each layer indicates the physical or virtual limits of a given set of features with identical semantics in the geographic environment, such as roads and buildings. The limits can overlap between two layers, and our model can merge the information.

Spatial decomposition: The second step consists of obtaining an exact spatial decomposition of the input data into cells. This process is entirely automatic, using a Delaunay Triangulation and can be divided into two parts in relation to the previous phase. First, an elevation map is computed and corresponds to the triangulation of the elevation layer. All the elevation points are injected into a 2D triangulation, the elevation being considered as an additional attribute. This process produces an environment subdivision composed of connected triangles (Figure 3(a)). Such a subdivision provides information about coplanar areas: the elevation of any point inside a triangle can be deduced thanks to the elevation



(a) (b) (c) (d) (e) (f)

Figure 2: Various semantic layers related to Quebec city in Canada: (a) road network; (b) old city wall; (c) marina; (d) governmental buildings; (e) houses; (f) sidewalk areas.

of the three original points. Second, a merged semantics map is computed, corresponding to a constrained triangulation of the semantic layers. Indeed, each segment of a semantic layer is injected as a constraint which keeps track of the original semantic data by adding additional attributes. The obtained map is then a constrained triangulation merging all input semantics (Figure 3(b)): each constraint represents as many semantics as the number of input layers containing it.

Maps unification: The third step to obtain our IVGE data consists of unifying the two maps previously obtained. This phase can be depicted as the mapping of the $2D$ merged semantic map (Figure 3(b)) onto the $2.5D$ elevation map (Figure 3(a)) in order to obtain the final $2.5D$ elevated merged semantics map (Figure 3(c)). First, preprocessing is carried out on the merged semantics map in order to preserve the elevation precision inside the unified map. Indeed, all the points of the elevation map are injected in the merged semantics triangulation, creating new triangles. Then, a second process elevates the merged semantics map. The elevation of each merged semantics point P is computed by retrieving the corresponding triangle T inside the elevation map, i.e. the triangle whose $2D$ projection contains the coordinates of P. Once T is obtained, the elevation is simply computed by projecting P on the plane defined by T using the Z axis.

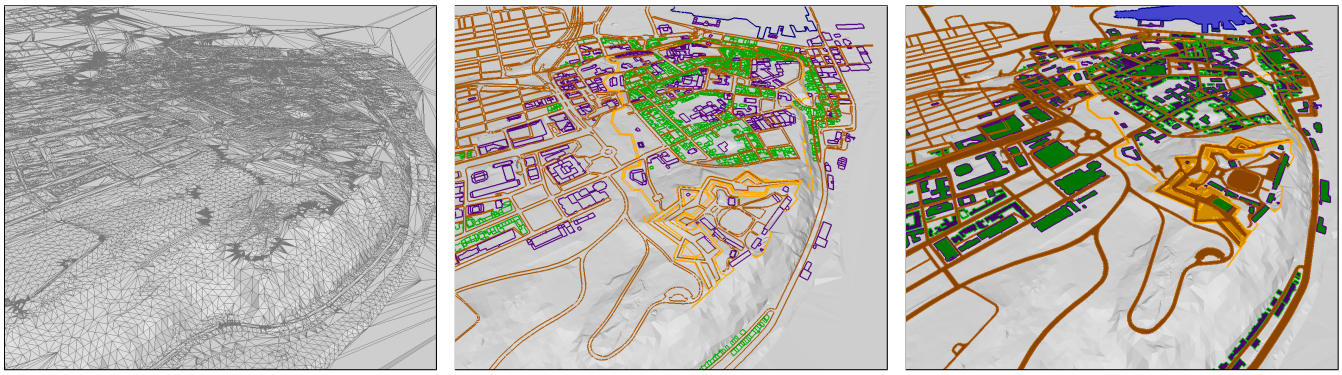
Informed graph generation: The resulting unified map now contains all the semantic information of the input layers, along with the elevation information. This map can be used as a topological graph in which each node corresponds to the map's triangles and each arc to the adjacency relations between these triangles. Then, common graph algorithms can be applied to this topological graph, especially graph traversal ones. One of these algorithms retrieves the node, and so the triangle, corresponding to given $2D$ coordinates. Once this node is obtained, it is possible to extract the data corresponding to the position, such as the elevation, and the semantic information. Many other algorithms can be applied, such as path planning and graph abstraction, but they are out of the scope of this paper and will not be detailed here.

IV. Topologic Abstraction

When dealing with large-scale and complex geographic environments the informed graph becomes very large. The size of a topologic graph has a direct impact on the computation time of the agent's spatial reasoning processes. In order to optimise such a computation time, we need to reduce the size of the informed graph representing the IVGE. The aim of the topologic abstraction is to provide a compact representation of the informed graph suitable for situated reasoning of situated agents. To this end, the topologic abstraction process extends the informed graph with new layers. In each layer (except for the initial layer which is called level 0), a node corresponds to a group of nodes of the immediate lower level. The topologic abstraction simplifies the IVGE description by combining cells (triangles) in order to obtain convex groups of cells. Such a hierarchical structure evolves the concept of *Hierarchical Topologic Graph* (HTG) in which cells are fused in groups and edges are abstracted in boundaries. To do so, convex hulls are computed for every node of the informed graph. Then, the coverage ratio of the convex hull is evaluated as the surface of the hull divided by the actual surface of the node. The topologic abstraction finally performs groupings of a set of connected nodes if and only if the group ratio is close to one. Let G be a group of cells, *Convex* be the convexity rate, and $CH(G)$ be the convex hull of the polygon corresponding to G. *Convex* is computed as follows:

$$Convex(G) = \frac{Surface(G)}{Surface(CH(G))} \quad and \quad 0 < Convex(G) \leq 1 \tag{1}$$

Each node c of the informed graph can be topologically qualified according to the number of connected edges given by the $arity(c)$ function: if $arity(c) = 0$ then c is a *closed* cell; if $arity(c) = 1$ then c is a *dead end* cell; if $arity(c) = 2$ then c is a *corridor* cell; and if $arity(c) > 2$ then c is a *crossroads* cell. The topologic abstraction algorithm is based on an in-depth exploration of the informed graph structure. At each step, the algorithm processes cells based on their topology in order to achieve a specific goal: 1) *Virtual Cells*: to characterise the outside of the IVGE; 2)



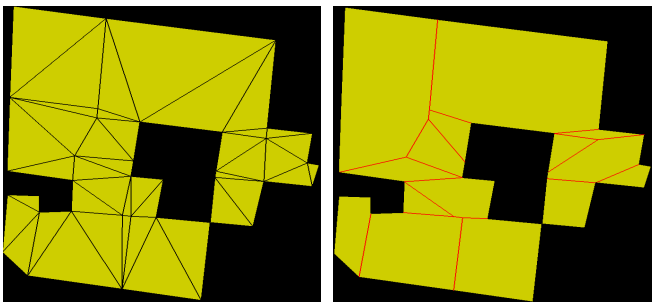
(a) Triangulated elevation map (2.5*D*). (b) Merged semantics map (2*D*). (c) Unified map (2.5*D*).

Figure 3: The two processed maps (*a*, *b*) and the unified map (*c*). The semantic colours are the same as in figure 2.

Access Cells: to identify access points corresponding to cells connected to at least one virtual cell (Figure); 3) *Corridor Cells*: to filter excessive discretization of space subdivision in narrow open areas (Figure); 4) *Crossroads*: to filter excessive discretization of space subdivision due to the misalignment of edges in open areas (Figure). 5) *Dead End*: Termination of the algorithm.

Let us detail the execution of the topologic abstraction algorithm which starts by processing the *virtual cells* and then their neighboring ones.

- Step 1 (processing of virtual cells): Bring together all the virtual cells in a virtual group, then merge into this group all the adjacent *dead end* cells. Proceed to Step 2.
- Step 2 (processing of access cells): Bring together all the access cells in a single group, then merge into this group all the *access end* cells. Proceed to Step 3, 4, or 5 depending on the type of the neighbooring cell *Cn*.
- Step 3 (processing of corridor cells): If the current cell *Cc* and its neighboor cell *Cn* are of type *corridor*, and if the current group is of type *crossroad*, proceed to Step 3-1, else proceed to Step 3-2.
  - Step 3-1: If $Convex(Gc \cup Cc) > Convex(Cc \cup Cn)$, then merge *Cc* into *Gc* and continue with *Cn*.
  - Step 3-2: Build a new group *Gp* of type *corridor* and assign *Cc* and *Cn* to it.
    - If $Gp = Gn$, where *Gn* corresponds to the group to which belongs *Cn*, then merge *Gc* with *Gp* and *Gn*
    - Else if *Gc* and *Gs* are of type *corridor* and if $Convex(Gc \cup Gp \cup Gn) > \max(Convex(Gc); Convex(Gp); Convex(Gn))$, then merge together *Gc*, *Gp*, and *Gn*.
    - Else if *Gc* and *Gn* are of type *crossroads* and If $Convex(Gp \cup Gn) > \max(Convex(Gp); Convex(Gn))$ then merge together *Gp* and *Gn*.
- Step 4 (processing crossroads cells): Build a group of type *crossroads Gc* and assign the current cell *Cc* to it as well as its neighbooring cells of type *crossroads* or *dead end*. Proceed to Step 3 for cells of type *corridor*.
  - If *Gc* has only one neighboor, turn it into a *dead end* group.
  - Else, if *Gc* has exactly two neighbours, transform it into *corridor* and apply the tests of Step 3;
  - Else, if $Convex(Gc \cup Gn) > \max(Convex(Gc); Convex(Gn))$ for any *Gs* in *dead end* neighboring groups, then merge *Gn* and *Gc*. Repeat the test of Step 4.
- Step 5 (processing dead end cells): Build a new group of type *dead end* and assign the current cell to it. This group will further be merged with its neighbour of type *corridor* or *crossroads* because of the tests in Steps 3 and 4.



(a) (b)

Figure 4: Illustration of the topologic abstraction process with a strict convex property (*C(gr) = 1*); (a) the exact space decomposition using CDT techniques (63 triangular cells) ; (b) the topologic abstraction (28 convex polygons)

Results: As an illustration, our IVGE generation model has been applied to an urban area representing the center part of Quebec City, with one elevation map and five semantic

layers. The creation of the IVGE takes less than five seconds on a typical computer (Intel Core 2 Duo processor 2.13Ghz, 1Go RAM). The resulting unified map approximately contains 122,000 triangles covering an area of $30km^2$. The necessary time to retrieve the triangle corresponding to a given coordinate is negligible (less than 10^{-4} seconds). We applied the topologic abstraction algorithm in order to build a three-level hierarchical and topologic graph. Level 0 corresponds to the informed graph resulting from the exact spatial subdivision. Level 1 of the topologic graph resulting from the topologic abstraction (with soft convex constraint, i.e. $Convex(c) = 1$) reduces the total number of cells (122,000) by merging them into 73,000 convex polygons (called groups) in 2.8 seconds. Level 2 of the topologic graph resulting from the topologic abstraction (with relaxed convex constraint, i.e. $Convex(c) = 0.9$) reduces the total number of groups by merging them into 12,000 convex polygons (called zones) in 1.9 seconds.

V. Hierarchical Path Planning

In this section, we present our hierarchical path planning algorithm (HPP for short). We then provide a computation analysis of the algorithm complexity which aims to point out the contribution of our algorithm. Finally, we propose a path enhancement method in order to optimise the computed paths for more realistic moving agents.

A. Algorithm

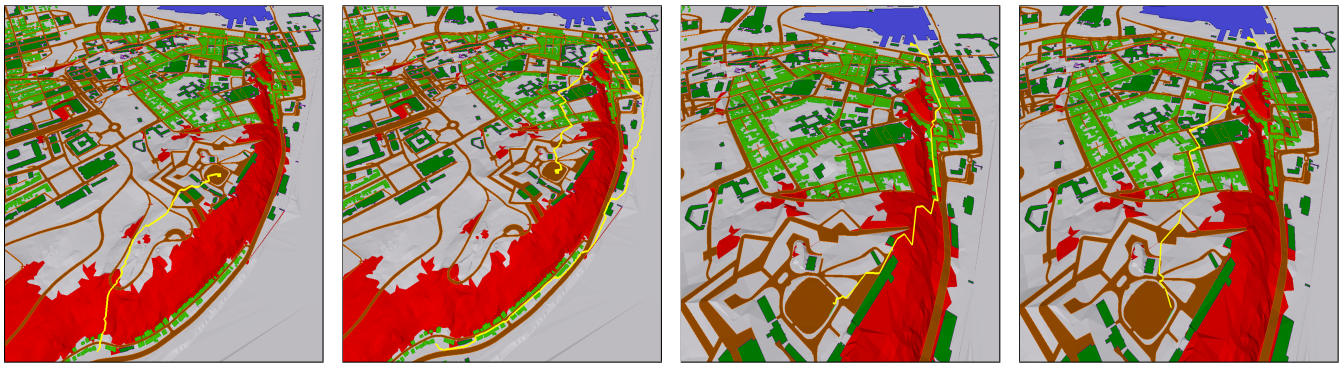
Let us consider the topologic graph extracted from the exact spatial decomposition before highlighting the usefulness of the topologic and semantic abstractions. Since cells are convex, it is possible to build an obstacle-free path by linearly connecting positions located at two different borders belonging to a given cell. Thus, it is also possible to use borders, represented by edges in the graph, to compute obstacle-free paths between different locations in the environment. Since the topologic graph structure is hierarchical, each node at a given level i (except at level 0) represents a group of convex cells or abstract cells of a lower level $i-1$. Hence, our approach can be used to compute a path linking two abstract nodes at any level.

Let us consider a hierarchical topologic graph G composed of i levels. Nodes belonging to level 0 are called *leaves* and represent convex cells produced by the exact spatial decomposition. Nodes belonging to higher levels ($i > 0$) are called *abstract nodes* and are composed of groups. Given a starting position, a final destination, and a hierarchical topologic graph G composed of i levels, the objective of our algorithm is to plan a path from the current position to the destination using G. The algorithm starts from the highest level of the hierarchy and proceeds as follows:

- **Step 1:** Identify the abstract nodes to which the starting position and the final destination belong.
  Two cases need to be considered:
  - Case 1: Both are in the same abstract node k at level i.
    Proceed to *step 1* with the groups (at level $i-1$) belonging to node k.
  - Case 2: They are in different abstract nodes k and j at level i. Proceed to *step 2*.
- **Step 2:** Compute the path from the abstract node k to the abstract node j.
  For each pair of consecutive nodes (s, t) belonging to this path, two cases are possible :
  - Case 1: Both are leaves. Proceed to *step 4*.
  - Case 2: Both are abstract nodes. Proceed to *step 3*.
- **Step 3:**
  - If the starting position belongs to s then identify to which group gs of s it belongs and proceed to *step 2*, in order to compute the path from the abstract node gs to the closest common boundary with the abstract node t. Else proceed to *step 2* in order to compute the path from the center of the abstract node s to the closest common boundary with the abstract node t.
  - If the final destination position belongs to t then identify to which group gd of t it belongs and proceed to *step 2*, in order to compute the path from the closest common boundary with the abstract node s to gd. Else proceed to *step 2* in order to compute the path from the closest common boundary with the abstract node s to the centre of the abstract node t.
- **Step 4:** Once in a leaf, apply a path planner algorithm (we used the Djkstra and A* algorithms) from the starting position to the final goal using the convex cells which belong to the informed graph.

The strategy adopted in this algorithm is to refine the path planning when getting closer to the destination. The algorithm starts by planning a global path between the start and the destination abstract nodes (step 1). Then, for each pair of successive abstract nodes, it recursively plans paths between groups (of lower levels) until reaching leaves (steps 2 and 3). Once at leaves (convex cells at level 0), the algorithm proceeds by applying a path planning algorithm such as Dijkstra and A* (step 4). Hence, at level i, the path planner exploration is constrained by the nodes belonging to the path computed at level $i+1$.

Moving agents can use this algorithm in order to plan paths within the IVGE. The path computed in step 2 is actually a coarse-grained path whose direction is only indicative. Since the path is refined in a *depth-first* way, agents



(a) (b) (c) (d)

Figure 5: HPP in the IVGE (the computed path is coloured in yellow). (a) path computed with no regard for the terrain shape; (b) path computed with regard for the terrain shape; (c) and (d) Search paths to get to a place (marina) in the IVGE (place described by semantics) without and with regard to terrain respectively.

can perform local and accurate navigation inside an abstract node without requiring a complete and fine-grained path computation towards the final destination. The lower levels' sub-paths (related to other abstract nodes) are computed only when needed, as the agent moves. Such a *just in time* path planning approach is particularly relevant when dealing with dynamic environments. Classic path planning approaches use the entire set of cells representing the environment and compute the complete path between a start and a final positions. These classical approaches suffer from two major drawbacks : 1) the computation time of a path is considerable since it involves all the cells composing the environment; 2) the planned path may become invalid as a consequence of changes in the environment. An interesting property of our hierarchical path planning approach is the optimization of calculation costs over time. Indeed, the entire path is only computed for the most abstracted graph, which contains a small number of abstract nodes compared to the informed graph (convex cells at level 0). In addition, our approach provides a *just in time* path planning which can accommodate a dynamic environment. Furthermore, this hierarchical path planning is adapted to any type of agents, whenever we are able to generate the abstracted graphs taking into account both the geographic environment and the agents' characteristics.

B. Complexity Analysis

In order to highlight the outcomes of our approach, let us compare the computation cost of our hierarchical path planning with the standard path planning. Let $G_0(V_0, E_0)$ be the graph representing the virtual environment at level 0, which corresponds to cells produced by the spatial decomposition process. Let V_0 correspond to the set of vertices and E_0 correspond to the set of edges at level 0. Let $|V_0| = N$ be the number of nodes of the graph G_0. Let us consider a starting position s and a destination position d located in the virtual environment. The computation cost of the shortest path between s and d at level 0 (represented by the graph G_0) is denoted by $C_0(N)$ and is given by the following equation:

$$C_0(N) = O(N * ln(N)) \tag{2}$$

Let us now compare $C_0(N)$ with the computation cost of our hierarchical path planning algorithm which relies on the hierarchical topologic graph with k levels. To this end, we need to raise some assumptions for the sake of simplification. First, let us assume that the topologic abstraction process may be thought of as a function h which abstracts a topologic graph G_{i-1} and builds a new topologic graph G_i. The function h can be written as follows:

$$h(G_{i-1}(V_{i-1}, E_{i-1})) = G_i(V_i, E_i) \quad with \quad 0 \leq i \leq k-1 \tag{3}$$

Let l_i be the *abstraction rate* between two successive levels $i-1$ and i (with $0 \leq i \leq k-1$). Since the abstraction process aims at reducing the number of nodes at each new level, we have $l_i > 1+\varepsilon$ (with $0 \leq i \leq k-1$) as illustrated in equation 4.

$$l_i = \frac{|V_{i-1}|}{|V_i|} \quad with \quad l_i > 1+\varepsilon \; and \; \varepsilon > 0 \tag{4}$$

Second, let us suppose that the k^{th} level of our hierarchical topologic graph is composed of m nodes. N which corresponds to the number of nodes of the graph G_0 can be expressed using equations 3 and 4 as follows:

$$N = m * l_{k-1} * ... * l_0 \tag{5}$$

$$N \geq m * (1+\varepsilon)^k \quad with \quad k > 0 \; and \; \varepsilon > 0 \tag{6}$$

$$N = m * \prod_{i=0}^{k-1} (l_i) \quad with \quad k > 0 \; and \; m > 0 \tag{7}$$

Let l_{Avg} be the average value of l_i (with $0 \leq i \leq k-1$). Using l_{Avg}, equation 7 becomes:

$$N = m * l_{avg}^k \quad with \quad k > 0 \; and \quad m > 0 \tag{8}$$

Let us replace the term N in equation 2 by its value in equation 8:

$$C_0(m) = O(m * l_{avg}^k * ln(m * l_{avg}^k)) \tag{9}$$

Equation 9 can be developed as follows:

$$C_0(m) = O(m * ln(m) * l_{avg}^k + m * l_{avg}^k * ln(l_{avg}^k)) \tag{10}$$

Let Nb_k be the number of nodes composing the computed path at level k. The computation cost of Nb_k is given by the following equation:

$$Nb_k = O(m * ln(m)) \quad with \quad k > 0 \; and \quad m > 0 \tag{11}$$

The hierarchical path planning algorithm involves the computation of the shortest path at level k and the refinement of the path linking each pair of successive nodes at lower levels. Therefore, the shortest path from s to d corresponds to the computation of Nb_k at level k and its refinement through the lowest levels. Such a shortest path is denoted C_k and has a computation cost which can be computed by the following equations:

$$C_k(m) = Nb_k * \sum_{j=0}^{k-1} l_{avg}^j \tag{12}$$

$$C_k(m) = Nb_k * \frac{l_{avg}^k - 1}{l_{avg} - 1} \tag{13}$$

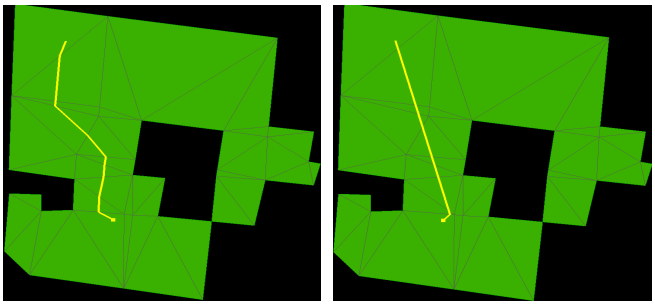
The term Nb_k in equation 13 is replaced by its value expressed in the equation 11 as follows:

$$C_k(m) = O(m * ln(m) + m * \frac{l_{avg}^k - 1}{l_{avg} - 1}) \tag{14}$$

Let us compare the computation costs of standard path planning approaches (equation 10) and our hierarchical path planning approach (equation 14). First, it is obvious that the first term $m * ln(m)$ in equation 10 is inferior to the first term $m * ln(m) * l_{avg}^k$ in equation 14 since the abstraction rate $l_{avg}^k > 1$. Second, in a similar way, the second term $m * (l_{avg}^k - 1/l_{avg} - 1)$ in equation 10 is inferior to the second term $m * l_{avg}^k * ln(l_{avg}^k)$ in equation 14. In conclusion, the hierarchical path planning algorithm along with the hierarchical topologic graph that we propose is at least $ln(l_{avg}^k)$ orders of magnitude faster than standard path planning approaches.

C. Path Optimisation

The topological abstraction only groups together adjacent cells or groups of cells with respect to the convexity criterion. While this approach is efficient to reduce the size of the topologic graph, it gives up the optimality of the computed path. Indeed, paths are optimal in the abstract graph but not necessarily in the initial problem graph (informed graph at level 0). In order to improve the quality of the computed path (i.e., length and visual optimisation), we perform a post-processing phase called *path optinmisation* (*Figure 6*). Our strategy for path optimisation is simple, but produces good results. The main idea is to replace local sub-optimal parts of the computed paths by straight lines. We start from one end of the path (*Figure 6(a)*) and for each node part of the computed path, we check whether we can reach a subsequent node in the path in a straight line. If this is possible, then the linear path between the two nodes replaces the initial sub-optimal sequence between these nodes (*Figure 6(b)*).



(a) (b)

Figure 6: (a) The original computed path ; (b) The computed path after optimisation.

Results: The HTG resulting from the topologic abstraction process is particularly suitable to support HPP in IVGE. Two types of HPP have been implemented: 1) a path linking two positions located in the IVGE using the A* algorithm (Figures 5(a) and 5(b)); and 2) a search path linking a position to a qualified area within the IVGE using the *Dijkstra* algorithm (Figures 5(c) and 5 (d)). *Figure 5*(a) shows a path planning which avoids obstacles such as *buildings*, *walls*, but does not take into account the terrain characteristics. Therefore, this path crosses areas coloured in red which represent steep slopes. However, *Figure 5*(b) respects both the terrain and the obstacles in the IVGE. To illustrate path planning towards a target area qualified by one or several semantics, *Figure 5*(c) shows the computed path to reach the marina (the marina is coloured in blue at the top of the figure). This path avoids obstacles such as *buildings*, *walls*, but does not take into account the terrain characteristics. *Figure 5*(d) avoids steep slopes (coloured in red) as well as obstacles situated in the IVGE and reaches a place identified by the semantic information (marina). Finally, in order to highlight the outcomes of the path optimisation process, we randomly selected 19 starting and destination positions in the IVGE. For each pair of positions, we compared the original computed path length with the optimised path length. *Figure 7* depicts the comparison of the non optimised computed path length and the optimised

path length. It shows how the optimisation process reduces the computed path length by an average of 16%.

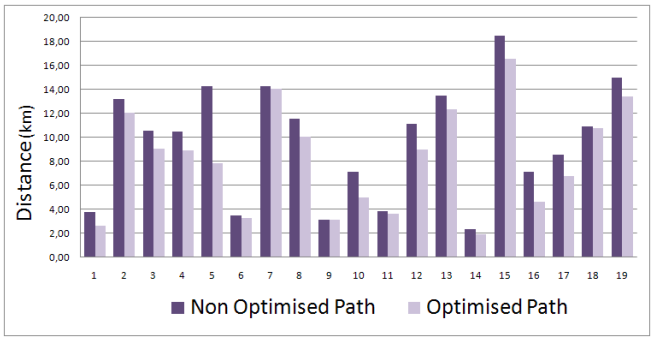


Figure 7: Optimised versus non-optimised paths lengths.

VI. Navigation

The geometrically precise spatial subdivision along with the topologic abstraction of the virtual geographic environment are not sufficient to handle the navigation of several moving agents populating the same IVGE. A structure and a mechanism allowing for dynamic collisions detection and avoidance is necessary to achieve consistent motion planning of moving agents in IVGE. In this section, we first introduce the concept of neighborhood graph (NG) for the support of agent navigation. Next, we detail the algorithm that we propose to build an NG while taking into account obstacles such as walls and fences as well as terrain elevation.

Neighborhood Graph: A *neighborhood graph* (NG) consists of a data structure reflecting the relative positions of moving agents while taking into account obstacles located in the IVGE. Two entities are considered neighbors if they are not separated by any obstacle. Since the NG is based on moving agents' positions, it should be updated as rapidly as the movement of agents. Therefore, its computation complexity must be optimised. Moreover, if we make no assumption about the perception distance and the angle of the agent's field of view, the complexity of building the NG should only depend on the number of agents rather than on their relative distances. Based on these assumptions, we define our NG using a *3D Delaunay Triangulation* (3D-DT) of moving agents (*dynamic information*) filtered using both the description of static obstacles situated in the IVGE such as walls and obstructions resulting for the terrain's elevation (*static information*). Figure 8 shows an example of construction of an NG considering obstacles and obstructions within the IVGE. In the following sud-section, we propose an algorithm to build NGs and we analyse its complexity.

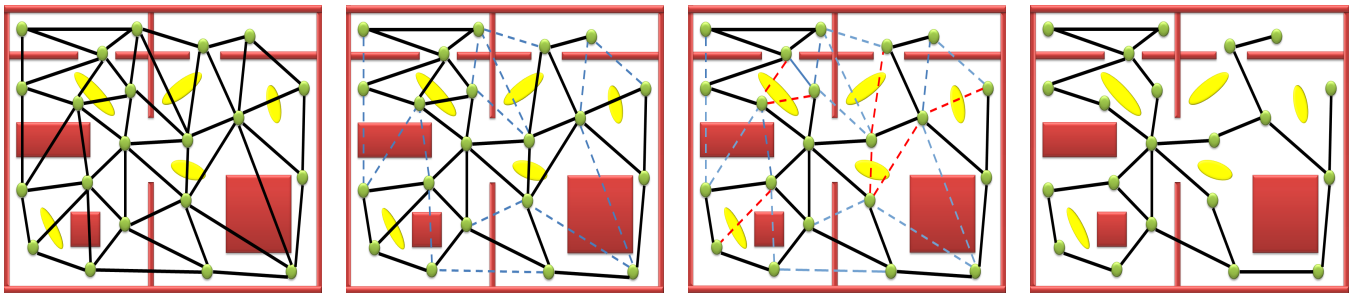
Algorithm: In this section, we present a step-by-step description of our algorithm to build NGs. In a first step, the spatial positions of the moving agents are collected and constitute the set of points to triangulate. Let n points be given by their Cartesian coordinates $p_1(x_1, y_1, z_1)$, $p_2(x_2, y_2, z_2)$, ..., $p_n(x_n, y_n, z_n)$. The algorithm is based on three steps. Step 1: Compute the 3D-DT; Step 2: For each edge $E_{i,j}$ of the DT linking a pair of points (P_i, P_j), verify if this edge crosses an area defined as an obstacle in the IVGE. If yes, remove $E_{i,j}$. Step 3: For each edge $E_{i,j}$ of the DT linking a pair of points (P_i, P_j), verify if P_i and P_j are obstructed as a result of the terrain's elevation. If yes, remove $E_{i,j}$.

The complexity of construction of the 3D-DT for n moving agents is of the order of $O(n \ln n)$, making it usable for a large number of moving agents (Figure 8(a)). In the second step, for each edge $E_{i,j}$ of the 3D-DT, a verification is computed to ensure the visibility (free of environment obstacles) between the moving agents (Figure 8(b)). In the third step, for each edge $E_{i,j}$ of the 3D-DT, a verification is computed to ensure the visibility (free of environment obstructions resulting from terrain elevation) between the moving agents (Figure 8(c)). If $E_{i,j}$ is not free of obstacles and obstructions, the edge is deleted from the 3D-DT (Figure 8(d)). Let us now analyse the complexity of our algorithm. The 3D-DT (Step 1) can be computed in $O(n\log n)$ running time [2]. In Step 2, for each edge $E_{i,j}$ considered, $O(n)$ verifications are computed to ensure that $E_{i,j}$ is free of environment obstacles. These verifications have a linear complexity which only depends on the number of vertices (corresponding to the moving agents). Step 2 runs in $O(n)$ time. In Step 3, for each edge $E_{i,j}$ considered, $O(n)$ verifications are computed to ensure that $E_{i,j}$ is free of obstructions due to the terrain elevation. Step 3 also runs in $O(n)$ time. Since Step 1 is $O(n\log n)$, the complexity of our NG algorithm is dominated by Step 1 and thus of $O(n\log n)$.

VII. Discussion

In order to reduce the search space, we proposed an HTG that groups convex cells and groups of cells. Hence, each abstract node at level i contains a subset of this graph at level $i-1$, composed by at least one node or abstract node. The extraction of this HTG only requires an acceptable one-time computation cost and a low memory overhead. Despite the reduction of the number of nodes, this technique raises two application-dependent issues that must be addressed: ***hierarchical traversal cost*** and ***information richness***.

First, the *hierarchical traversal cost* increases with each grouping, which might limit the performance of the search space reduction brought by the hierarchical representation. Indeed, despite the number of levels of the HTG, the path planning process provides moving agents with a set of convex cells (belonging to level 0) to pass through in order to reach the final destination. This means that the path planning process must inevitably traverse the HTG from top to bottom



(a) (b) (c) (d)

Figure 8: Generation of the neighborhood graph (NG): (a) initial 3D-DT using the moving agents positions (green circles); (b) filter of 3D-DT considering obstacles (red shapes) in the IVGE; (c) filter of 3D-DT considering terrain elevation (yellow shapes) in the IVGE.

in order to compute such a set of cells.

Second, the *information richness* decreases with each grouping level, which could lead to useless additional abstraction levels that may not improve the decision-making of the HPP algorithm. Indeed, the more potential sub-paths an abstract node contains, the less its choice influences the path planning process. Therefore, the determination of the number of topologic abstraction levels must be carefully analyzed with respect to these two critical issues in addition to the application requirements.

Another important aspect of our IVGE is its capability to represent geographic environments which are distributed in space. By using the HTG, our model is capable of representing portions of geographic environments which are not adjacent in space. For example, consider the problem of traveling by car from Quebec city (QC, Canada) to New York (NY, USA). We need to compute the shortest (minimum distance) path from a given address in Quebec city, such as 312 *Marie-Louise*, to a given address in New York city, let us say 1213 4*th* Avenue, *Brooklyn*. Given a detailed description of the geographic environment showing all roads annotated with driving distances, a classic planner can compute such a travel route. However, this might be an expensive computation, given the large size of the description of the geographic environment. This problem may be solved in a three-step process. First, we compute the path from 312 *Marie-Louise* to a major highway leading out of Quebec city. Second, we compute the path from Quebec to the boundaries of New York. Third, we compute the path from the incoming highway to 1213 4*th* Avenue, *Brooklyn*. Assuming that the second path is mostly composed of highways and can be quantified (distance and travel time), it is easy to model this path using a *conceptual node* in our hierarchical topologic graph. A conceptual node allows for linking spatially distributed geographic environments and hence allows us to accurately compute optimal paths with respect to these environment characteristics.

In contrast to Lamarche and Donikian's NG [12] which only takes into account static obstacles such as walls and fences (Figure 8(b)), our NG model also includes obstructions resulting from the terrain's elevation (Figure 8(c)). Lamarche and Donikian's NG is based on a 2D-DT implementing the algorithm proposed in [4] whose computation cost is of $O(n\log n)$ [12]. However, Attali *et al.* proposed an optimised algorithm to build a 3D-DT which also runs in $O(n\log n)$ [2]. Our NG extends Lamarche and Donikian's approach with respect to the algorithm's complexity ($O(n\log n)$). Our NG takes into account the terrain's elevation and uses Attali's optimised 3D-DT and thus runs in $O(n\log n)$. In addition to its good properties in terms of computation complexity, the DT also has good topological properties. Indeed, it ensures that each point is connected to its nearest neighbor. An NG inherits from this property by adding the concept of filtering visibility. We define the $k-direct$ neighborhood $V_k(E)$ of an agent E as all agents related to E by k arcs in the NG. This set contains the nearest visible neighbors to an agent within k hops from E, considering the NG. This property shows that for the collision detection purposes, only agents belonging to $V_1(E)$ (also called *immediate neighbors*) need to be tested. On the other hand, according to the method of construction, the $k-direct$ neighborhood adapts to the density of population. For example, in a dense environment, the $k-direct$ neighborhood contains a set of agents which are visible and close to the agent E, and in environment of low density, it contains a set of remote visible agents. The k parameter allows to specify the number of hops while accessing the agent neighbors by agent type.

VIII. Conclusion and Future Works

In this paper, we proposed an accurate and automated approach for the generation of semantically enhanced and geometrically precise virtual geographic environments using GIS data. This novel approach offers several advantages. First, the description of the IVGE is realistic since it is based

on standard GIS data and accurate because it is produced by an exact spatial decomposition technique which uses data in a vector format. Hence, this description preserves both the geometric and the topological characteristics of the geographic environment and enables a graph-based description of the virtual environment enhanced with semantics. The main outcome of such a semantically enhanced and geometrically precise virtual geographic environment concerns agents' situated reasoning capabilities such as path planning and navigation in large-scale and complex geographic environments. We proposed a hierarchical path planning algorithm (using *Dijkstra* and *A**) which takes advantage of our IVGE model to provide paths which take into account the agents' and environment's characteristics. We also proposed an algorithm to build neighborhood graphs, a graph-based structure used by moving agents to support navigation (collision detection and avoidance).

We are currently working on further improvements of the IVGE description by integrating enriched knowledge representations (called *the environment knowledge*) using *Conceptual Graphs* aimed at assisting situated agents' interactions with the IVGE and helping them achieve their goals. The goal of the environment knowledge integration is to extend the agents' knowledge about their surrounding environment. We are also working on the extension of the neighborhood graph concept to support agents' perception capabilities within the IVGE. The above-mentioned contributions of our model offer new opportunities for many applications in a variety of application domains including the entertainment industry (games and movies), security planning and crowd management (planning events involving large crowds), and environment monitoring in natural environments.

ACKNOWLEDGEMENT

Mehdi Mekni benefited from a PDF scholarship granted by FQRNT (***F**onds **Q**uébécois de la **R**echerche sur la **N**ature et les **T**echnologies*).

REFERENCES

[1] W. Ali and B. Moulin. 2D-3D multiagent geosimulation with knowledge-based agents of customers' shopping behavior in a shopping mall. In *Spatial Information Theory*, pages 445–458. Elsevier, 2005.

[2] D. Attali, J.-D. Boissonnat, and A. Lieutier. Complexity of the Delaunay triangulation of points on surfaces: the smooth case. In *SCG '03: Proceedings of the Nineteenth Annual Symposium on Computational Geometry*, pages 201–210, New York, NY, USA, 2003. ACM.

[3] I. Benenson and P. Torrens. *Geosimulation: Automata-Based Modeling of Urban Phenomena*. John Wiley and Sons Inc., 2004.

[4] J.-D. Boissonnat and M. Yvinec. *Algorithmic Geometry*. Cambridge University Press, New York, NY, USA, 1998.

[5] A. Botea, M. Müller, and J. Schaeffer. Near optimal hierarchical path-finding. *Journal of Game Development*, 1:7–28, 2004.

[6] D. Demyen and M. Buro. Efficient triangulation-based pathfinding. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference (AAAI'06)*, Boston, Massachusetts, USA, July 16-20 2006.

[7] S. Donikian and S. Paris. Towards embodied and situated virtual humans. In *Motion in Games*, pages 51–62, 2008.

[8] N. Farenc, R. Boulic, and D. Thalmann. An informed environment dedicated to the simulation of virtual humans in urban context. In P. Brunet and R. Scopigno, editors, *Computer Graphics Forum (Eurographics '99)*, volume 18(3), pages 309–318. The Eurographics Association and Blackwell Publishers, 1999.

[9] H. Haddad, M. Khatib, S. Lacroix, and R. Chatila. Reactive navigation in outdoor environments using potential fields. In *Proceedings of the International Conference on Robotics and Automation*, pages 1232–1237, Leuven, Belgium, May 1998. IEEE.

[10] D. Harabor and A. Botea. Hierarchical path planning for multi-size agents in heterogenous environments. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS'08)*, Sydney, Australia, September 14-18 2008.

[11] D. Helbing, I. Farkas, and T. Vicsek. Simulating dynamical features of escape panic. *Nature*, 407:487, 2000.

[12] F. Lamarche and S. Donikian. Crowds of virtual humans: a new approach for real time navigation in complex and structured environments. *Computer Graphics Forum, Eurographics'04*, 2004.

[13] S. LaValle. *Planning Algorithms*. Cambridge University Press., Cambridge, 2006.

[14] J. Lengyel, M. Reichert, B. R. Donald, and D. P. Greenberg. Real-time robot motion planning using rasterizing computer graphics hardware. In *SIGGRAPH '90: Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques*, pages 327–335, New York, NY, USA, 1990. ACM.

[15] J.-E. Marvie, J. Perret, and K. Bouatouch. Remote interactive walkthrough of city models. In *Proceedings of the 11th Pacific Conference on Computer Graphics and Applications (PG'03)*, pages 389–393, Oct. 2003.

[16] N. Nilsson. *Principles of Artificial Intelligence*. Springer-Verlag, Berlin ; Heidelberg ; New York, third edition, 1982.

[17] S. Paris. *Characterisation of the levels of services and modeling of the movement of people inside exchange areas*. PhD thesis, Université de Rennes 1, October 2007.

[18] S. Quinlan and O. Khatib. Elastic bands: Connecting path planning and control. In *ICRA (2)*, pages 802–807, 1993.

[19] S. Rodriguez, V. Hilaire, S. Galland, and A. Koukam. Holonic modeling of environments for situated multi-agent systems. In *Environments for Multi-Agent Systems II*, pages 18–31. 2006.

[20] W. Shao and D. Terzopoulos. Environmental modeling for autonomous virtual pedestrians. *Digital Human Modeling for Design and Engineering Symposium*, 2005.

[21] A. Sud, R. Gayle, E. Andersen, S. Guy, M. Lin, and D. Manocha. Real-time navigation of independent agents using adaptive roadmaps. In *SIGGRAPH '08: ACM SIGGRAPH 2008 classes*, pages 1–10, New York, NY, USA, 2008. ACM.

[22] G. Thomas and S. Donikian. Virtual humans animation in informed urban environments. *Computer Animation 2000*, pages 112–119, 2000.

[23] M. Wooldridge. *Introduction to Multiagent Systems*. John Wiley and Sons Inc., London, UK, 2001.

[24] Y. Yang and O. Brock. Elastic roadmaps: Globally task-consistent motion for autonomous mobile manipulation in dynamic environments. In *Robotics: Science and Systems*, 2006.

[25] J. Zhu, J. Gong, H. Lin, W. Li, J. Zhang, and X. Wu. Spatial analysis services in virtual geographic environment based on grid technologies. *MIPPR 2005: Geospatial Information, Data Mining, and Applications*, 6045(1):604–615, 2005.