MEHDI MEKNI

Automated Generation of Geometrically-Precise and Semantically-Informed Virtual Geographic Environments Populated with Spatially-Reasoning Agents

Thèse présentée
à la Faculté des études supérieures de l'Université Laval
dans le cadre du programme de doctorat en informatique
pour l'obtention du grade de Doctor ès sciences (Ph.D.)

FACULTÉ DES SCIENCES ET DE GÉNIE
UNIVERSITÉ LAVAL
QUÉBEC

2010

Résumé

La Géo-Simulation Multi-Agent (GSMA) est un paradigme de modélisation et de simulation de phénomènes dynamiques dans une variété de domaines d'applications tels que le domaine du transport, le domaine des télécommunications, le domaine environnemental, etc. La GSMA est utilisée pour étudier et analyser des phénomènes qui mettent en jeu un grand nombre d'acteurs simulés (implémentés par des agents) qui évoluent et interagissent avec une représentation explicite de l'espace qu'on appelle *Environnement Géographique Virtuel* (EGV). Afin de pouvoir interagir avec son environnement géographique qui peut être dynamique, complexe et étendu (à grande échelle), un agent doit d'abord disposer d'une représentation détaillée de ce dernier. Les EGV classiques se limitent généralement à une représentation géométrique du monde réel laissant de côté les informations topologiques et sémantiques qui le caractérisent. Ceci a pour conséquence d'une part de produire des simulations multi-agents non plausibles, et, d'autre part, de réduire les capacités de raisonnement spatial des agents situés. La planification de chemin est un exemple typique de raisonnement spatial dont un agent pourrait avoir besoin dans une GSMA. Les approches classiques de planification de chemin se limitent à calculer un chemin qui lie deux positions situées dans l'espace et qui soit sans obstacle. Ces approches ne prennent pas en compte les caractéristiques de l'environnement (topologiques et sémantiques), ni celles des agents (types et capacités). Les agents situés ne possèdent donc pas de moyens leur permettant d'acquérir les connaissances nécessaires sur l'environnement virtuel pour pouvoir prendre une décision spatiale informée.

Pour répondre à ces limites, nous proposons une nouvelle approche pour générer automatiquement des *Environnements Géographiques Virtuels Informés* (EGVI) en utilisant les données fournies par les *Systèmes d'Information Géographique* (SIG) enrichies par des informations sémantiques pour produire des GSMA précises et plus réalistes. De plus, nous présentons un algorithme de planification hiérarchique de chemin qui tire avantage de la description enrichie et optimisée de l'EGVI pour fournir aux agents un chemin qui tient compte à la fois des caractéristiques de leur environnement virtuel et de leurs types et capacités. Finalement, nous proposons une approche pour la gestion des connaissances sur l'environnement virtuel qui vise à supporter la prise de décision informée et le raisonnement spatial des agents situés.

Abstract

Multi-Agent Geo-Simulation (MAGS) is a modelling and simulation paradigm which has attracted a growing interest from researchers and practitioners for the study of various phenomena in a variety of domains such as traffic simulation, urban dynamics, environment monitoring, as well as changes of land use and cover, to name a few. These phenomena usually involve a large number of simulated actors (implemented as software agents) evolving in, and interacting with, an explicit spatial environment representation commonly called *Virtual Geographic Environment* (VGE). Since a geographic environment may be complex and large-scale, the creation of a VGE is difficult and needs large quantities of geometrical data originating from the environment characteristics (terrain elevation, location of objects and agents, etc.) as well as semantic information that qualifies space (building, road, park, etc.). Current MAGS approaches usually consider the environment as a monolithic structure, which considerably reduces the capacity to handle large-scale, real world geographic environments as well as agent's spatial reasoning capabilities. Moreover, 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. Available path planners provide agents with obstacle-free paths between two located positions in the VGE, but take into account neither the environment's characteristics (topologic and semantic) nor the agents' types and capabilities. In addition, agents evolving in a VGE lack for mechanisms and tools that allow them to acquire knowledge about their virtual environment in order to make informed decisions.

In this thesis, we propose a novel approach to automatically generate a semantically-enriched and geometrically-precise representation of the geographic environment that we call Informed Virtual Geographic Environment (IVGE). Our IVGE model efficiently organizes the geographic features, precisely captures the real world complexity, and reliably represents large-scale geographic environments. We also provide a new hierarchical path planning algorithm which leverages the enriched description of the IVGE in order to support agents' reasoning capabilities while optimising computation costs and taking into account both the virtual environment's characteristics and the agents' types and capabilities. Finally, we propose an environment knowledge management approach to support the agents' spatial decision making process while interacting with the IVGE.

Foreward

Completing this doctoral work has been a wonderful and often overwhelming experience. There are many persons who deserve all my gratitude and acknowledgment. First, all my gratitude and respect to the members of my thesis jury: Prof. Jules Desharnais, Prof. Christophe Claramunt, Prof. Nadir Belkhiter and Prof. Luc Lamontagne. I really appreciate all their comments and advise.

Thank you to Prof. Bernard Moulin, for giving me the opportunity to be part of your team and for being my advisor and putting up with me for four years. I started my Ph.D. knowing absolutely nothing about Multi-Agent Geo-Simulation and I like to think that now, thanks to you, I can manage a bit. I just hope that one day I will be as knowledgeable and as competent a scientist as you. This thesis owes a lot to your patient and intelligent guidance, your open-mindedness, and to the important research experience I had in your lab.

Thank you to Prof. Phil Graniero, my thesis co-advisor. You have so much to do all the time and yet still find the time to answer my emails and be there for me. We did not spend a lot of time together, but you always found the right words to say at the right time. This is a rare talent at which you excel. Thank you.

A special thank you goes to Prof. Nadir Belkhiter who opened his door to me. You keep saying I have all the merit, that you did not do anything but one thing is certain: without you I would not be here. You are the one who gave me the initial push to fly with my own wings, and for that I will be eternally grateful.

I would like to take this opportunity to express my gratitude to my friends for their support and constant encouragement. Many thanks to all my colleagues from the *Cognitive Informatics Laboratory* I have worked with. Thank you for your collaboration, keen advice, friendship and support during my Ph.D. program.

Finally, I want to recognize the valuable support from my parents, I thank them for everything I have achieved. To my sisters and my family that I missed a lot every Sunday.

Most of all, I thank my soul mate for her great care, unconditional support and continuous encouragement throughout this time, without which this dissertation would not have been possible.

To my mother and father, and to all my family.

"'You are fastened to them and cannot understand how, because they are not fastened to you."'
Antonio Porchia, Voces, 1943, translated from Spanish by W.S. Merwin

Contents

List of Tables

List of Figures

Chapter 1

General Introduction

Multi-Agent Geo-Simulation (MAGS) is a modelling paradigm [BT04] which attracts a growing interest from researchers and practitioners to simulate various phenomena in a variety of domains such as traffic simulation, crowd simulation, environment monitoring, and changes of land use and cover, to name a few. Such approaches are used to study various phenomena (i.e. car traffic, crowd behaviours, sensor web deployment, etc.) involving a large number of simulated actors (implemented as software agents) evolving in, and interacting with, an explicit spatial environment representation usually called *Virtual Geographic Environment* (VGE).

Creating a VGE populated by autonomous agents is a pretty demanding task which can be broken down into two main blocks: *creating the virtual environment itself*, and *creating the autonomous agents populating it*. On the one hand, so many challenges arise when creating a VGE: what it looks like, how it is represented, what are the objects and agents it contains, etc. Recreating real environments through a computer is a complex task which is still under constant study and development, further reducing the boundaries between virtuality and reality, as they go along.

On the other hand, creating autonomous agents for populating the environment is a task which is just as difficult and complex. The autonomous agents, even though they can be considered as objects within the environment, have a lot of additional properties with their own set of problems to solve: how to represent them, how to make them move, how to make them reason, plan, and act with respect to the VGE's characteristics, etc. What makes the task even more daunting is the fact that the inner workings of real humans and animals are extremely complex and still not fully understood. Approximations and theories are made, implementation solutions are presented and computer programs are created trying to imitate as best as possible the complexity and inner workings of real live creatures, be they instinct

driven animals, or conscious and reasoning humans.

Actually, creating realistic environments on one side, and plausible autonomous agents on the other is not enough. Another important problem still needs to be solved: how to make the agents interact with the VGE? How can they evolve within it, perceive it, plan their actions according to the VGE's characteristics? Agents should be able to interact with the VGE and with the objects it contains. Indeed, in order to create reliable MAGS, the agents immersed in the VGE should be able to perform all the tasks their real living counterparts can: perceive their surroundings, gather information and memorize it, detect and avoid obstacles as well as other agents as they move, and plan a path according to what they see and what they know, interact with objects surrounding them, interact with other agents, etc. This is yet another difficult task, with its own set of problems to solve, as complex and essential as the creation of the environment and the agents themselves.

Building precise and enriched VGE and enabling agents to interact with such environments is the problem addressed by this thesis.

1.1 Problems and Research Issues

Building MAGS using agents which can reason about space not only requires appropriate computation algorithms, but also an efficient description of the spatial environment. Such a description must represent the geometrical and topological information which corresponds to geographic features which are provided by *Geographic Information Systems* (GIS). Moreover, this representation should qualify space by associating semantics with geographic features in order to allow spatial reasoning.

Since a geographic environment may be complex and large-scale, the creation of a VGE is difficult and needs large quantities of geometrical data originating from the environment characteristics (terrain elevation, location of objects and agents, etc.) as well as semantic information that qualifies space (building, road, park, etc.). The VGE description should rely on an efficient structure which supports easy and optimized access and query techniques. The complexity of building such a description should only depend on the geometrical complexity of the geographic environment rather than on its scale.

A number of challenges arise when creating such an informed VGE, among which we mention: 1) automatically creating a precise geometric representation of a 3D VGE; 2) automatically integrating several types of semantic information in the geometric representation; and 3) making use of this representation in spatial reasoning algorithms such as navigation

and path planning which are required for MAGS.

To enable an autonomous agent to interact with its environment, we might think of storing the entire interaction process within the agent's knowledge model. Thus, the agent would be able to observe the world that surrounds it and to gather raw information from its sensors. After that, it would process this raw data through a complex reasoning module in order to try to derive high-level information and to determine the interaction possibilities offered by the objects it is observing. This approach is extremely complex, very difficult to implement, and is rarely applicable to complex interaction processes. The more complex the object is, the harder it is to derive abstract information and the more complex the reasoning algorithm needs to be. This process can become extremely costly in terms of calculation time and resources when the complexity of the environment and the objects contained in it increases.

Another approach takes advantage of the fact that the agents are evolving in a virtual world which fully stores the entire interaction information. Such a method can ease the load on the agents since it removes the need to determine abstract data by directly providing the agent with the required interaction information. As a consequence, the computation and processing cost of the interaction information is reduced by creating it offline (a sort of an interaction script) and applying it every time agents need it. This method also has its drawbacks: every time an agent interacts with an object the interaction takes place in the exact same way without taking into account its own characteristics, since the interaction process is defined once and for all. Indeed, this technique describes interaction processes for homogeneous agents. An object has to describe the same interaction process in as many ways as the types of agents that can interact with it, taking into account the fact that they are humanoid or animals for example.

1.2 Objectives

The objective of this thesis is sixfold:

1. *to propose a method and a set of tools to automate the generation of virtual geographic environments:* MAGS is basically a modelling and simulation paradigm which frequently involves VGEs of various extents. Time and efforts spent on the generation of these VGEs are considerable and should be reduced through an automated approach.

2. *to propose a method and associated algorithms to precisely describe virtual geographic environments:* most of current VGE approaches lack precision when dealing with complex

geographic environments. A precise description of a VGE should take into account both geometric and topologic data characterising the geographic environment.

3. *to propose an approach in order to semantically inform the description of virtual geographic environments:* since spatial reasoning often needs to manipulate qualitative information rather than quantitative data, a conceptualized semantic information should be associated with geographic features. This process of qualification aims at supporting the spatial reasoning capabilities of agents.

4. *to propose an abstraction technique which uses a graph-based structure in order to model large-scale and complex geographic environments:* since geographic environments may be complex, large-scale, and populated with various geographic features, the description of virtual geographic environment should be abstracted, structured in order to support such geographic environments. The abstraction of the virtual geographic environment description should take into account the geometric, topologic, and semantic characteristics of geographic environments.

5. *to propose an approach which allows to represent knowledge about the environment and to provide it to spatial agent:* spatial agents need to access, acquire, and reason about knowledge about the virtual environment in order to make decisions that take into account its characteristics. Knowledge about the environment should be represented using a standard formalism and spatial agents should be provided with a mechanism to acquire and reason about it.

6. *to populate virtual geographic environments with spatially-reasoning agents:* MAGS usually involves a large number of agents of different types. Such agents should be able to evolve and interact with the VGE while taking into account both agents' and VGE's characteristics.

1.3 The Proposed Approach

What we propose is a method and associated algorithms to *automatically generate geometrically-precise and semantically-enhanced virtual geographic environments populated with spatially-reasoning agents*. We propose a novel approach to model virtual geographic environments which meets the previously mentioned objectives. First, our approach uses reliable standard spatial data provided by Geographic Information Systems (GIS). In addition, it relies on an exact spatial decomposition technique which fully preserves the geometric and topologic characteristics of geographic features and thus yields a precise description of the virtual geographic environment. Moreover, this description is based on a graph-based structure which allows our approach to handle large-scale geographic environments. This description

is enriched with semantic information in order to better qualify the geographic features. Furthermore, since our aim is to support agents' spatial reasoning capabilities, we propose to enhance the virtual environment's description with conceptual semantic information using well-established knowledge representation techniques. Indeed, since the nature of objects is invariant during the MAGS process, and because the interaction process may vary depending on the agent's type, the conceptual semantic information needs to be generic and interpretable by any type of agent. The conceptual semantic information will be separated in two parts: the first part contains the information inherent to the object type (i.e. the car is red), and the second part contains the information inherent to the agent which uses the object's information to reason and act on the environment (i.e. the agent drives the car). Such a conceptual semantic information stored within the environment extends the agents' knowledge about their environment, without being too specific and detailed, and provides agents with useful information in addition to the geometric and topologic data characterizing the VGE. Path planning and navigation are examples of spatial reasoning algorithms that may leverage our geometrically-precise and semantically-enhanced model of large-scale virtual geographic environments.

1.4 Application Domains

In order to illustrate our approach, we propose the following application domains: *urban simulation*, *radio signal propagation in virtual geographic environments*, and *sensor webs deployment in virtual geographic environments*.

Urban Simulation We propose to use our approach to build a a three-dimensional virtual urban environment populated with agents representing humans, buses, cars, etc. The main novelty of our approach is that we do not rely on representing land-use and other attributes on a regular grid, but instead build a geometrically-precise simulation that uses an exact spatial decomposition technique. Our VGE model also enriches the virtual urban environment's description with semantics in order to qualify the geographic features such as streets of arbitrary orientation, street widths and shapes, and irregular building footprints. The second novelty is the automated and fast generation of such a virtual urban environment. We also provide a set of scenarios involving agents interacting with the informed VGE. These scenarios illustrate the agents' capabilities to detect and avoid collisions with obstacles situated in the VGE as well as path planning, while taking into account the enriched description of the VGE.

Radio Signal Propagation In the real world, radio transmissions are subject to propagation effects which affect the received signals because of geographic and environmental character-

istics (foliage and vegetation, buildings, mountains and hills, etc.). Using our informed VGE, we are able to easily generate a virtual geographic environment which precisely describes the geographic features of the real world. Then, we are able to predict the attenuation effect due to the radio signal's traversal through vegetation area, and buildings using a 3D line-of-sight technique.

Sensor Web Sensor webs can be thought of as distributed network systems composed of hundreds of resource-constrained nodes. Sensor webs are deployed in large-scale geographic environments for in-situ sensing and data acquisition purposes. The deployment of a large number of sensor nodes in a large-scale geographic environment is a complex task. Moreover, the sensing and communication performance of a sensor web relies on the pattern of its deployment within the specific environment. Using our IVGE model, we are able to simulate a sensor web deployment where sensor nodes are modeled using software agents. These sensor agents leverage our knowledge management approach to acquire knowledge about the environment and adapt their behaviors while taking into account both the IVGE's and sensors' characteristics.

1.5 Contributions

In a departure from the substantial literature on so-called *spatial modelling*, the first contribution of this thesis is a geometrically-precise and semantically-enhanced model of virtual geographic environments based on a graph structure that we call an *Informed Virtual Geographic Environments* (IVGE). The second contribution is a methodology for the automated generation of informed virtual geographic environments based on reliable data provided by *Geographic Information Systems* (GIS) and using the *Constrained Delaunay Triangulation* (CDT) technique as a decomposition paradigm. The third contribution is a geometric, topologic, and semantic abstraction of informed virtual geographic environments in order to build a *Hierarchical Topologic Graph* (HTG) which supports the representation of large-scale geographic environments. The resulting HTG uses a standard formalism (Conceptual Graphs) to express semantic characteristics of the virtual geographic environment. The fourth contribution is a *Hierarchical Path Planning* (HPP) algorithm coupled with a Neighborhood Graph (NG) model to provide agents with for efficient motion planning (navigation and path planning) in informed virtual geographic environments. The fifth contribution is an environment knowledge management approach which allows spatial agents to make decisions that take into account the geographic environment characteristics.

From an application perspective, this thesis illustrates the above mentioned theoretical

contributions in domains such as *Urban Mobility*, *Wireless Communications* and *Sensor Webs*. It also provides a set of tools that implement the proposed models and allow users to take advantage of the enriched description of the virtual geographic environment.

1.6 Organisation of the Thesis

The thesis is composed of two parts: 1) *the state of the art* which presents the context of our work and its related research fields and 2) *the contributions* which proceed with a full explanation of the IVGE model that we propose, the means of its enhancement, and the way we leverage it for motion planning purposes (Figure 1.1).

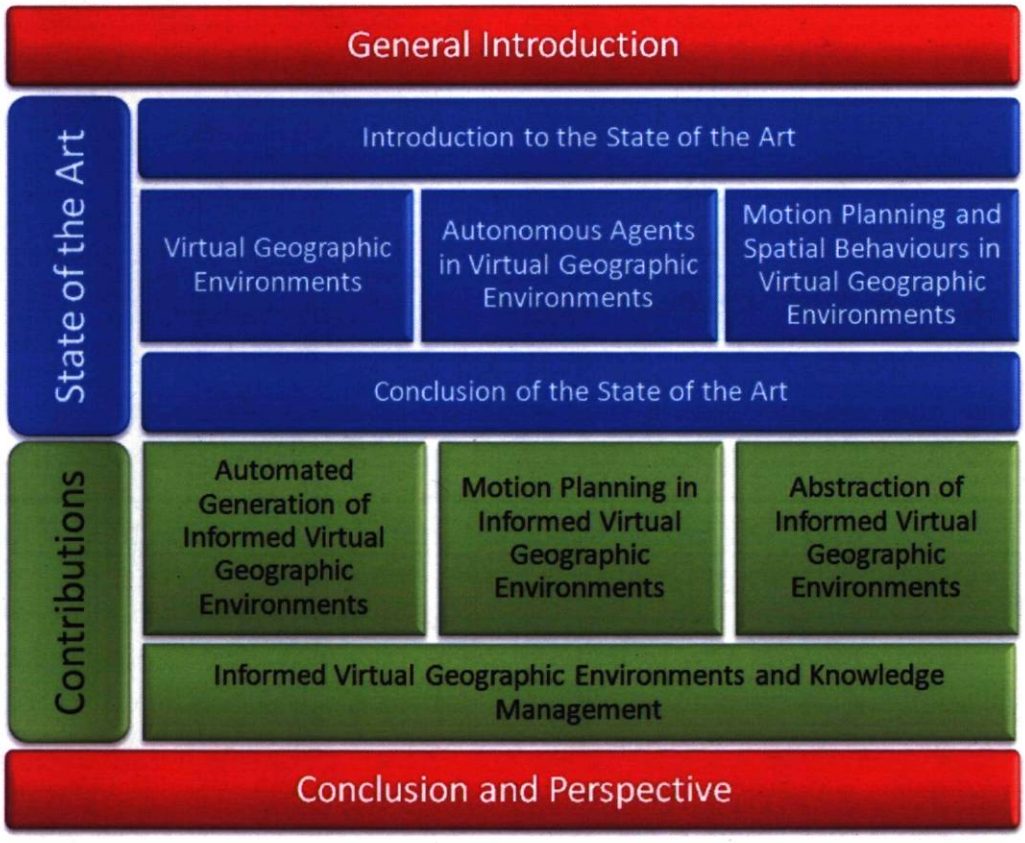


Figure 1.1: Organisation of the Thesis.

The first part is itself composed of three chapters: *Virtual Geographic Environments*, *Autonomous Agents in Virtual Geographic Environments*, and *Motion planning and Spatial Behaviours in Virtual Geographic Environments*.

Chapter 2 provides a survey on the representation of virtual geographic environments.

First, it introduces *Geographic Information Systems* and presents the two common spatial decomposition techniques: the *exact* and the *approximate* spatial decompositions. Then, it discusses the use of semantic information in the description of virtual geographic environments. It also outlines the need for a standard formalism to represent such semantic information.

Chapter 3 details the concepts of situated agents and multi-agent systems and focuses on the importance of the environmental characteristics in the agent's decision making process. It also highlights the way agents interact with their environments in terms of perception, reaction, decision, and action.

Chapter 4 is the last chapter of this first part and it introduces the motion planning research field. First, it provides an overview of common path planning algorithms (*Dijkstra* and $A*$). Then, it introduces the hierarchical path planners that deal with complex large-scale geographic environments. It also presents previous works on spatial behaviours and the way that agents reason about the environment's description and behave accordingly.

Finally, Chapter 5 draws conclusions from this state of the art and outlines the limits of currents approaches.

The contributions part of this thesis is composed of four chapters: *Automated Generation of Informed Virtual Geographic Environments*, *Abstraction of Informed Virtual Geographic Environments*, *Motion planning in Informed Virtual Geographic Environments*, and *Informed Virtual Geographic Environment and Knowledge Management*.

Chapter 6 presents our methodology for the automated generation of informed virtual geographic environments. First, it details the different steps which compose this methodology. Then, it introduces a scenario which illustrates the application of our methodology in order to address the issue of radio signal propagation and attenuation in informed virtual geographic environments.

Chapter 7 presents how to abstract IVGEs in order to support large-scale and complex geographic environments. This chapter details the three processes involved in the abstraction mechanism; (1) the geometric abstraction, (2) the topologic abstraction and, (3) the semantic abstraction. It also introduces a holonic approach coupled with our abstraction model which are used to build the *Hierarchical Topologic Graph* (HTG).

Chapter 8 presents our optimised *Hierarchical Path Planning* (HPP) algorithm which leverages the hierarchical topologic graph resulting from the abstraction process. In addition, it details how we support spatial agents' navigation and obstacle avoidance in informed virtual geographic environments using a Neighbour Graph (NG) approach.

Chapter 9 presents our approach to manage knowledge about the environment in order to support spatial agents' decision making. This approach uses a standard formalism to express the semantic information that characterises the geographic environment and involves an inference engine in order to allow agents to get knowledge about virtual geographic environments. In order to illustrate our environment knowledge management approach, we present two scenarios in the fields of urban mobility and sensor webs.

Finally, Chapter 10 draws conclusions from our work, outlines its limits, and discusses avenues for future research and applications.

Part I

State of the Art

Introduction to the State of the Art

Multi-Agent Geo-Simulation (MAGS) is a modeling paradigm which is characterized by an explicit spatial environment called Virtual Geographic Environment (VGE) and situated agents which evolve in and interact with this VGE [BT04]. MAGS has a great potential when it comes to explaining the subtle interactions of heterogeneous actors in complex systems, taking into account the geographic aspect of the simulation environment [TB05]. The characteristics of the situated agents (autonomy, proactiveness, perception, navigation, etc.) and the spatial features of the simulation environment make MAGS an attractive approach to develop simulations of complex systems involving agents interacting with each other and with the geographic environment [DHK+07]. In addition, the complexity of the simulation models and their visualization capabilities (cartographic visualization, 2D and 3D displays) make them more realistic and usable for decision-making purposes [AM05]. Thus, MAGS approaches potentially open numerous avenues for exploratory and applied simulations in different fields [BT04].

There are several advantages to use a MAGS approach integrating agent-based models and GIS data in order to simulate various phenomena in a variety of domains such as traffic simulation, crowd simulation, environment monitoring, and changes of land use and cover, to name a few [DHK+07]. Indeed, MAGS facilitates the simulation of complex phenomena using either *micro-simulation* or *macro-simulation* models [TB05]. *Micro-simulation* models operate at the level of the individual behavioural entity, such as a person, a car, or a building. Such models simulate large representative populations of these low-level entities in order to draw conclusions that apply to higher levels of aggregation such as an entire city or an entire country [TB05]. This type of model is distinct from *macro-simulation* models whose explanatory variables already represent collective properties of the complex phenomena. Macro-simulation models evaluate the complex phenomena as a whole without consideration of the characteristics and features of individual entities involved in such phenomena [TB05]. An example of such an aggregate explanatory variable might be the national unemployment rate of a country. Certain types of modeling problems are best dealt with using micro-simulation whereas for others an aggregate approach is more appropriate. However, when it comes time to study and analyze interactions occurring between individual agents or between individual

agents and their virtual geographic environment, macro-simulation models are inadequate since they are not able to capture interactions at the individual level [HC07].

For example, MAGS is frequently used to simulate and examine traffic flow and congestion and to understand the interaction of vehicles on the roadway [Wan05]. Macro-simulation models evaluate traffic flow as a whole without consideration of the characteristics and features of individual vehicles in the traffic stream [WCC+07]. In contrast with macro-simulation approaches, micro-simulation models simulate the individual vehicles in the traffic stream and consider the features and characteristics of the individual vehicles and use car-following logic and algorithms to predict and model the movement of each vehicle in the traffic stream [WCC+07].

Several research works applied the MAGS approach to simulate dynamic and complex phenomena in geographic environments [DHK+07]. These research works succeed to measure and model these phenomena and help, for example, to understand how people move in a geographic environment or in an urban setting [TB05]. However, we found few MAGS-based research studies attempting to simulate what we call *Spatial Behaviors*. Spatial behaviors are basically behaviors involving the apprehension of spatial features of the environment in large-scale and complex geographic environments. The nature of the geographic environment, the spatial behaviors to be simulated, and the characteristics of the interactions between the agents and the environment in the simulation (spatial interaction), make this kind of simulation a challenge. Several important features must be considered when building a MAGS model at the micro-level in order to simulate agents' spatial behaviors, among which we mention: (1) *an explicit representation of the virtual geographic environment*; (2) *a situated agent model which is able to evolve within this virtual geographic environment*; and (3) *efficient spatial behaviours algorithms which allow agents to interact with the virtual geographic environment*.

In order to enable agents to interact with their virtual environment, it is important to provide an explicit representation of the geographic environment in which agents evolve. In order to capture and analyze such interactions which occur between agents on the one hand and between agents and their geographic environments on the other hand, we use the micro-simulation approach. However, the micro-simulation approach requires a detailed description of the geographic environment as well as of situated agents.

On the one hand, the virtual geographic environment description should provide agents with quantitative and qualitative information in order to extend their knowledge about their environment and to let them reason about it in order to achieve their goals. Indeed, the virtual geographic environment description should be geometrically-accurate in order to capture the complexity of the geographic environment and the geographic features it contains. Moreover,

this description should include the topological characteristics of the geographic environment in order to support agents' motion planning algorithms while taking into account dead-end areas, corridors, and crossroads. In addition, this description should also be enriched by semantic information which qualifies the geographic features in order to allow agents' spatial behaviour algorithms to take into account the geometric, topologic, and semantic characteristics of the geographic environment in which they evolve. Chapter 2 presents and discusses different techniques to build an explicit spatial representation of geographic environments. It also outlines the complexity of virtual geographic environments' generation.

On the other hand, situated agents models should take advantage of such an enriched description of the virtual geographic environment in order to reason about it. Chapter 3 and Chapter 4 introduce the notion of situated agents and emphasize the need to take into account the characteristics of the geographic environments in order to support spatial behaviours such as motion planning including path planning and navigation algorithms.

In this state of the art, Chapter 2 provides a short survey of different techniques and models used to build virtual geographic environments. It also discusses the use of semantic information to describe virtual environments and emphasizes the need for using a standard representation. Next, Chapter 3 introduces the concept of autonomous agents evolving in and interacting with a virtual geographic environment. Finally, Chapter 4 offers an overview of motion planning and spatial behaviours of agents within virtual geographic environments.

Chapter 2

Virtual Geographic Environments

A *virtual environment* is a computer generated scene, composed of objects, in which an autonomous agent operates [Bad06]. This definition is pretty vague and leaves a very broad panel of interpretation of what the *computer generated scene* can be: it can be *realistic*, attempting to faithfully copy the real world in every aspect, or *imaginary* and completely invented. The vagueness of this definition seems to be intentional, and implies a large richness in the nature of the virtual environment. Useful overviews of virtual environments and their many applications are provided in [FMJ06, LB09]. A virtual environment may be thought of as a space subdivision which converts complex geometric data, made up of a great number of polygons, in a more or less informed database. The geometric data may be provided by *Computer Aided Geometric Design* (CAGD) tools in order to design curves and figures in two-dimensional ($2D$) space, or surfaces and solids in three-dimensional ($3D$) space for computer animations purposes [Far01b]. Geometric data may also be provided by *Geographic Information Systems* (GIS), evolving the concept of *Virtual Geographic Environments* (VGE) which is widely used for spatial analysis and decision support purposes [BRR+05].

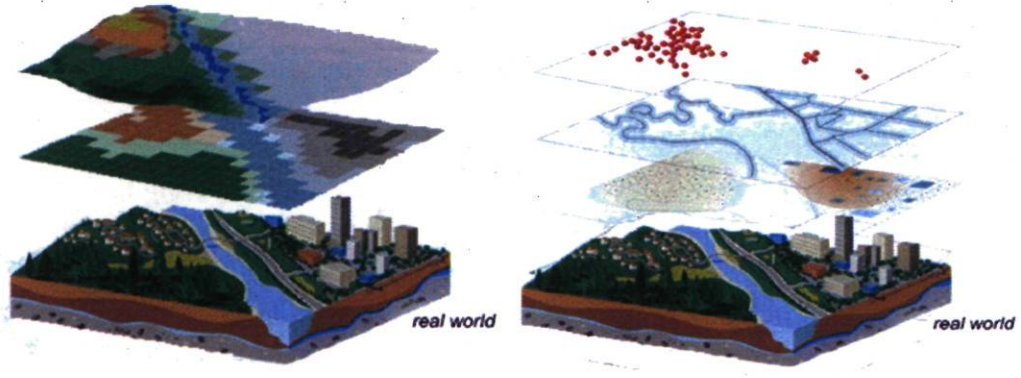
Building a VGE based on reliable GIS data is a big challenge. Indeed, a geographic environment may be complex and large-scale, and its creation is difficult and needs large quantities of geometrical data originating from the environment characteristics as well as semantic information that qualifies space (building, road, park, etc.). To be able to create a VGE, one should have an explicit representation of space using space decomposition techniques.

This chapter aims to describe relevant components in the creation of a VGE, as well as some current research issues. First, Section 2.1 introduces the concept of *Geographic Information System* as a fundamental source of georeferenced data. It presents the different models used to describe geographic data and discusses their advantages and disadvantages. Second, Section 2.2 introduces approximate space decomposition approaches. Section 2.3

provides a short survey of exact space decomposition approaches and focuses on Delaunay Triangulation techniques. Section 2.4 introduces and discusses the topologic approach to represent virtual geographic environments. Section 2.5 presents the abstraction of virtual environments and Section 2.6 outlines the integration of semantic information in virtual environments' descriptions. Finally, Section 2.7 concludes this chapter with a synthesis of the proposed models.

2.1 Geographic Information Systems

A *Geographic Information System* (GIS) is a system of hardware, software and procedures to facilitate the management, manipulation, analysis, modelling, representation and display of georeferenced data to solve complex problems regarding planning and management of resources [Goo06]. GISs have grown over the past two decades as an essential tool for urban and resource planning and management [Chr01]. Their capacity to store, retrieve, analyse, model and map large areas with huge volumes of spatial data has led to an extraordinary proliferation of applications [Goo06]. GISs offer two primary data models to describe a geographic environment: *raster* and *vector* data representations [AZ04]. In the following sub-sections, we first present the raster model and then the vector model of GIS data. We then discuss the use of GIS data in agent-based modelling approaches.



(a) (b)

Figure 2.1: Raster model (a) and *Vector* model (b) of GIS data [Chr01].

2.1.1 Raster Model

The *raster* representation of GIS data is a method for the storage, processing and display of continuous fields of spatial data [LGMR02, AZ04]. Within the raster data model, a region

of interest is divided into discrete units, which form a regular grid of cells (Figure 2.1(a)). Each cell is typically rectangular in shape, but not necessarily square [Chr01]. Each cell within this matrix contains a single attribute value and a separate raster is created for each attribute [LGMR02]. The spatial location of each cell is implicitly contained within the ordering of the matrix and a specification of an origin or extent and a cell size (or resolution) [FR02]. A contiguous area containing the same attribute value may be recognised as a polygon, however raster structures cannot identify the precise boundaries of areas such as polygons [Chr01]. With the raster data model, spatial data are not continuous but divided into discrete units [LGMR02].

2.1.2 Vector Model

The vector format is defined by the explicit representation of geographic data [ZLS+08]. Point features are defined by one coordinate pair, a vertex [LGMR02]. Vector features are characterized by the use of sequential points or vertices to define linear segments that approximate a curve (Figure 2.1(b)). Vector lines are often referred to as arcs and consist of a string of vertices terminated by a node [Chr01]. A node is defined as a vertex that starts or ends an arc segment. Polygonal features are defined by a set of arcs that bound the polygon [AZ04]. In vector representation, the storage of the vertices for each feature is important, as well as the connectivity between features, e.g. the sharing of common vertices where features connect [FR02].

2.1.3 GIS and Agent-Based Modelling

Data used to build virtual environments in the computer and behavioral animation research fields are mostly provided by *Computer Aided Design and Graphics* (CADG) systems. CADG has been used to construct virtual cities [FRMS+98, FBT99] and urban contexts [Don97, TD00, MPB03], virtual train stations [ST05, PDB06], and virtual shopping malls [AM05]. In contrast to GIS data, CADG data are used to create precision drawings or technical illustrations for animation and visualization purposes without taking into account the georeferencing of spatial data; that is, explicit mapping of a data point to its geographic location on the surface of the earth. The CADG data model lacks the definition of spatial relationships between features that is defined by GIS data models both in raster and vector formats. In addition, interactions between agents and the environment are most of the time very simple, permitting the agents only to perceive and avoid obstacles in a 2D or 3D virtual scenes [Don97]. This is due to the fact that the description of virtual environments is often limited to the geometric level, while it should also contain topological and semantic information for other types

of applications.

In [NN04], Najlis and North discuss the growing interest for the integration of GIS and agent-based modelling systems ([Par04, BBA06, TB05, BRR+05]; to name a few). Examples of recent applications include pedestrian dynamics, urban growth models and land use models. For agent-based modellers, this integration provides the ability to manipulate agents that are related to actual geographic locations. For GIS users, it provides the ability to model the emergence of phenomena through individual interactions of features on a GIS over time and space [NN04].

Several different levels of integration of agent-based models and GIS are possible. Models may run on grid-based representation of geographic environments, implementing simple spatial functionalities within the agent-based model. Agents' models may read real-world spatial data from a GIS and also potentially write output into a format readable by GIS. When significant GIS functionality is required at run time, integrated models which implement both agent-based models and GIS functionality dynamically may be constructed. Three approaches to such models are identified in [HWS02]: (1) models that use separate GIS and agent-based programs/libraries and communicate via files written to disk; (2) models that use separate programs but communicate via a shared database or virtual memory, (3) and standalone models that implement GIS functionalities within the agent-based model. In the latter category, agents evolve and interact with an explicit representation of the geographic environment built using GIS data. The semantically-enhanced virtual geographic environment model which constitutes the focus of this thesis falls in this category.

2.1.4 Synthesis

There are several advantages and disadvantages for using either the vector or raster data model to represent spatial data. On the one hand, raster models provide an easy and fast access to the geographic location of each cell since it is implied by its position in the cell matrix. However, raster models are sensitive to the resolution at which the data are represented. As a consequence, it is difficult to integrate different geographic features represented at different scales with distinct resolutions. Moreover, raster data inherently reflect a single attribute or characteristic for an area. Therefore, the processing of the several attributes and their associated raster layers may be cumbersome if large amounts of data exist.

On the other hand, vector models preserve the geometric and topologic characteristics of the geographic environment since GIS data can be represented at their original resolutions and forms without approximation. Besides, they provide a more accurate location of geographic

features. In addition, vector models enable precise visualization of geographic environments. However, vector models are complex to manipulate and their interpretation may be processing intensive [Mol98]. For effective processing, Molenaar suggested in [Mol98] that vector data models be mapped to graph-based structures using a topologic approach (see Section 2.4).

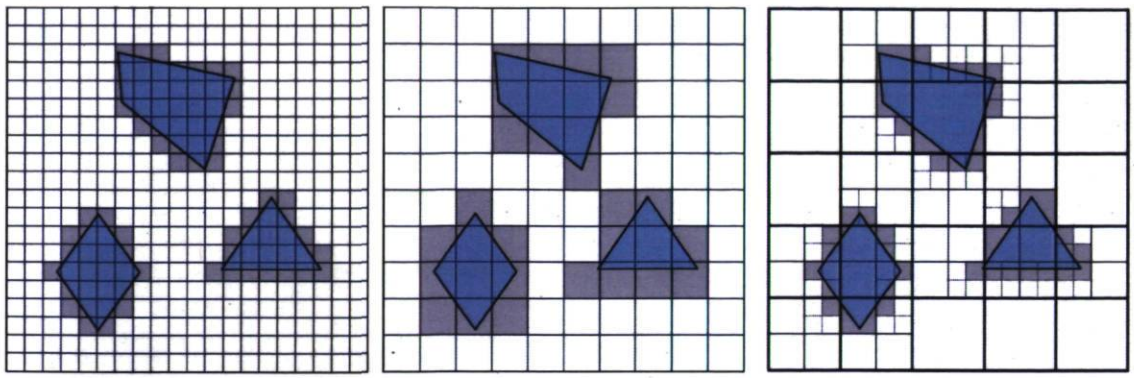
2.2 Approximate Space Decomposition

The *Approximate Space Decomposition* (ASD) techniques aim to construct a collection of non-overlapping regions so that the union of all these regions ***approximately*** covers the original space. These techniques are mostly used in the computer animation research field. Indeed, using such techniques it is easy to build an approximate subdivision of space in which *free* and *occupied* regions denote navigable and impassable areas of the environment. Based on free and occupied regions, it is possible to support the agents' navigation and path planning capabilities. In this section we provide a short survey of ASD techniques. First, we introduce the *grid-based* models (Section 2.2.1). Then, we present the *roadmaps* and the *potential fields* techniques. We conclude with a synthesis on these approximate space decomposition techniques.

2.2.1 Grid-Based models

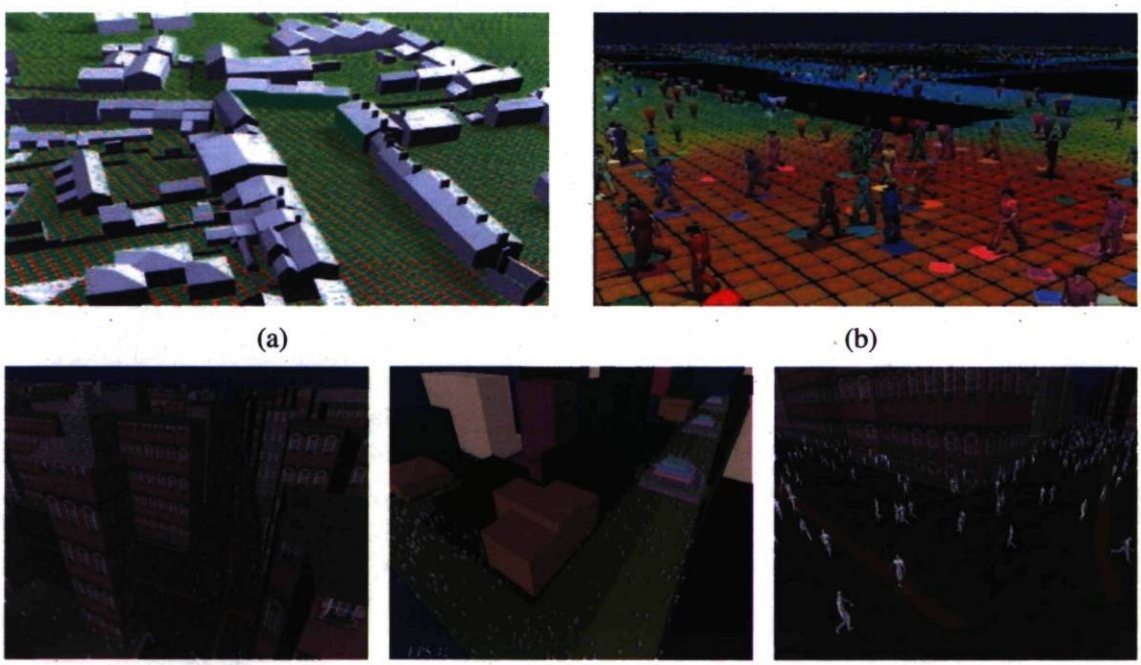
The first approximate representation model uses regular grids formed of square cells in two dimensions (Figure 2.2(a)) and cubes in three dimensions. The environment is completely covered by these cells, which can have three states: *free*, *partially obstructed*, and *totally obstructed* or *obstacle*.

The accuracy of the obtained representation basically depends on the cell size: the larger cells are, the less precise is the representation. Of course, increased precision leads to an increase in memory use. The memory footprint of this method is thus its first weak point. It directly affects the complexity of path search in the environment. In order to reduce this problem, an extension of this model has been proposed in the form of *hierarchical* grids [War95, ST05]. This method describes the space by a set of overlay grids which are increasingly accurate, organized as a tree. The algorithm examines the environment, starting with the least specific cells, then recursively cutting partially obstructed cells in four parts for 2D (quadtree Figure 2.2(c)), or eight parts for 3D (octree). The algorithm stops when a sufficiently precise subdivision is reached. This method is advantageous when the environment contains a limited number of obstacles.



(a) (b) (c)

Figure 2.2: Approximative decomposition by grids considering a fine (a) and coarse (b) grain resolutions; (c) Grid abstraction using a three levels quatree. White boxes are free, grey are obstacles.



(a) (b)

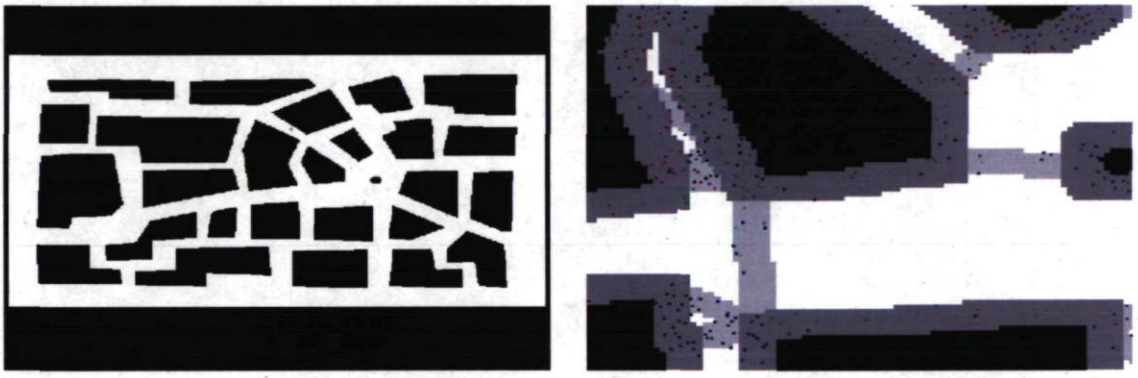
(c) (d) (e)

Figure 2.3: (a) and (b): examples of grid-based virtual environments for computer animation purposes; (c), (d), and (e): views of central London populated with 10 000 humans using a grid-based virtual environment [TC00].

The grid-based model has been widely used because of its simple yet rich representation (Figure 2.3), but it has two main drawbacks: 1) the grid can not capture the topological relationships between regions; 2) it usually suffers from high cost in both time and memory use when fine grids are used [ST05]. Moreover, this model is also difficult to generalize to any

kind of environment, particularly if they involve obstacles aligned with the axes. Finally, it should be noted that the discretization level of the grid introduces an implicit limit to population densities that can be simulated [ABG05]. Grid-based methods are frequently used in the computer and behavioral animation fields [Kor85, TC00] because of their implementation simplicity and their operation speed. We can thus see animations of thousands of individuals based on this mode of representation (Figure 2.3(c), Figure 2.3(d), and Figure 2.3(e)) [TC00].

Bandi *et al.* proposed an intuitive technique to produce a uniform cell decomposition in the context of a pedestrian simulation project [BT98]. With such a decomposition, a pedestrian's global motion results from successive local motions from cell to cell, guided by probabilistic rules [KS02]. Loscos *et al.* presented a behavioral model based on a 2D approximate environment decomposition: after identifying buildings, sidewalks, and crosswalks in a pre-processing step, characters followed successive goals associated with cells [LMM03]. Figure 2.4 illustrates the environment map obtained using Loscos *et al.*'s approach.



(a) Binary image representing the virtual city: white for the ground and black for the buildings.

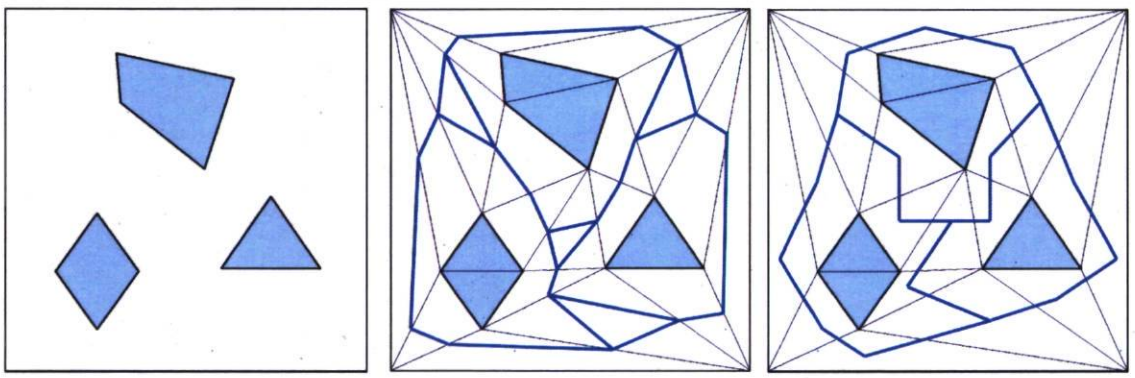
(b) Modelling pavements (in gray) and pedestrian crossings (in light gray).

Figure 2.4: A 2D model of a virtual city as proposed by Loscos *et al.* [LMM03].

2.2.2 Roadmaps

The roadmap approach consists of computing a network of standardized paths (lines, curves) passing through free space. Once the roadmap has been constructed, a path can be calculated by connecting the initial and final positions in the network and finding a path in the roadmap (Figure 2.5). *Probabilistic Roadmaps*, *Visibility Graphs* and *Generalized Voronoi Diagarams* are examples of roadmap methods.

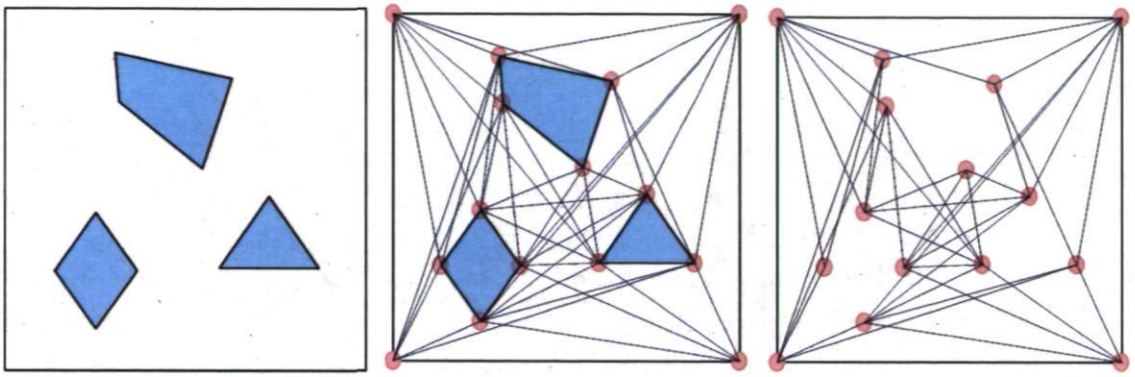
Probabilistic Roadmaps The roadmap approach has been extended by Kavraki *et al.* to *Probabilistic Roadmaps* (PRM) [KSLO96]. They applied this approach to model collision-free regions for robots in static workspaces [KKL98], while Arikan *et al.* computed a visibility graph to simulate virtual agents in a grid-based model of the environment [ACF01]. The PRM-based approach was later adapted to plan individual paths and the underlying motions of characters [CL03, PLS03, HBL05]. Improvements on the PRM-based approach have been proposed to better handle navigation. Bayazit *et al.* integrated flocking techniques with roadmap-based path planning [BLA02, BLA03] . The flock members store global information, such as dead-ends, in the roadmap nodes in order to warn other members. Sung combined probabilistic roadmaps with motion graphs to find paths and animations to steer characters towards a goal [SKG05]. PRM-based approaches suit high-dimensional problems well. However, roadmap techniques are not the most efficient ones for navigation planning when dealing with three-dimensional virtual environments.



(a) Original Environment. (b) First example of Roadmap (c) Second example of Roadmap.

Figure 2.5: Two examples of *RoadMap* generation using a Delaunay Triangulation.

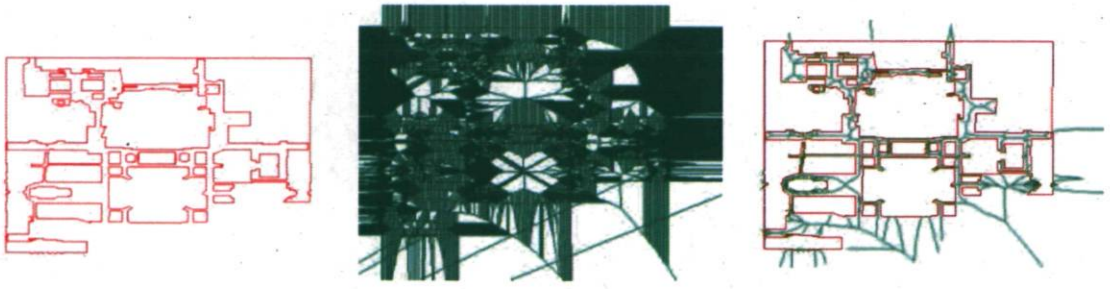
Visibility Graphs A visibility graph is a graph of intervisible locations [Pet99]. Each node or vertex in the graph represents a point location, and each edge represents a visible connection between points (that is, if two locations can see each other, an edge is drawn between them). The resulting graph minimizes distances and provides minimum length paths. The size of the generated graph depends on the number of intervisible point locations. The complexity of building visibility graphs increases when dealing with complex environments with a large number of obstacles. Visibility graphs have practical uses, for example, to calculate the placement of radio antennas, or as a tool used within architecture and urban planning through visibility graph analysis. Visibility graphs are also used in mobile robotics as a (generally offline) motion planning tool when the geometry of the environment is known.



(a) Original Environment. (b) Computation of VG. (c) Obtained VG.

Figure 2.6: An example of Visibility Graph computation using a 2D environment.

Generalized Voronoi Diagrams Given a set of primitives, called Voronoi sites, the *Generalized Voronoi Diagram* (GVD) partitions the space into regions, one per site, such that all points in a region have the same closest site according to some given distance function. Many variants of these diagrams can be considered: by taking sites of different shapes or natures, associating weights to the sites, changing the underlying metrics, or using individualized distance functions for the sites [Aur91, AK00, OOB00]. GVD are widely used in many scientific fields such as computer graphics, geometric modelling, shape analysis, robot motion planning or scientific visualization (Figure 2.7).

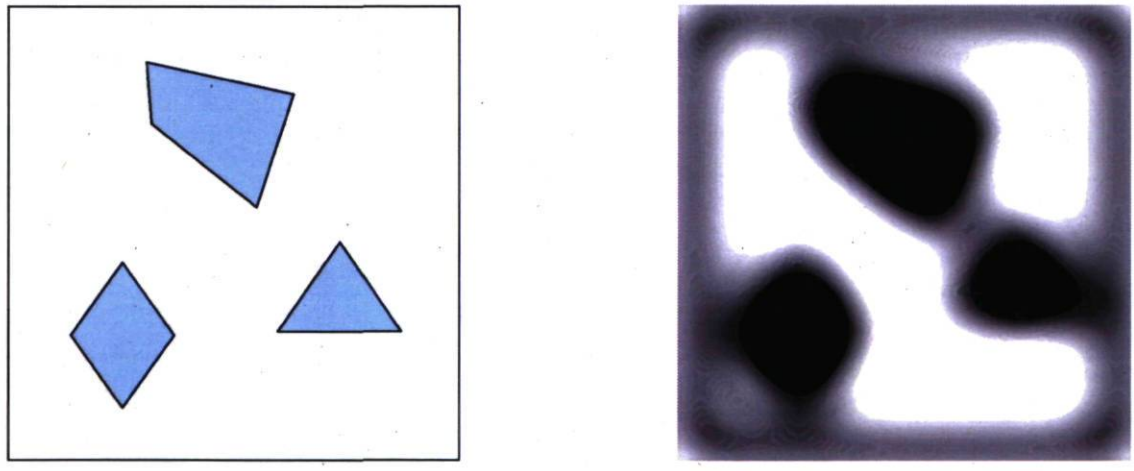


(a) A 2D environment. (b) The computed GVD. (c) The filtered GVD.

Figure 2.7: Example of a *Generalized Voronoi Diagram* for a planar region (the northern half of Columbia's Morningside Campus) with specified obstacles. The diagram is filtered by eliminating those edges which have one or both endpoints lying inside any of the obstacles [ASG+01].

2.2.3 Potential fields

Potential fields are a part of approximate space decomposition techniques. They use the metaphor of magnetic field or gas spreading, first mentioned in [Kha86]. This method is usually efficient because it can easily determine the position of a moving agent using the sum of the potential fields [HKLC98]. The potential fields approach easily lends itself to extensions. For example, since potential fields are additive, adding a new obstacle is easy because the field for that obstacle can be simply added to the old one. However, potential fields have some limits, especially their computational complexity. Moreover, the method's major drawback is the existence of local minima. Because the potential field approach is a local rather than a global method (it only considers the immediate best course of action), the moving entity can get *stuck* in a local minimum of the potential field function rather than heading towards the global minimum, which is the target destination. This is frequently resolved by coupling the method with techniques to escape local minima, or by constructing potential field functions that contain no local minima (Figure 2.8).



(a) Original Environment. (b) Computation of Potential Fields

Figure 2.8: An example of Potential Fields corresponding to the original environment: clear (grey) areas correspond to attractive regions (minima) and dark (black) areas correspond to reject (repulsive) regions.

2.2.4 Synthesis

The main quality of ASD techniques is the facility to build and maintain an approximate number of convex cells representing the environment. This number is not linear with the number of points used to describe the geographic features located in the environment. It depends on the required *accuracy* and *scale* of the approximate representation in addition to the inherent

complexity of the geometry representing the environment. In order to deal with large-scale environments, the approximate space decomposition approach offers the possibility to adjust the size of cells and to build an hierarchical representation of the environment. This approach uses simple geometric forms (squares and rectangles) of cells which are easy to use and process. However, it only *approximates* the geometric complexity of the environment rather than *fully preserving* it as exact space decomposition approaches do.

2.3 Exact Space Decomposition

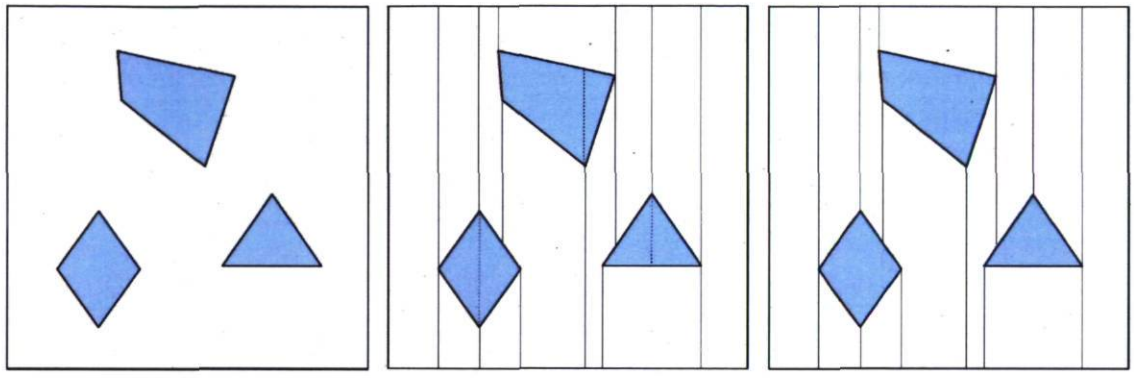
The *Exact Space Decomposition* (ESD) aims to organize spatial data in order to accurately represents the environment while preserving its geometric and topologic characteristics. As stated in [Lat91], the exact cell decomposition techniques decomposes an environment E into a collection K of non-overlapping cells so that the union of all the cells ***exactly*** equals the environment E, i.e., $E = \bigcup_{k \in K} k$. Two cells $k_1, k_2 \in K$ are *adjacent* if and only if $k_1 \cap k_2$ is a line segment of non-zero length (i.e., not a single point). The geometry of cells should be simple (triangles, trapesoides, etc.) and convex so that it is easy to compute a path between any two positions located in a cell. It should be pretty easy to test the adjacency of two cells, i.e., whether they share a boundary. It should also be pretty fast to find a path crossing the portion of the boundary shared by two adjacent cells.

Much work has been done on decomposing space into convex regions. First, minimal convex decomposition of a polygonal region was investigated. It was proved that the number of cells in a minimal decomposition is linear to the size of the input, $O(n)$, and that it can even be computed in the time polynomial in the number of vertices n [SW06]. However, Lingas has shown that holes in the configuration space make the problem NP-hard [Lin82]. So, minimal decomposition of polygon areas is neither practical nor fast. Therefore, non-optimal decompositions were examined. In the following sub-sections, we focus on the most used non-optimal decomposition techniques: the *trapezoidal* decomposition and the *Delaunay* triangulation.

2.3.1 Trapezoidal Decomposition

The *Trapezoidal Decomposition* (TD) (also called *vertical decomposition*) subdivides the environment into trapezoidal cells [Cha87]. The algorithm input is a collection of line segments $S = s1, s2, ..., sn$ such that these line segments do not intersect except at their endpoints. To construct a trapezoidal decomposition, one can imagine *shooting a bullet* vertically upwards

and downwards from each vertex in the polygonal subdivision until it hits another segment of S. The resulting *bullet paths*, together with the initial line segments, define the trapezoidal decomposition of the space. An important fact to observe about each trapezoid is that it is defined by exactly four entities from the original subdivision: a segment on top, a segment on the bottom, a bounding vertex on the left, and a bounding vertex on the right. An example of a trapezoidal decomposition is shown in the Figure 2.9. However, the trapezoidal decomposition has a few drawbacks. The cells are not very natural-looking. They tend to be long and skinny, which is not suitable for position verification. A trapezoidal decomposition can be built in $O(nlogn)$ time [Cha87].



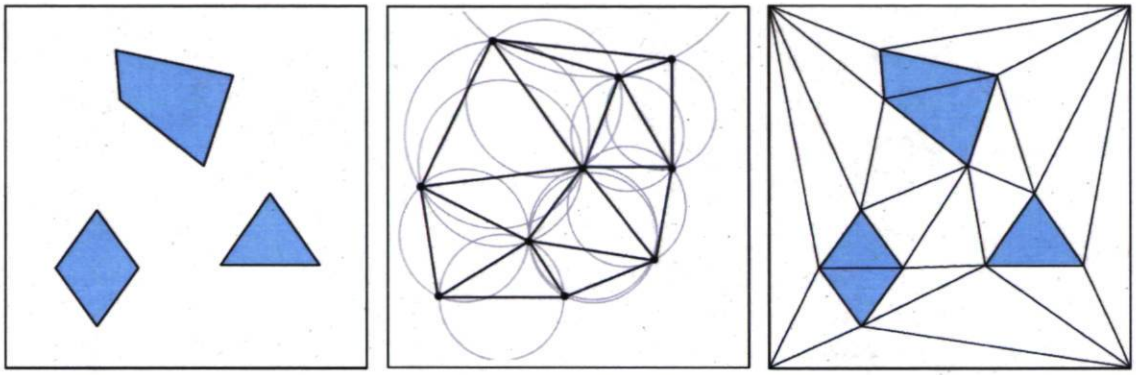
(a) Original Environment. (b) Incremental update of trapezoides. (c) Exact decomposition using TD.

Figure 2.9: The exact space decomposition using Trapesoidal Decomposition.

2.3.2 Delaunay Triangulation

A *Delaunay Triangulation* for a set **P** of n points in the plane is a triangulation DT(**P**) such that no point in **P** is inside the circumcircle of any triangle in DT(**P**) (Figure 2.10(b) and Figure 2.10(b)). In contrast with trapezoidal decompositions, Delaunay triangulations maximize the minimum angle of all the angles of the triangles in the triangulation; they tend to avoid skinny triangles. The DT, with n points and d dimensions, offers several relevant properties to accurately represent a spatial environment (Figure 2.10). First, the union of all triangles in the triangulation is the convex hull of the points. Also, a DT contains at most $O(n^{d/2})$ triangles. Several algorithms for computing Delaunay triangulations rely on fast operations for detecting when a point is within a triangle's circumcircle and an efficient data structure for storing triangles and edges. Some algorithms include the *Flip* technique [DBCVKO08] which can take $O(n^2)$ edge flips, but they do not extend to three dimensions. The *Incremental* algorithm builds a DT with an overall $O(n^2)$ runtime [PCTC07]. The *Divide and Conquer* algorithm recursively draws a line to split the vertices into two sets [Lea92]. A DT is computed

for each set, and then the two sets are merged along the splitting line. The merge operation can be done in time $O(n)$, so the total running time is $O(nlogn)$ [Lea92]. The *Sweepline* algorithm uses a sweepline technique to build a DT achieves $O(nlogn)$ runtime in the planar case (2D) [For97]. Recently, Attali *et al.* proposed an algorithm to build a 3D DT in $O(nlogn)$ runtime [ABL03]. When representing geographic environments, we usually want to be able to compute a modified version of the Delaunay triangulation in which user-defined edges or *constraints* may be specified. Although the Delaunay triangulation technique accurately represents space while avoiding skinny triangles, it does not preserve constraints referring to geographic features such as walls, roads, etc.



(a) Original Environment. (b) No points inside the circumcircle of any triangle. (c) Exact decomposition using DT.

Figure 2.10: The exact space decomposition using DT.

2.3.3 Constrained Delaunay Triangulation

A *Constrained Delaunay Triangulation* (CDT) is a triangulation of a set of points that has to include among its edges a given set of segments joining the points. The corresponding edges are called *constrained edges*. The endpoints of constrained edges are of course vertices of the triangulation. However, the triangulation may include other vertices as well. An example of a CDT is depicted in Figure 2.11. The boundary edges are preserved and not split into smaller edges by avoiding the insertion of additional points (Figure 2.11(a)). Their presence in the CDT is ensured through local modifications where Delaunay edges are removed or flipped. The resulting DT is said to be constrained by the boundary since elements near the boundary are not guaranteed to satisfy the Delaunay criterion as illsutrated in Figure 2.11.

The Constrained Delaunay Triangulation technique provides the advantages of the Delaunay Triangulation and also allows user-defined edges or *constraints* to be specified and

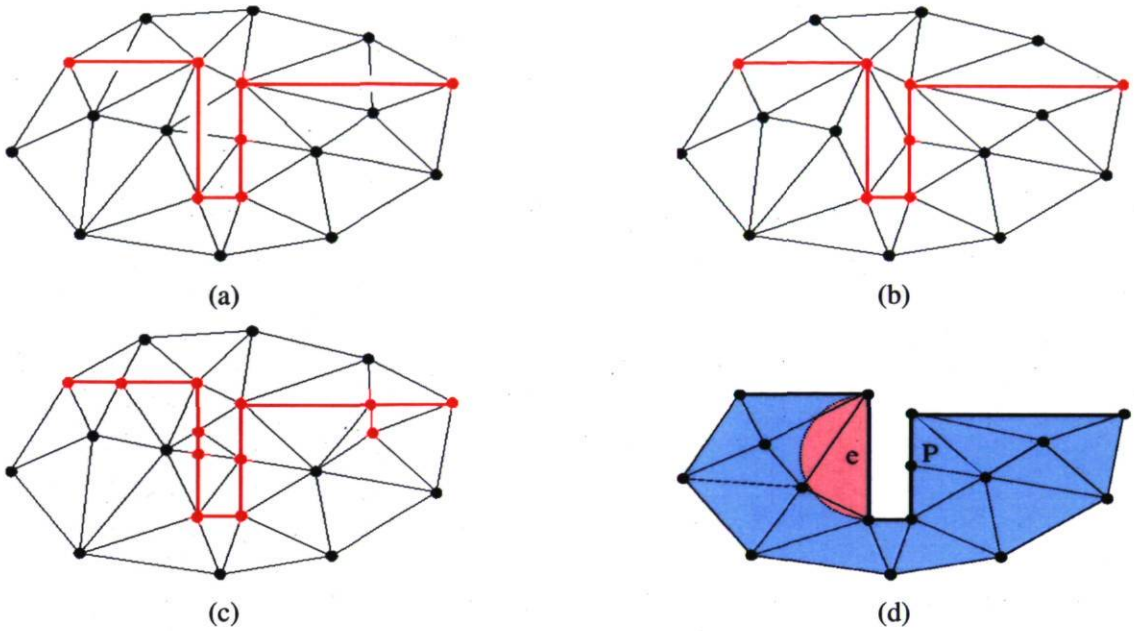


Figure 2.11: Illustration of a CDT building process [FP93]: (a) boundaries which are not conform with DT;(b) a constrained DT; (c) a conforming DT; (d) a CDT with a non-Delaunay edge e. The point P does not affect edge e. The half of the smallest circle which lies inside the mesh is highlighted.

preserved within the final exact space decomposition.

2.3.4 Synthesis

In this section, we introduced a few exact space decomposition techniques, namely: Trapezoidal decomposition, Delaunay and Constrained Delaunay triangulations. the trapezoidal decomposition technique is easy to build but it usually generates skinny shapes which are not suitable for spatial reasoning algorithms such as path planning and navigation. In contrast with the trapezoidal decomposition technique, the Delaunay triangulation allows to maximize the minimum angle of all the angles of the triangles. The constrained triangulation technique performs as well as the classical Delaunay Triangulation (usually in $O(nlogn)$ runtime) while allowing to specify user-defined edges as *constraints* in order to preserve them within the generated triangulation. Such a property is of interest when targeting an accurate representation of geographic environments which contains geographic features of various extents. Indeed, the boundary of these geographic features (i.e. buildings, roads, walls) must be preserved as constraints.

The main quality of these exact space decomposition methods is that they accurately

represent the environment using a number of polygonal cells. This number is only linear with the number of points used to describe the environment rather than with the scale of the environment. It does not depend on the accuracy of the targeted representation, but only on the inherent geometric complexity of the environment. The second property is the automatic adjustment of the cells' size to the density of obstacles within the environment. Exact space decomposition methods allow preservation of the geometric and topologic information of the geographic environment.

2.4 Topological Approach

Topology is a mathematical approach structures data based on the principles of feature adjacency and feature connectivity. It is in fact the mathematical method used to define spatial relationships. Without a topologic data structure in a vector-based GIS, most data manipulation and analysis functions would not be practical or feasible [Chr01]. The most common topological data structure is the [arc,node] data model. This model contains two basic entities, the *arc* and the *node*. The arc is a series of points, joined by straight line segments, that start and end at a node. The node is an intersection point where several arcs meet. Nodes can also occur at the end of a dangling arc, e.g. an arc that does not connect to another arc such as a dead end street. Isolated nodes are not connected to arcs and represent point features. A polygon feature comprises a closed chain of arcs.

The topological approach represents a geographic environment as a graph. Both exact and approximate space decomposition techniques provide a collection K of convex cells representing the environment E. The topological approach builds a ***connectivity graph*** G which is associated with cells in K [KB91]. G is an undirected graph where nodes in G correspond to cells in K. Two nodes are connected by an edge in G *if and only if* their corresponding cells are adjacent in K. Several distinct topological mapping approaches have been proposed in [KB91, Mat92, SK97]. They differ in semantics, in particular about what makes a place, and what should be considered as a node or an edge in the graph.

Topological approaches provide more compact representations than those generated by exact and approximate space decomposition techniques. Since the environment is abstracted to a graph (Figure 2.12), movement errors that accumulate between graph nodes do not necessarily accumulate across a global frame of reference. A recent topological mapping approach employs Voronoi graphs to obtain the discrete representation of the environment from sensor data [CN01]. Huang and colleagues address distributed multi-robot topological mapping [CN01]. Pattern matching on graphs is exploited to evaluate different hypotheses about how to merge different partial topological maps, built by different robots, that po-

tentially overlap or match [HB05]. The topologic approach combined with an exact space decomposition technique provide an accurate and compact representation of the geographic environment. Such a representation only depends on the geographic environment's geometric complexity rather than on its scale or resolution. Moreover, this representation allows exploitation of efficient algorithms provided by graph theory.

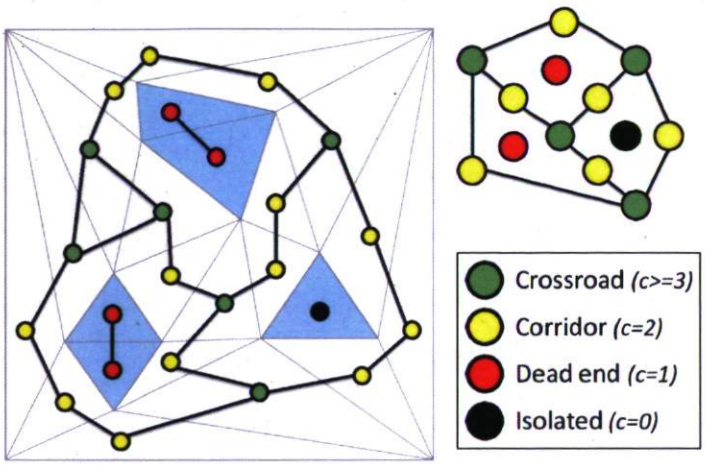


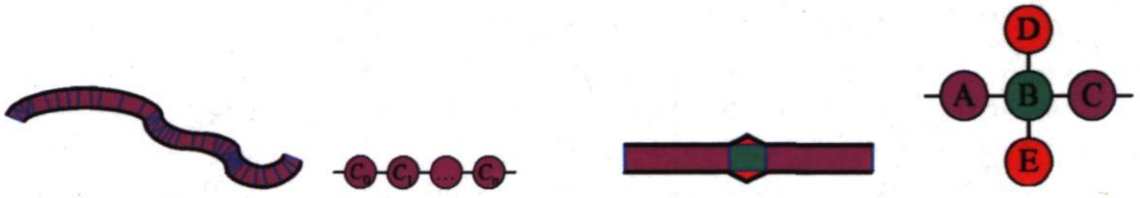
Figure 2.12: An Example of a Topologic Map.

The topologic data structure is often referred to as an *intelligent* data structure because spatial relationships between geographic features are easily derived from the graph structure. The topological model is used because it effectively models the relationship of spatial entities. Accordingly, it is well suited for operations such as contiguity and connectivity analyses. Contiguity involves the evaluation of feature adjacency, i.e. with features that touch one another, and proximity, i.e. features that are near one another. The primary advantage of the topological model is that spatial analysis can be done without using coordinate data. Many operations can be done mostly, if not entirely, by using the topological definition alone. This is a significant advantage over the raster data structure that requires the derivation of spatial relationships from the coordinate data before any analysis can be undertaken. Topologic approaches combined with exact space decomposition techniques provide an accurate and compact representation of the geographic environment which only depends on the its geometrical complexity rather than on its scale or resolution.

2.5 Abstraction of Virtual Environments

When dealing with large-scale and complex geographic environments, the spatial subdivision which can be either exact or approximate produces a large number of cells. The topologic approach allows representation of such a spatial subdivision using a graph structure and to

take advantage of efficient algorithms provided by the graph theory. However, the graph size may still remain large when dealing with geographic environments with dense geographic features. Moreover, geographic features with curved geometries (Figure 2.13) produce a large number of triangles since they are initially represented by a large number of segments. Therefore, there is a need to optimize the graph resulting from the space decomposition.



(a) Curved geometries. (b) Alignment anomalies.

Figure 2.13: Cells resulting from curved geometries (a) and alignment anomalies (b) [PDB06].

An *environment abstraction* is a process used to better organize the information obtained at the time of spatial subdivision of the geographic environment. The unification process is addressed principally in two ways: (1) a *pure topological* [LD04] unification which associates the subdivision cells according to their number of connexions; (2) a more *conceptual* unification which introduces a semantical definition of the environment, like with the *IHT-graph structure* [TD03].

Lamarche and Donikian proposed a topologic abstraction approach which assigns to each node of the graph resulting from the space decomposition a *topological qualification* according to the number of connected edges given by its arity [LD04]. This qualification enables the topological abstraction of the environment. For example, a sequence of corridor cells can be interpreted at a certain level of abstraction as a unique corridor. Thus, when planning, the geometric information related to the low level cells can be omitted and summarized in a higher abstract level. The topologic abstraction algorithm aims to generate an abstraction tree by merging interconnected cells while trying to preserve topological properties [LD04]. When merging several cells into a single one, the composition of cells is stored in a graph structure in order to generate the abstraction tree. The topologic abstraction proposed by Lamarche and Donikian reduces the size of the graph that represents the space subdivision [LD04]. The grouping process relies on the topological properties of the cells. The resulting graph contains fewer nodes and preserves the topologic and geometric characteristics of the geographic environment. However, the topological characteristics are not sufficient to abstract a virtual environment when dealing with a large-scale and complex environment involving areas with various qualifications (buildings, roads, parks, sidewalks, etc.).

Thomas and Donikian proposed an Informed Hierarchical Topologic (IHT) graph repre-

senting a part of the city of Renne (France) for human behavior animation purposes [TD03]. This graph is composed of three layers: (1) the *Basic Topological* layer which contains real urban objects modelled as simple spaces such as buildings and road sections; (2) the *Composite Space* layer which is composed of simple spaces or composite spaces of lesser importance; (3) the *Local Area* layer which is the highest level of the IHT-graph and which is composed of composite spaces. This hierarchical urban model allows abstraction of buildings into blocks and road-sections and crossings into roads. The abstraction process is done by the user which constrains its application to actual large-scale and complex geographic environments. Thomas's approach relies on a pre-defined decomposition of the virtual environment which is dedicated to urban environments. It does not take into account the topologic and the geometric characteristics of the environment. This approach is also application-dependent since it targets urban environments and can hardly be generalized to any large-scale and complex geographic environment.

There is still a need for an abstraction approach that optimizes the representation of the geographic environment while taking into account the geometric, topologic and semantic characteristics of the environment. This abstraction approach should rely on an exact space decomposition technique in order to preserve the geometric and topologic characteristics of the geographic environment rather than on a pre-difined space decomposition.

2.6 Informed Environments

Enabling autonomous agents to interact with their environment requires extending their knowledge about this environment. Agents not only need the geometric and topologic data which describe the environment but also semantic information that qualifies geographic features within the environment. Such semantic information enables agents' spatial reasoning and enriches the description of the virtual environment and objects located in it. Not much research has been done on semantic integration in the description of a virtual environment. The *Computer Animation* and *Behavioral Animation* research fields provide a few approaches to integrate the semantic information in order to assist agents interacting with their environments. Semantic information has been used for different purposes, including the simulation of inhabited cities [FBT99], computer animation [Kal01], and simulation of virtual humans [GRM09].

Farenc has first used the notion of *Informed Environments* [Far01a]. She defined informed environments as a database which represents urban environments with semantic information representing urban knowledge [FBT99]. An informed environment is thus characterized as a place where information (semantic and geometrical) is dense, and can be structured and

organized using rules [Far01a]. The notion of *urban knowledge* encloses urban structural information and objects usable according to a set of conventions, and more particularly as association between places (geometrical area) and semantic information [FBT99]. Building an informed environment as presented by Farenc consists of adding a semantic layer onto a core corresponding to a classical scene (a set of graphical objects) modeled using graphical software [Far01a]. The semantic layer associates objects with properties usable during simulation of urban life.

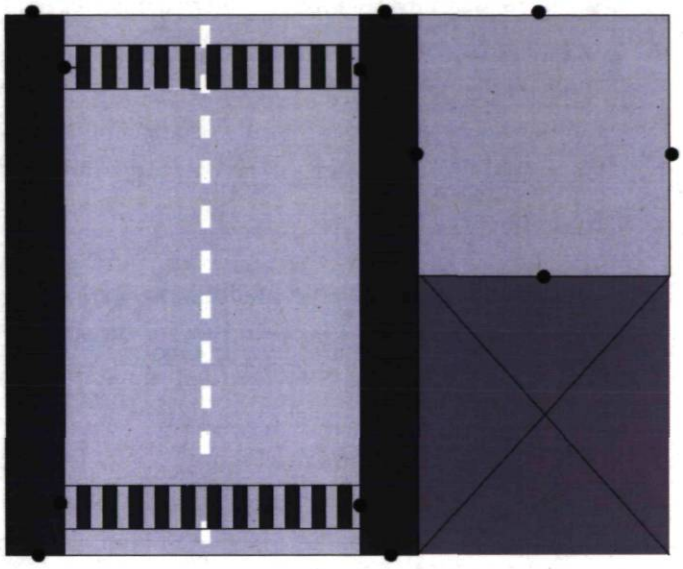


Figure 2.14: An example of an urban informed virtual environment [Far01a]. Notice how shapes are geometrically simple. Black circles refers to entry and exit points to areas used for path planning computations.

The information is organized in a hierarchical manner by space partitioning [FRMS+98]. For example the block of buildings is subdivided into buildings and streets. Streets are further subdivided into sidewalks and roads. Roads are divided by junctions and segments (Figure 2.14). Each segment in turn contains information about objects contained in it, such as crosswalks, bus stops, signs, benches etc. The hierarchically organized information is then used by agents (simulated pedestrians) navigating in the virtual city [FSK+99]. The inherent space partitioning enables fast queries about the objects and properties of the agent's surroundings. Another use of the information is to make decisions by situated agents, as for example to allow agents to walk only on sidewalks and crosswalks, but not in the middle of the road.

Embedding the information directly in the environment allows the support of agents' spatial reasoning capabilities. However, the preparation of the fully augmented geometric model is very time consuming and difficult due to the sheer amount of data. For example, a typical model of a city quarter as used by Farenc can contain several thousands of primitives of many

types (such as polygons modeling sidewalk pieces, benches, trees, bus stops, etc.). Moreover, Farenc built the urban environment using data provided by CAGD systems since the purpose of the simulation is computer animation. Such data refers to geometric objects which shapes are usually regular and simple. However, when building virtual geographic environments representing large-scale and complex geographic environments based on reliable GIS data, Farenc's approach can not be used since it is dedicated to exclusively represent urban environments. Indeed, the manual hierarchical space partitioning is not feasible when dealing with geometrically complex environments. An automated approach should instead be used in order to easily generate such a space partitioning. Moreover, the data structure of the urban environment's description as proposed by Farenc needs to be enhanced in order to manage a large amount of geometric and topologic data. Finally, the hierarchical structure should be built using the geographic environments's characteristics rather than being defined *a priori* as Farenc proposed.

Smart objects introduced by Kallmann and colleagues allow easy reusability of already defined objects located within the virtual environment [Kal01]. This property stems from the fact that the smart objects are self-contained and the animation control is decentralized, alleviating the need for complex updates of the agents in the simulation whenever a new object is added to the virtual environment (Figure 2.15). Unfortunately, the smart objects are strictly targeted towards animation - the semantic information contained within them is animation-oriented and geometric in nature. Hence, semantic information is not available for agents to reason about them.

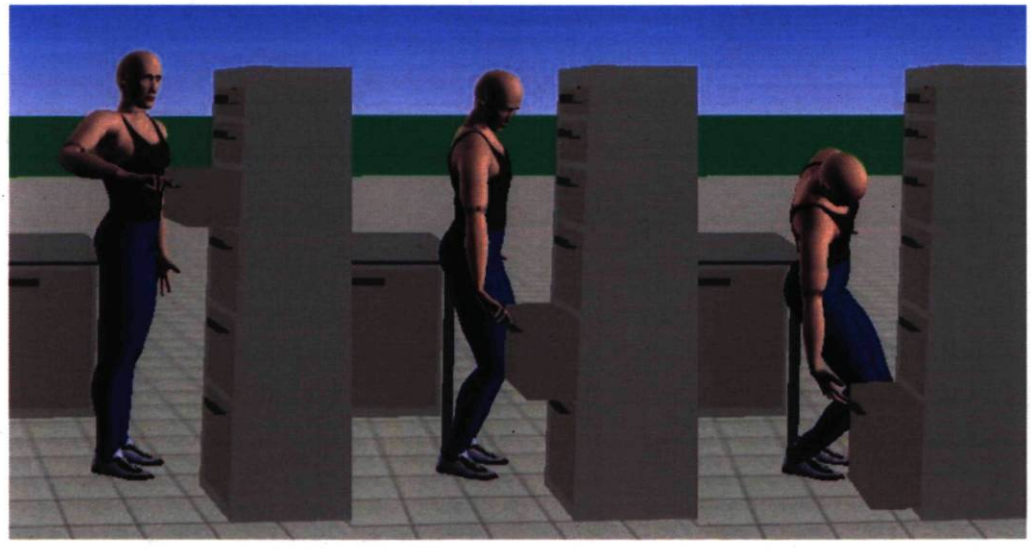


Figure 2.15: An agent representing a human operating a smart object - a drawer.

Despite the multiple designs and implementations of VGE frameworks and systems, the creation of geometrically-accurate and semantically-enriched geographic content is still an

open issue. Indeed, research has focused almost exclusively on the geometric and topologic characteristics of the virtual geographic environment. However, the meaning of the geographic features contained in the environment as well as the ways to interact with them have received less attention. The work done towards representation of semantic information in virtual environments has been mostly carried out at a geometric level [GRM09]. An enriched representation of virtual environments has been proposed by Gutiuerez for computer animation and user interface purposes [GA05]. This approach addresses the lack of flexibility of virtual objects within a virtual environment. It mainly refers to the need for adaptive entities, from the perspectives of the geometric representation and the user interface. Gutiuerez proposes a semantic model which aims to represent the meaning, and functionality of objects in a virtual scene [GA05]. However, since the purpose of Gutiuerez's approach is computer animations, the semantic information integration is located at the object description level rather than enriching the description the geographic environments.

Virtual environments are usually created as computer graphics applications, with minimal consideration given to the semantic information [GA05]. The typical data are a convex mesh representing some geometric shapes. There is a gap between geometry and semantic information in current virtual geographic environment models. Only a few works have attempted to integrate semantic information into a VGE's description [Yer09]. Moreover, semantic information has been used in an *ad hoc* way without any standard formalism. Indeed, semantic information integration into a VGE's description is by nature a knowledge representation problem. A suitable and standard knowledge representation formalism should be used to integrate semantic information in the VGE's description.

2.7 Conclusion

In this chapter, we provided a short survey on virtual geographic environments. First, we introduced GIS as a reliable source of standard geographic data. Geographic data are provided in either a vector or a raster format. However, when we need to **precisely** represent geographic environments, the vector model, even though complex to manipulate, is more appropriate since it preserves the geometric and topologic characteristics of the geographic features.

Second, we presented the two common methods for space decomposition, namely *exact* and *approximate* space decomposition techniques. The study of these methods revealed the existence of a tradeoff between accuracy and complexity. In contrast with the exact space decomposition method, the approximate space decomposition method lacks precision and depends on both the environment *scale* and *complexity*. Therefore, the exact space decomposi-

tion method it more suitable for accurate representation of large-scale environments. Among the presented exact space decomposition techniques, the constrained Delaunay triangulation method (Section 2.3.3) not only accurately represents the geographic environments but it also allows preservation of user-defined segments which are specified as constraints.

Since the major drawback of the exact space decomposition method is the complexity of its computation and manipulation, the topological approach may be of interest as it provides a mapping of cells produced by the exact space decomposition to a connectivity graph. The connectivity graph provides a graph-based representation of the geographic environment and allows to take advantage of the graph theory's algorithms. On the one hand, the connectivity graph encompasses topologic properties which facilitates spatial reasoning algorithms such as navigation and path planning, but on the other hand, it does not support detailed navigation due to its lack of metric information, such as absolute position. Nodes of the connectivity graph need to be extended with the geometric data that characterize cells produced by space decomposition techniques [ZFP94]. The enhanced topologic approach may then provide agents with the information they need to navigate, perceive, and plan a path in virtual geographic environments.

Another important fact to observe about the support of large-scale geographic environment is the hierarchical representation of the virtual environment. We have seen that hierarchical representations, which can optimize the representation of virtual environments, may be built according to a geometric grouping approach (Section 2.2.1) or result from an abstraction process (Section 2.5). Two kinds of abstractions have been proposed: *topologic* and *semantic* abstractions. On the one hand, the topologic abstraction approach which has been proposed in [LD04] needs to be extended in order to take into account the geometric characteristics and the semantic information that qualify the geographic environment. On the other hand, the semantic abstraction should take into account both the geometric and the topologic characteristics of the geographic environment rather than exclusively relying on a pre-defined and application-dependent space decomposition as proposed in [TD03].

Using semantic information in order to enrich the description of the virtual environment has been proposed by a few researchers [Yer09]. The notion of *informed virtual environments* has first been introduced by Farenc to refer to virtual scenes built using data provided by CAGD systems since the purpose was computer animations. Others used this idea in order to integrate semantic information within the description of objects [Kal01], agents' animations [GRM09] and user interfaces [GA05]. However, semantic information integration remained optional and located at the object description level rather than in the geographic environment. Moreover, semantic information has been used in an *ad hoc* manner without using any standard formalism. Indeed, semantic information integration in VGE's description is by nature a knowledge representation problem. A suitable and standard knowledge

representation formalism should be used to integrate semantic information in the VGE's description.

Finally, despite the multiple approaches of space decomposition techniques, the creation of geometrically accurate and semantically enhanced virtual geographic environments is still an issue. Research focuses on geometric and topologic characteristics of space decompositions, but important questions such as the information that qualifies the geographic features and the ways to represent it in the VGE have received less attention.

Chapter 3

Autonomous Agents in Virtual Geographic Environments

In this chapter, we focus on the notion of *agency*, *situatedness*, and *situated decision-making* and their role in multi-agent geo-simulations. An agent is an autonomous entity exhibiting qualities of perceiving its environment and acting on it through actuators. In the context of this work most agents will be considered as standalone software entities, intelligent and embodied, represented in the virtual world in the form of a virtual character. However, a few agents encountered in this thesis will be non-embodied, and serving in auxiliary roles, such as groups. These do not have direct visual representation in the virtual geographic environment, but they provide important services to the rest of the agents.

This chapter is organized as follows. First, Section 3.1 introduces situated multi-agent systems and focuses on the *Perception-Decision-Action* (PDA) loop. Next, Section 3.2 details the different means agents use to perceive the VGE. Section 3.3 presents some reactive agents' models which act in response to perception of the VGE. Section 3.4 highlights some cognitive agents' models which plan and decide before acting on the VGE. Section 3.5 focuses on the way agents act on their surrounding VGE in the pursuit of their objectives. Section 3.6 summarizes the work done on perception-decision-action loop with respect to a virtual geographic environments' characteristics.

3.1 Situated Agents and Multi-agent Systems

Situatedness is a property of agents adopted by most researchers in the domain of multi-agent systems. A well known example is Wooldridge and Jennings' definition of an agent in [WJ95]: "*an agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its objectives*". In this definition, *situatedness* expresses the spatial relationships between agents and their virtual environment in which they evolve and with which they interact. Actually, these relationships give the system meaning and drive the evolution of the MAS [WOO07]. Through its situatedness an agent is placed in a context that it is able to perceive and in which it can interact with other agents. Intelligence in a situated MAS originates from these interactions and the richness of the VGE description, rather than from capabilities of individual agents [WH03].

The term *agent* is widely used in artificial intelligence to refer to an entity based on perceptions and actions [WSH05]. A good survey of existing software agents, and a discussion of their properties, are provided in [WJ95], but the paper does not come to a final definition of what an agent is. Franklin defines an agent as "*a system situated within and a part of an environment, that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future*" [FG96]. This definition, after two minor modifications, accurately represents our vision of the notion of an autonomous situated agent: first, the environment is a *virtual* one and second, the agent acts on the environment *while taking into account its geometric, topologic, and semantic characteristics*. This brings us to the definition that we will use throughout this thesis: "*An autonomous situated agent evolves within and is a part of a virtual environment, that senses that environment and acts on it while taking into account its geometric, topologic, and semantic characteristics, over time, in pursuit of its own agenda*". This leads us to the representation of an agent as a *perception-decision-action* loop which can be seen in Figure 3.1. The agent perceives the environment through sensors, which provide information to the decision and reasoning module, which in turn uses the agent's effectors to interact with the environment and to modify it. The blocks represented in the figure will be discussed at length throughout this chapter.

In the following subsections, we provide a short survey on autonomous agent models. Indeed, different agent models have been proposed and doing an exhaustive presentation of all of them can be quite challenging. While the models we are going to present affect to different degrees the entire perception-decision-action loop, we will try to classify them according to which part of the loop they affect most.

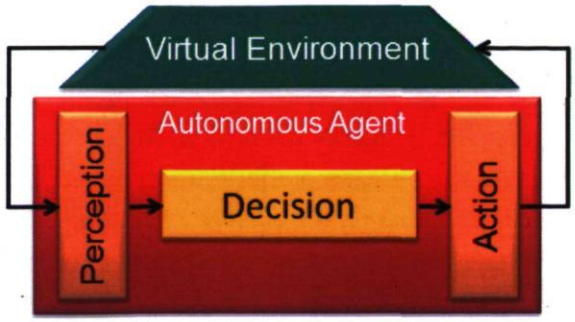
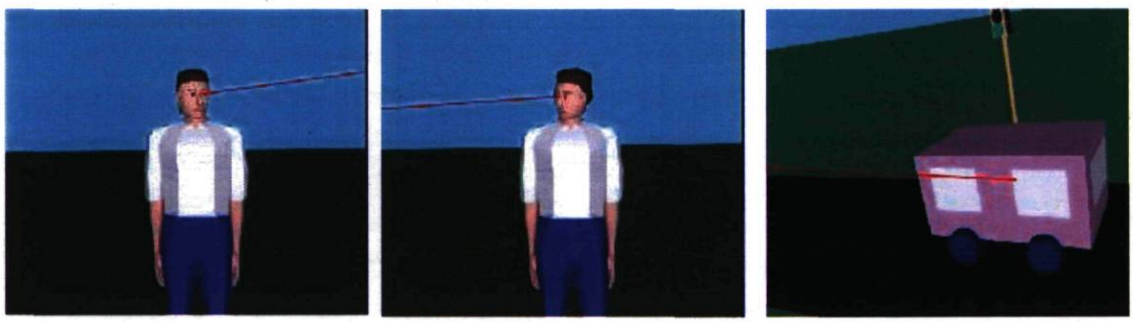


Figure 3.1: The perception-decision-action loop representing an autonomous situated agent [Mal97].

3.2 Perceiving the Environment

In order for an agent to perceive its environment, it has to be given a way to sense it. The most studied sense is vision, since it is the sense that allows agents to gather information from the environment (Figure 3.2). Some techniques take advantage of the fact that the agent is evolving in a virtual environment and allow it to perceive the environment in a *perfect* way that would have been impossible if it were evolving in the real world. This approach is

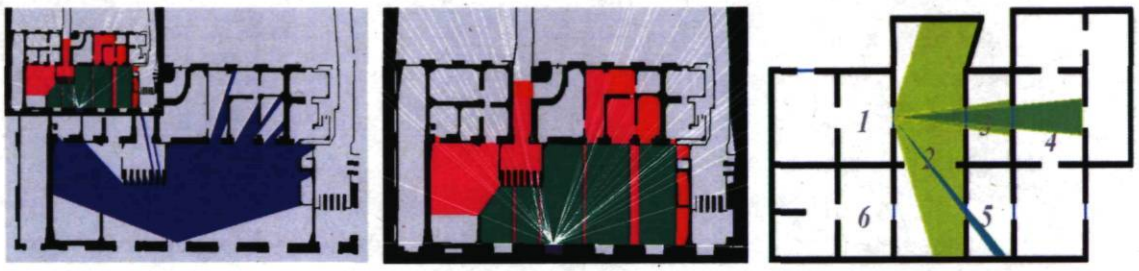


(a) (b) (c)

Figure 3.2: A virtual human monitoring a traffic light and avoiding vehicles [CKB99].

unrealistic since the agents become omniscient, knowing everything there is to know about the environment. To avoid this problem, filters can be applied in order to reduce the quantity (and even the quality) of the information the agent can access [BBT99]. Filters are usually based on perceptual behavior and examples of them have been used by Badler [BWB+96] and Bollini [Bol94] to provide a high level of abstraction of the environment to their agent. In addition to these filters, Paris *et al.* use multiple levels of abstraction allowing the agent to perceive the environment at different levels: from simple geometry occlusion to more topologic abstract notions such as *corridors* and *dead ends*, as can be seen in Figure 3.3 [PDB06].

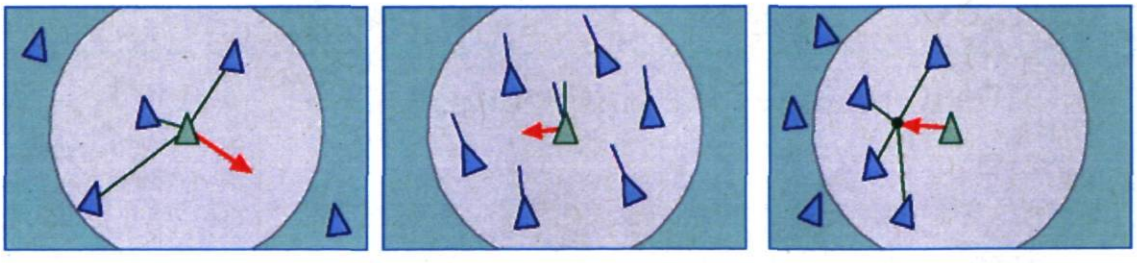
Paris *et al.* used a pre-computed Potential Visibility Set (PVS) for each pair of adjacent areas (Figure 3.3(c)). The topologic abstraction of Paris and colleagues is an improvement of the informed subdivision of the environment proposed by Lamarche [Lam03] (see Section 2.5 for more details on topologic abstraction of virtual environments). Besides environment perception, Lamarche's agents also perceive their neighboring agents through a dynamically constructed neighborhood graph [LD04]. The agents have direct access to the position and to the speed of their neighbors, and hence can adapt their trajectory accordingly in order to avoid collisions.



(a) (b) (c)

Figure 3.3: (a) Managing occlusions in the environment; (b) topologic abstraction of the environment; and (c) an example of PVS computation [PDB06].

In the flocks of birds and schools of fishes of Reynolds [Rey87], and the crowds of virtual humans of Musse [MT97], all agents have direct access to the position of their neighbors (Figure 3.4). Tu *et al.* use both a vision and a temperature sensor, allowing their agents to access the surrounding geometry as well as the physical state of the environment in which they evolve [TT94].



(a) (b) (c)

Figure 3.4: Rynolds's *Flocks of Boids* model. Agents use a direct access to the virtual environment to determine the position of their neighbours [Rey87].

Some techniques add semantic information to the database describing the environment. In this way, the agent not only has access to the geometry of the environment, but also

has access to additional information that qualifies what the agent is perceiving. Farenc *et al.* [FSK+99, FBT99] use this technique to describe an urban environment, allowing pedestrian agents to plan their paths through a virtual city (see Section 2.6 for more details on informed environments).

Thomas and colleagues used the urban environment proposed by Farenc with the purpose of simulating traffic in urban environments [TD00]. They included pedestrian navigation information such as sidewalks and pedestrian crossings. Axial and directional information is also included in the zones allowing the agent to know which direction to follow when navigating in an area. This process filters the attention of the agent to what directly concerns it. For example, an agent will not pay attention to cars if it is walking on a sidewalk, but only if it is crossing a street [GD00]. Similarly, two pedestrians will only try to avoid each other if they are navigating on the same lane.

To conclude, perception is a key element of the PDA loop. Most of the existing perception models use a direct access to the virtual environment description and only consider its geometric characteristics for path planning and navigation purposes. Topological abstraction techniques of virtual environments, which we introduced in Section 2.5, allow agents to perceive and interact with an abstracted view of the virtual environment. The perception of such an abstracted view requires less computation and memory use, which can increase the number of simulated agents within MAGS. In order to provide agents with an enriched description of their surrounding, some approaches used semantic information. However, the use of semantics was *ad hoc* and generally lacked standard structuring and formalization. Indeed, semantic information was accessible to agents through their perception process. However, agents do not have the capabilities to reason about this semantic information in order to make decisions that take into account the virtual environment's characteristics.

3.3 Reacting to the Environment

Once the perception data are collected from the environment, they need to be processed through the agent's decision module. Two types of decision mechanisms can occur: *reactive* and *cognitive*. This section focuses on reactive systems, and the next section focuses on cognitive systems. Reactive systems allow the agent to directly react to the stimuli it receives from the environment, without having to go through a complex and abstract decision process. *Stimulus Response Systems*, *Rule-Based Systems*, and *State-Machines* are examples of reactive decision mechanisms.

3.3.1 Stimulus-Response Systems

Stimulus-response systems are part of the first generation of behavioral models which use nodes of interconnected simple neurons. These nodes only respond by emitting a certain output signal depending solely on the input signal, much like the neurons in a human brain [Bro91]. That is why these networks are also referred to as neural networks. In such networks, input nodes are connected to the agent's sensors and propagate the perceived information to the output nodes which are connected to the effectors. Then, a mathematical function must be defined by using this network of nodes to directly correlate perception with action.

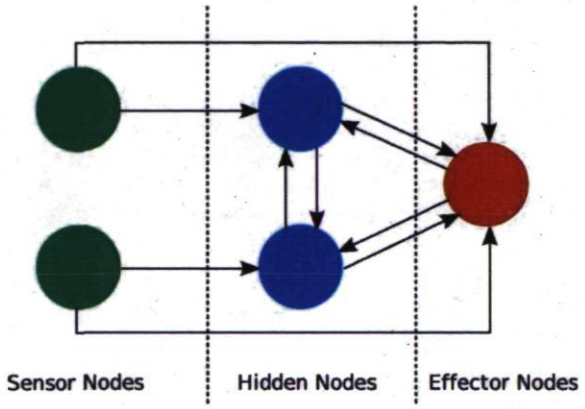


Figure 3.5: Example of a *Sensor Actuator Networks* [VDP93].

Van De Panne used *Sensor Actuator Networks* (SAN) to control the locomotion of virtual creatures [VDP93]. Figure 3.5 shows an example of such a SAN with a three leveled network: *sensor nodes*, *hidden nodes* and *effector nodes*. Each sensor node is connected to all the hidden and effector nodes in the network, whereas the effector nodes are only connected to the hidden nodes. A similar approach called *Sense Control Action* (SCA) loops has been proposed by Granieri and colleagues [GBR+95]. Three types of nodes are used to control the navigation of an autonomous agent: *perceptual* nodes that act as sensors, *control* nodes that act as attraction or repulsion of the agent towards certain parts of the environment, and *motor* nodes that generate the movements of the agent.

The reason neural networks are well adapted to correlate perception with action is that they are universal approximators, which means that they can approximate any continuous mathematical function. However, taking into account the characteristics of the geographic environments using mathematical functions is usually complex. Therefore, stimulus response systems and their variants are not sufficient to take into account the spatial characteristics while driving the agent's behaviours.

3.3.2 Rule-Based Systems

Rule based systems model the behavior of an agent as a set of rules. The choice of the behavior to adopt is thus defined by a pre-condition in the environment stating the context in which the corresponding behavior should be adopted. This approach supposes the existence of a set of rules covering all possible situations. For example, the reactive decision model of the *Soar*[1] architecture [LNR87], is built on the notion of operators. These operators can be a primitive action (move around, turn), an internal action (remember something) or more abstract goals to satisfy (pass through a door, pick up an object). These operators are afterwards broken down into simpler operators until primitive action operators are reached [Lai01]. These primitive actions are then represented using *if-then* type rules (Figure 3.6).

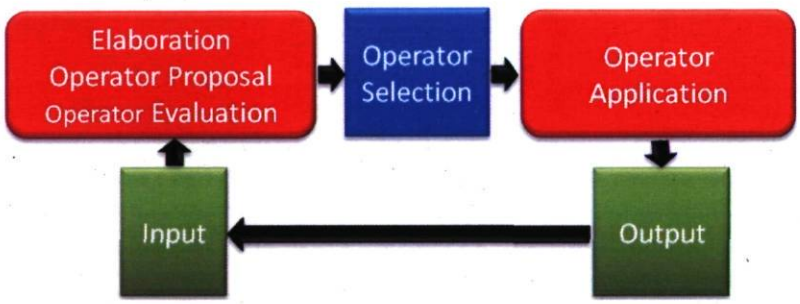


Figure 3.6: Soar's Processing Cycle.

As in Petworld [Cod89], an agent's behavior may be represented using a decision tree. Each leaf represents a simple behavior (attack, eat, etc.) and the nodes are like experts who choose between the possibilities offered by the subtrees attached to them [Blu97]. This process thus allows running multiple actions concurrently, and choosing the one best adapted to the situation.

Rule-based systems, represented by either *if-then* rules or decision trees, allow the description of behaviors and their concurrency in a rather intuitive way. Unlike stimulus-response systems they can be easily interpreted, and they can be extended and enriched just as easily. However, constantly adding rules to the system can become arduous since one has to pay special attention not to introduce conflicting rules which may render the system unstable or incoherent. Indeed, exhaustively describing the agent's reactive behaviours in front of the different geographic features that a virtual geographic environment may contain is not feasible. In addition, a geographic environment may be large-scale, complex, and densely populated with geographic features of various extents. As a result, a large number of rules

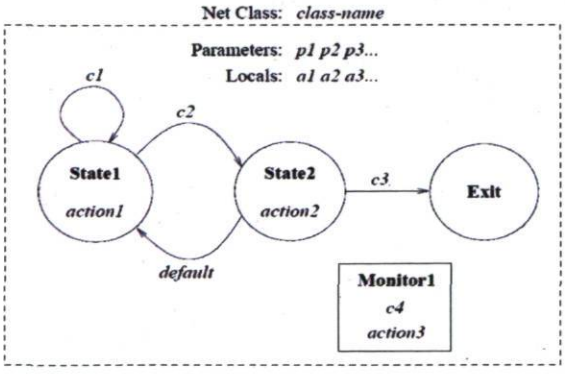
[1] Until recently, SOAR stood for *State, Operator And Result*, but over time, the community no longer regards Soar as an acronym: this is why it is no longer written in upper case

are required to specify the agent's behaviors while taking into account the geographic environment characteristics. Moreover, both stimulus-calculus and rule-based systems do not insure the temporal coherence between behaviors, since the behaviors are chosen to best fit the current situation, without taking into account all the precedent decisions.

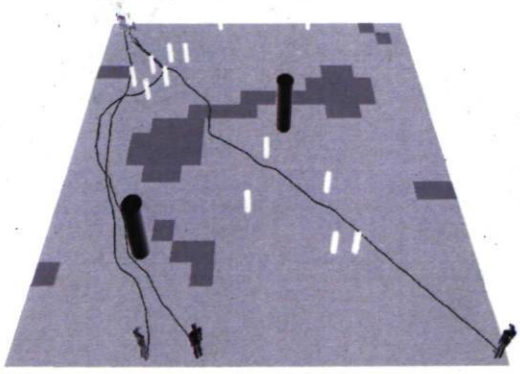
3.3.3 State Machines

To solve temporal coherence problems, behaviours can be modeled under the form of state machines. Indeed, describing a behavior consists of describing a series of tasks that need to be followed in succession, and finite state machines (FSM) correspond quite well to that description. Nodes within the FSM correspond to the task to be performed, and the transitions between the states represent the conditions linking these tasks together. This approach was used to simulate a tennis player within an environment where the ball obeys the laws of dynamics and the match is regulated by events sent by a referee [NT97]. While a single running FSM can easily be implemented, simulating complex behaviours in an agent necessitates running multiple FSMs, either in parallel or in succession and even a mix of both.

Parallel Transition Networks (PaT-Nets) [BWB+95] use multiple complex FSMs running concurrently. This provides more realism to the simulation since it allows the agent to execute several behaviours at once (Figure 3.7(a)). To insure coherence between the concurrent FSMs, semaphores may be used. Priorities can be assigned to the transitions between states to favor a path within the FSM and force certain behaviours (Figure 3.7(b)). Moreover, probabilistic weights can also be assigned to transitions to induce variety in the agent's behavior.



(a) A sample PaT-Net shown graphically.



(b) Attraction, avoidance, and terrain awareness.

Figure 3.7: Parallel Transition Networks (PTN) [BWB+95].

An extension to the parallel FSMs are the hierarchical and parallel FSMs which add the notion of hierarchy to that of concurrency. *Hierarchical Concurrent State Machines* (HCSM) [CKP95] and *Hierarchical Parallel Transition System* (HPTS) [Don01] are examples of such systems. Each FSM can have a certain number of children that run in parallel with their parent FSM [LD01, LD02]. Another parallel and hierarchical system is the *Hierarchical Parallel Transition System* (HPTS) [Don01]. It contains explicit time management and takes into account reaction times. HPTS has been extended into HPTS++ [LD01, LD02] which introduced both the notions of automatic behavior synchronization and the use of resources, but also the notion of FSM inheritance. Through inheritance, it is possible to extend the functionality of an already existing FSM or to factorize the common functionalities of several FSMs. Resources associated with states are managed by the FSM controller (either HPTS or HPTS++) which allows different FSMs to reserve and query these resources, thus avoiding interlock problems. An illustration of a hierarchical FSM can be seen in Figure 3.8, where an agent synchronizes three different concurrent behaviours: (1) a ***Reading*** behaviour using the *Eyes* resource to read a paper, and the Right Hand resource to turn the pages; (2) a ***Smoking*** behaviour using the *Left Hand* resource to hold the cigarette and the *Mouth* resource to smoke it; (3)a ***Drinking*** behaviour using the *Right Hand* resource to hold the coffee mug and the *Mouth* resource to drink the coffee.

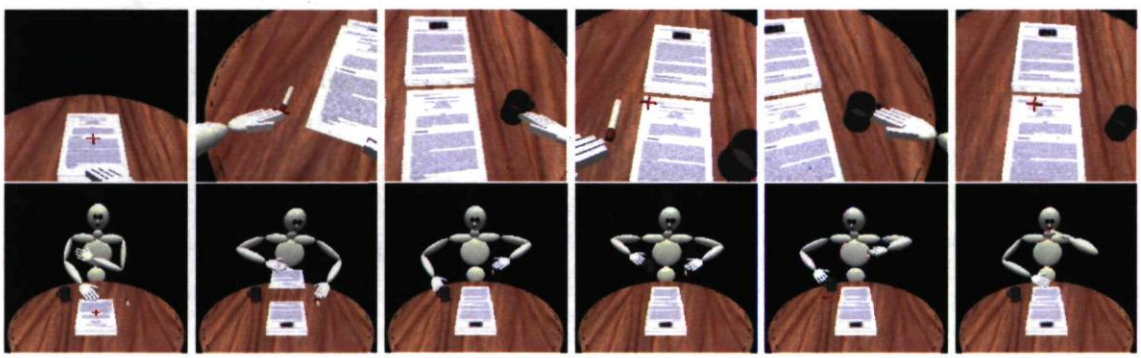


Figure 3.8: A *reading*, *smoking*, and *drinking* behavioural coordination using an hierarchical FSM [LD02].

The proposed state machines, either standard or hierarchical, usually use rules to govern transitions between states. Therefore, they are not fully sufficient to take into account complex characteristics of the the geographic environment while driving the agents' behaviours.

3.3.4 Synthesis

Reactive systems can be pretty efficient for describing simple behaviours. They can produce actions depending directly on the inputs received from perception. They have a high respon-

siveness and possess low calculation costs. However, these systems do not allow high level reasoning and planning. A reactive agent cannot anticipate the results that its actions will have on the world or decide which sequence of actions it needs to perform in order to reach a certain goal. Reactive systems need an abstract view of the geographic environment in order to support high level spatial reasoning and planning. The topological and the semantic abstraction techniques may be of interest to build such an abstracted view of the virtual environment. Indeed, spatial reasoning algorithms need a virtual geographic environment's description that is *accurate*, *enriched with semantic information* and *abstracted*. First, this description should be accurate in order to take into account the geometric and topological characteristics of the geographic environment which may be complex and densely populated with geographic features of various extents. Second, this description should also include a qualitative description of these geographic features in order to enable agents' spatial reasoning algorithms. Third, as geographic environments may be large-scale, this description should be abstracted in order to reduce and optimize spatial reasoning algorithms such as path planning and navigation. Hence, reactive systems need an accurate, enriched, and abstracted description of the geographic environments in order to support agent's spatial behaviours.

3.4 Planning Actions within the Environment

All the previously presented systems are reactive, which means that the behaviours implemented by them are simple reactions to the stimuli of the environment and do not take into account any internal abstract knowledge that the agent may have. In contrast to these reactive systems, we will now present a number of cognitive systems, or goal-oriented systems that allow the representation and manipulation of abstract knowledge.

3.4.1 Situation calculus

One of the earliest formalisms describing cognitive systems is situation calculus [MH69] which was proposed to reason about the world and its changes. Situation calculus allows the exploration of worlds, that is all the possible worlds that can result from the execution of one or several actions. The reasoning within situation calculus is based on four concepts: *situations*, *fluents*, *actions*, and *knowledge*. A *situation* describes the complete state of the world, at a given time, through facts that describe the properties of the environment. They can be used to deduce more facts that would occur in the future. A *fluent* describes a property of the world, that can change through time. It is a function which takes a situation as input and returns the state of the property. *Actions* allow to change a situation as long as a set of

properties, or pre-conditions, is verified. An *action* effects a change to a situation to create a new situation, thus modifying the description of the world. *Knowledge* can be gathered from the world through actions that describe perception for example, which allows reasoning on a rather high-level of abstraction. This allows the gathering of information necessary to achieve a specific goal. Using these concepts, an agent is able to create the possible future worlds, and to infer the correct actions to accomplish in order to satisfy a certain goal. Situation calculus has been implemented as part of several system such as GOLOG [LRL+97] (alGOL in LOGic) and CML [Fun98] (*Cognitive Modeling Language*).

The specification of a complete state of the world which involves a large-scale and complex geographic environment using facts and fluents is a complex task. Situation calculus are difficult to use to precisely describe complex and large-scale geographic environments.

3.4.2 The STanford Research Institute Planning System: STRIPS

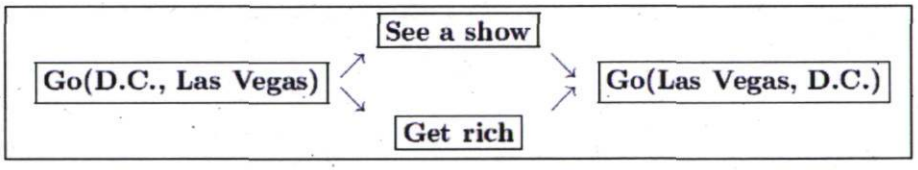
STRIPS is an action description language manipulating a subset of situation calculus [FN71]. The world is described as properties, which are simplified fluents represented by booleans: *true* meaning the property exists in the world; *false* meaning that it does not. Actions are also simplified and use sets of properties as preconditions. These properties have to be present in the world in order for the actions to be doable. Performing an action thus affects the existence or non-existence of properties in the world. For example, the *Heuristic Search Planning* (HSP) [BG01] and *GRAPHPLAN* engines [BF95] use a graph representation of the STRIPS formalism for planning. The nodes of the graph can either be a proposition, representing a fact present in the world, or an action allowing an agent to modify the world. As for the arcs, they can be *preconditions*, *adding operators*, or *suppression operators*. The graph itself is composed of alternated layers of facts and actions. Thus, an algorithm travels the graph, until it finds a layer where all goals are satisfied and none are mutually exclusive. STRIPS is less expressive than situation calculus but the planning algorithms using it are much more powerful. Indeed, the graph-based approach takes advantage of graph theory algorithms.

Obviously, a real world may be large-scale and complex and completely describing its characteristics using boolean properties is not feasible. Indeed, the status of geographic features located within a geographic environments may not be exclusively *true* or *false*. For example, a door may not be fully open neither fully closed, but may be in between. Such a situation is usually better qualified using semantics rather than booleans. STRIPS provides a relevant approach to describe actions but fails to address our need of describing actions which take into account large-scale and complex geographic environments.

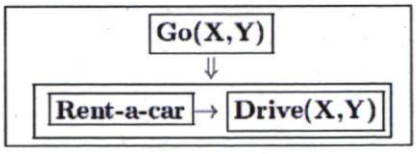
3.4.3 Hierarchical Task Networks

Instead of using atomic actions that modify the world and search for a linear series of actions to perform, the *Hierarchical Task Networks* (HTN) approach uses hierarchical planning based on three concepts: *tasks*, *methods*, and *actions* [EHN94]. The HTN formalism decomposes a task network into lower-level task networks, as can be seen in Figure 3.9, or to instantiate generic task networks. Consequently, planning consists in asking for a specific task to be done and in decomposing the task into its different methods.

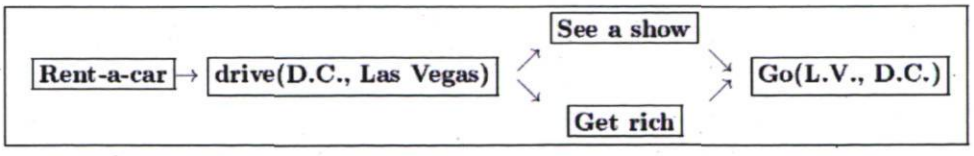
This approach relies on two hypotheses: tasks must be decomposable into subtasks, and tasks must be ordered in a way that their actions do not invalidate the preconditions of the following actions. This limits the HTN approach since the description of the methods must be done carefully. A particular attention must be given to concurrent tasks. Moreover, the HTN method does not provide tools or formalism to specify tasks that take into account the virtual environment characteristics. Tasks are pre-specified which limits the capabilities of agents to make decisions that take into account the environment characteristics rather than being able to map the world status to a hierarchical task-based planning structure or framework. Nevertheless, HTN allows the introduction of multiple resolutions of the same task and a fine control over the realization of the tasks [CCM02].



(a) A task network.



(b) A simplified method for going from X to Y.



(c) A decomposition of the task network in figure (a).

Figure 3.9: Decomposing a task network into other task networks [EHN95].

3.4.4 Action Selection Mechanisms

Action selection mechanisms were introduced to manage both aspects of the approaches exposed until now: reactivity to the properties offered by the environment, and selecting a series of actions to achieve a certain goal. To that effect, Maes introduced an action selection mechanism based on action description through non parameterized STRIPS operators [Mae90]. Actions are then represented in a graph. The nodes of the graph are the actions themselves and contain four types of information: the facts that need to be present in the world to allow the use of the action, the facts added to the world if the action is undertaken, the facts removed from the world if the action is undertaken, and the activation level of the action. The activation level serves as a threshold above which the action is considered doable. The selection mechanism computes energies associated with each action, and the action with the highest energy is considered to be the most adapted. The energy calculation spreads through the graph along three types of links: succession links, precedence links and conflict links between two nodes, or actions. Goals to be reached send positive energy through the graph [Tyr94]. The action with the highest energy after the algorithm terminates and whose preconditions are true is the one selected. The ASM has been extended by Bradly into *Planning Heuristically In Situated Hybrid Networks* (PHISH-Nets) [Rho97] and by Decugis and Ferber [DF98]. Action selection mechanisms allow the agent to take into account dynamic changes in the world that occur without the intervention of the agent itself. These changes can then be perceived by the agent and invalidate its current plan of action, or even present new and better opportunities to achieve its plan. Plans are thus constantly being rebuilt and adapted and they are never explicitly described.

3.4.5 Beliefs, Desires, Intentions

BDI reasoning is inspired from the practical reasoning of human beings and relies on three concepts: Beliefs, Desires, Intentions [Bra99]. The first implementation of a BDI system through a logic formalization is attributed to Rao [RG95]. The agent starts out with a set of plans which offer it a series of actions to undertake in order to achieve its desires. If the preconditions of a plan are verified, this plan becomes an intention and is executed, in steps, until another selection is required. The steps of a plan execution can be either adding a new desire, modifying the beliefs of the agent, or executing an atomic action. If multiple plans can be executed, a selection algorithm chooses the most adapted one. Plans are thus constantly being refined, accessing other plans in order to account for the changes in the environment dynamically.

The BDI approach can efficiently specify an agent's behaviours while taking into account

its beliefs about the world, its desires, and its goals. However, this approach does not provide spatial agents with the capability to take into account the characteristics of the geographic environment in which they evolve.

3.4.6 Synthesis

Cognitive systems allow agents to set long term goals and to formulate plans in order to achieve them. They offer high-level abstraction and knowledge representation which enables reasoning and planning. Cognitive systems are not easy to define and can become very tedious when the geographic environments and situations become more and more complex. Moreover, due to the complicated nature of reasoning, these systems require a high memory consumption and long calculation times. There is still a need for cognitive systems that allow agents to plan a suite of actions that take into account both the agent's and the geographic environment's characteristics in order to support the simulation of spatial behaviours.

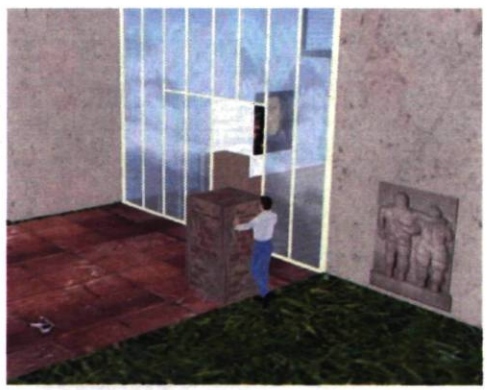
3.5 Acting on the Environment

After collecting information from the environment through perception and after processing this information according to knowledge, plans and desires, the agent now needs to act on its surrounding environment in order to achieve its goals. Opening a door, picking up an object, driving a car or simply walking in a park, all these actions affect the environment and consequently, the way the agent perceives the environment, bringing us back to the beginning of the loop. Not much research has been done on interaction between an agent and its environment. There are studies that examine fine animations of grasping movements [DLB96, WHBAJ06] but it is not exactly what concerns us in this section. What we would like to explore is how the agent interprets the information it gathers in order to perform complex interaction tasks within the environment. This type of approach mixes behavior and interaction information within the objects which advertise behavioral level services. The agent is constantly looking for interactions that will satisfy its energy, comfort, hunger, room, social connection, fun or mood. The objects themselves may have direct access to the actor's internal values, and modify them accordingly during the interaction process. So interaction with an object may be thought of as a direct consequence of an actor's internal variables, and the service advertised by a nearby object. *Smart objects* [KT98] and *Parameterized Action Representation* [BBB+98] are examples of such types of agents' interactions in virtual environments

3.5.1 SMART Objects

Smart Objects are a part of an object interaction architecture developed by Kallmann [KT98, Kal01]. All the information necessary to interact with the object is contained in the object itself. A Smart Object's description can hold a relatively complex action's synopsis, and will instruct the autonomous agent on the actions to do step by step. The first part of a smart object's description contains four main types of information: *intrinsic properties*, *interaction*, *behavior*, and *expected behavior*. The second part of the description uses finite state machines (FSM) to describe higher level behaviors for the object itself, as well as for the agent. These FSMs consist of a chaining of the simple actions an object can perform.

A recent addition to the *Smart Object* architecture has been proposed by Abaci which allows the running and mixing of simultaneous actions during interaction [AT05]. Information within the objects is given to an animation engine which can create and mix interaction animations in real time. Figure 3.10 shows two agents moving two crates (a small and a large one) to a virtual art gallery. The crates are modeled using the extended smart object approach and contain information regarding the geometry of the object (mesh, textures, etc.), position and orientation data for the hands of the agents during the animation, proper position where the agent has to be before the interaction and finally the planning data [Cig05]. The difference between the small and the large crates is that the large crate has been defined as a heavy object and therefore it needs two people to move it (Figure 3.10(b)).



(a) A single agent pushing the small crate.

(b) Two agents pushing the large crate.

Figure 3.10: Two agents moving two crates (smart objects) inside the lobby of a virtual art gallery [AT05].

3.5.2 Parameterized Action Representation

Parameterized Action Representation (PAR) is another interaction approach introduced by Badler and colleagues [BBB+98], where objects are integral parts in defining the action to be taken by an agent. Four main elements are needed to define a PAR: *Objects*, *Agent*, *Preparatory actions*, and *Action*. PAR offers a rather high-level description of the actions that can be performed and allows, through the preparatory actions, a certain kind of planning. However, the PAR itself is entirely pre-specified, and thus needs to be re-adapted offline if the targeted object is to be changed. Although no information is present within the environment itself, the actions to be taken directly depend on the objects they are applied to and on the agent doing the action. A good property of the system is that different behaviors and animations may arise when the same agent interacts with different objects with the same function, or when different agents interact with the same object.

3.5.3 Synthesis

In order to interact with objects located within the environment, the agent needs access to knowledge describing how the objects can be used. Unlike low-level data used for object manipulation, what is needed here is high-level abstract knowledge specific to the object itself. For example, the reasoning module of an agent may decide to move towards a final destination inside a building, but the actual process of taking the elevator or the stairs cannot be inferred. Further knowledge stored within the environment may instruct the agent on whether to take the elevator or to take the stairs in order to reach the final destination. This allows the agent to act on the environment, modifying its structure and thus modifying the perception that it has of the environment, which brings us back to the beginning of the perception-decision-action loop.

3.6 Conclusion

According to the definition we provided, autonomous agents' reasoning capabilities rely on the virtual geographic environment characteristics in which they evolve. In order to access and exploit such characteristics, autonomous agents use their sensors to collect data about their surroundings. Such a data collection process aims at emulating the human senses and visual perception remains the most evolved means of VGE perception.

Once the perception data are gathered from the environment, it has to be processed by the agent's decision module. Two types of decision mechanisms can occur: *reactive* and *cognitive*. On the one hand, reactive agents maintain no internal model of how to predict future states of the world. They choose actions by using the current environment state as an index into a table of actions, where the indexing function's purpose is to map known situations to appropriate actions. These types of agents are sufficient for limited and simple environments where every possible situation can be mapped to an action or set of actions. The reactive agent's major drawback is its lack of adaptability. This type of agent cannot generate an appropriate plan if the current environment's state was not considered *a priori*. In domains that cannot be completely mapped, using reactive agents is too restrictive. On the other hand, cognitive agents evolve autonomously in a virtual environment and are expected to show some responsive capabilities to changes occurring in the virtual environment. Moreover, this type of agents exhibits opportunistic, goal-oriented behavior and take the initiative when appropriate. The cognitive agent's major drawback is the complexity of its modeling.

Both reactive and cognitive agents evolve in and interact with their virtual environment. Providing reactive and cognitive agents with an accurate description of the virtual environment augmented with semantic information may be of interest when dealing with agent-based models taking place in large-scale and complex spatial environments.

Reacting and reasoning about the immediate surrounding area including other agents or objects situated within that area defines the *spatial behaviors*. Motion planning is widely considered as a spatial behavior among the fundamental interactions between agents and their environment. Using a geometrically-accurate and semantically-enhanced VGE description would enable agents to navigate and plan paths while taking into account obstacle locations as well as other geometric, topologic, and semantic characteristics. In the following chapter, we provide a survey on various motion planning techniques and spatial behavior models and we highlight the importance of the accuracy and semantic richness of the VGE's description for their support.

Chapter 4

Motion Planning and Spatial Behaviours in Virtual Geographic Environments

A motion planning problem occurs every time a situated agent has to reach a destination with a collision-free motion within a virtual environment containing obstacles. Humans unconsciously solve motion planning problems every time they are moving: walking in streets or buildings, grasping objects, transporting objects, scratching their head, driving their car, shaking hands, etc; an infinity of other examples could be enumerated here. Historically, the field of robotics first formulated the motion planning problem and raised the foundations of path planning and navigation algorithms. Indeed, this domain is motivated by the need for giving robots the required autonomy of motion in order for them to achieve tasks without colliding with obstacles in their environment which may be of various extents: inside buildings, streets, natural environments, air, water, and even on other planets. Looking from the computer animation perspective, it is clear that several major developments were achieved in the past decades [ST05, Bil07]. Several techniques dedicated to autonomous agent motion planning have been proposed [Kal01, Cig05, ST05, Aba06, Bad06, Bil07, Yer09].

A first level of motion autonomy for an agent has thus been already reached. It is already possible to have agents autonomously performing elementary actions with convincing motions. However, such motion planning techniques generally do not take into account the full environment where the action takes place. Terrain elevation as well as topologic and semantic characteristics are usually not considered. The risk of not taking into account such characteristics is serious: in most cases agents need to perform their motions in constrained environments where collisions with obstacles are not acceptable. A second level of autonomy is therefore still required for providing autonomous agents with spatial reasoning abilities, which has been the main goal of the robotics motion planning domain.

The rest of the chapter is organized as follows. Section 4.1 presents a short survey on path planning algorithms such as A^* and *Dijkstra*. Next, Section 4.2 introduces hierarchical path planning approaches to address motion planning in large-scale complex geographic environments. Section 4.3 highlights a few research work on the usage of the VGE's description to support spatial behaviors. Finally, Section 4.4 proposes a synthesis on path planning and spatial behaviors in VGE and concludes this chapter.

4.1 Standard Path Planning

The path planning issue, which consists of finding an obstacle-free path between two distinct positions located in a virtual environment, has been extensively studied. An excellent survey of this topic is available in [LaV06]. In order to understand why agents need path planning abilities to move within a virtual environment, let us consider the situation illustrated in Figure 4.1(a). The agent α is initially at the bottom of the map and wants to get to the top. There is nothing in the area it scans (shown in pink) to indicate that the unit should not move up, so it continues on its way. Near the top, α detects an obstacle and changes direction. It then finds its way around the "*U*" shaped obstacle, following the red path. In contrast, a pathfinder would have scanned a larger area (shown in light blue), but found a shorter path (shown in blue), never sending the agent α into the concave shaped obstacle. It is possible to extend a navigation algorithm to work around traps like the one shown in Figure 4.1(a) as shown in Figure 4.1(b).

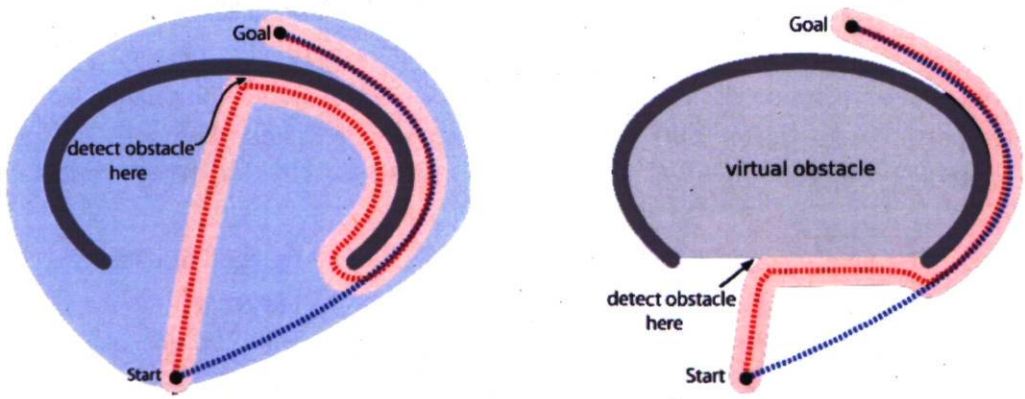


Figure 4.1: The need of path planning: (a) moving without pathfinding; (b) moving with pathfinding

Pathfinders let agent α look ahead and make plans rather than waiting until the last moment to discover there's a problem. There is a tradeoff between planning with pathfinders

and reacting with movement algorithms. Planning generally is slower but gives better results; movement is generally faster but can get agents stuck. If the environment is highly dynamic, planning ahead is less valuable. Therefore, agents should use pathfinding for regional-scale, slow changing obstacles and long paths; and navigation for local-scale, fast changing, and short paths. Choosing a pathfinding technique that performs well is important to the success of artificial intelligence in multi-agent geo-simulation, because pathfinding is a fundamental building-block for the field of Artificial Intelligence. There are a wide variety of pathfinding techniques, and one technique may excel under certain circumstances, but do badly under others. The success of a given approach depends on the requirements and assumptions of the simulation.

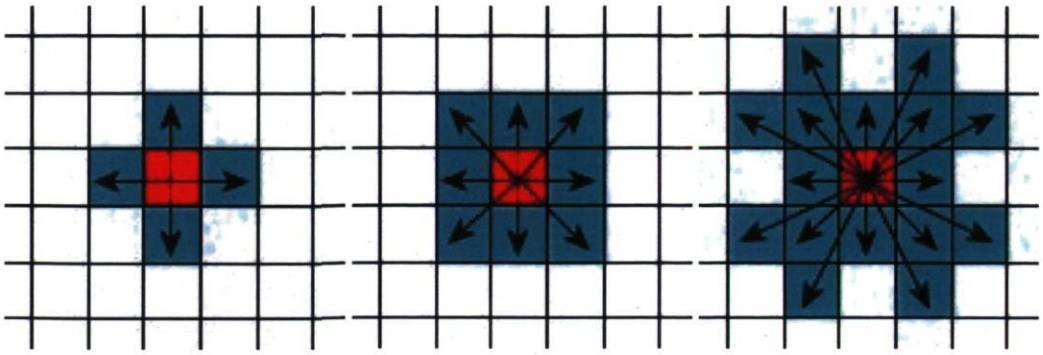
In the following sections, we first introduce the concept of *adjacency* between nodes representing the VGE. Next, we present the two well established path planning algorithms: *Dijkstra* and A^*. We also introduce the concept of *heuristic* which is a fundamental part of the A^* algorithm. Finally, we provide a short synthesis on standard path planning techniques.

4.1.1 Adjacencies

To find a path from a starting node to a goal node, we must define a way in which successor nodes can be selected, i.e. *Where can we move to from a given location*? In the real world, a person can take a step in any direction he/she pleases but in virtual environments we are more restricted in our choices. In grid-based VGE, there are two common approaches; *4-adjacency* (see Figure 4.2(a)) and *8-adjacency* (see Figure 4.2(b)). *4-adjacency* restricts the movement to the four main orthogonal directions: north, south, west and east. *8-adjacency* allows more freedom in movement as it, in addition to the directions of *4-adjacency*, also allows movement in the northeast, northwest, southwest and southeast directions. In addition to 4 and 8 adjacency, we can also use 16-adjacency (see Figure 4.2(c)) which allows even greater freedom of movement as we can now move north-north-east, northeast- east, etc. Using *16-adjacency*, we can also achieve a smoother looking road by reducing the sharpness of the turns. To reduce sharp turns even further, we can penalize sharp turns. For example, it is cheaper to turn 45 degrees rather than 90 degrees.

4.1.2 Dijkstra Algorithm

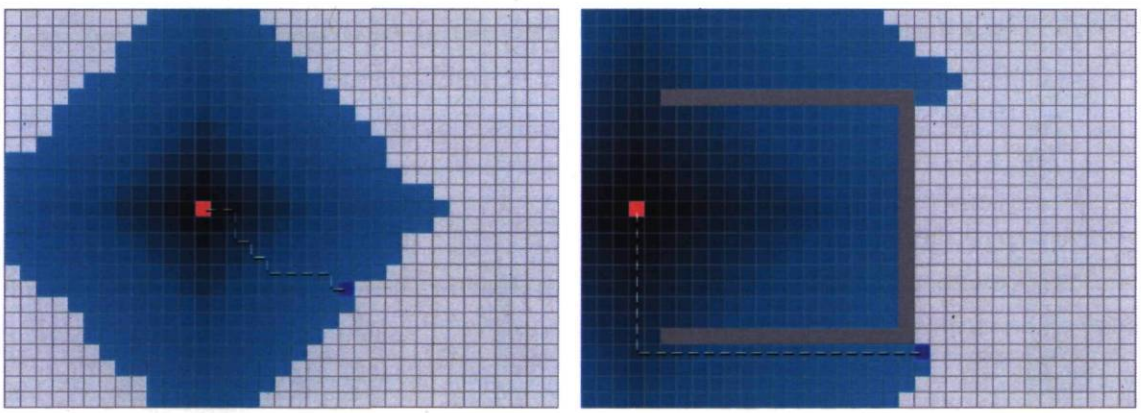
Dijkstra's algorithm was conceived by Dutch computer scientist Edsger Dijkstra in 1959 [Dij59]. This algorithm works by visiting vertices in the graph starting with the object's starting point. It then repeatedly examines the closest not-yet-examined vertex, adding its vertices to the set



(a) 4-adjacency. (b) 8-adjacency (c) 16-adjacency

Figure 4.2: Adjacent successor nodes in grid-based VGE.

of vertices to be examined. It expands outwards from the starting point until it reaches the goal. Dijkstra's algorithm is guaranteed to find a shortest[1] path from the starting point to the goal, as long as none of the edges have a negative cost. In Figure 4.3, the pink square is the starting point, the blue square is the goal, and the teal teal (blue-green) areas show which areas Dijkstra's algorithm scanned. The lightest teal areas are those farthest from the starting point, and thus form the *frontier* of exploration:



(a) Dijkstra Algorithm exploring cells to find a path in an obstacle-free environment

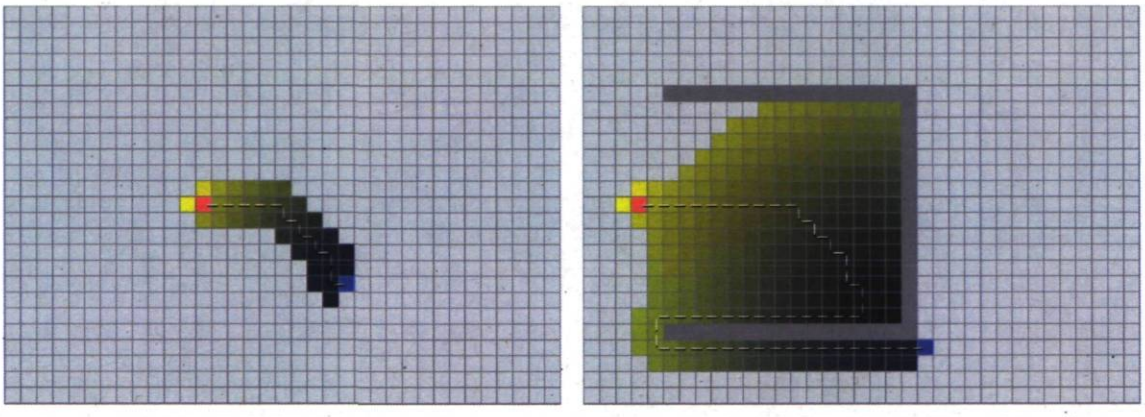
(b) Dijkstra Algorithm explores more cells to find a path in a virtual environment with obstacles.

Figure 4.3: Cells explored by the Dijkstra Algorithm to find a path to the destination node. The pink cell represents the source node; the dark blue cell represents the destination node. The gradient of blue corresponds to the increase of distance, the lightest being the farthest.

[1] we explicitly note "*a shortest path*" because there are often multiple equivalently-short paths.

4.1.3 Best First Search Algorithm

The *Best-First-Search* (BFS) algorithm works in a similar way as Dijkstra's algorithm, except that it has some estimate (called a heuristic) of how far from the goal any vertex is [Pea84]. Instead of selecting the vertex closest to the starting point, it selects the vertex closest to the goal. BFS is not guaranteed to find a shortest path. However, it runs much quicker than Dijkstra's algorithm because it uses the heuristic function to guide its way towards the goal very quickly. For example, if the goal is to the south of the starting position, BFS will tend to focus on paths that lead southwards.



(a) Best-First-Search (BFS) exploring cells to find a path in an obstacle-free environment.

(b) BFS explore less cells to find a path in a virtual environment with obstacles.

Figure 4.4: BFS explores less cells than Dijkstra Algorithm to find a path. However, the path computed by the Dijkstra Algorithm is shorter.

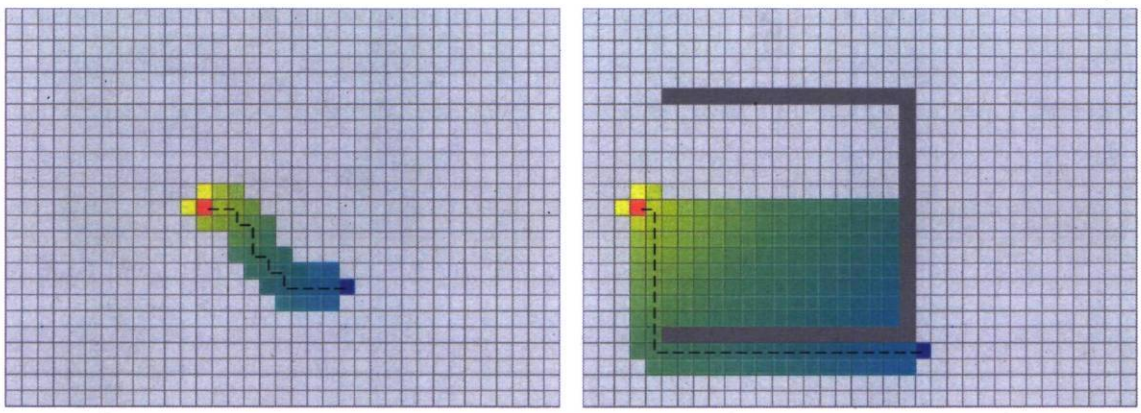
In Figure 4.4, yellow represents those nodes with a high heuristic value (high cost to get to the goal) and black represents nodes with a low heuristic value (low cost to get to the goal). a comparison between Figure 4.3(a) and Figure 4.4(a) shows that BFS can find paths very quickly compared to Dijkstra's algorithm. These figures illustrate the simplest case where the map has no obstacle, and the shortest path really is a straight line. Let us now consider an environment with obstacles. Dijkstra's algorithm works harder but it is guaranteed to find a shortest path Figure 4.3(b). BFS on the other hand does less work but its path is clearly not as good as the one found by Dijkstra algorithm Figure 4.4(b). The trouble is that BFS is greedy and tries to move towards the goal even if it is not the right path. Since it only considers the cost to get to the goal and ignores the cost of the path so far, it keeps going even if the path it is on has become really long.

It would be promising to combine the best of both approaches (Dijkstra and BFS) and this leads us to the A^* algorithm.

4.1.4 A* Algorithm

A^* was developed in 1968 to combine heuristic approaches like BFS and formal approaches like Dijsktra's algorithm [HNR72, Nil82]. It is a little unusual in that heuristic approaches like BFS usually gives an approximate way to solve problems without guaranteeing the best answer. However, A^* is built on top of the heuristic, and although the heuristic itself does not give a guarantee, A* can guarantee a shortest path. A^* is the most popular choice for pathfinding, because it is fairly flexible and can be used in a wide range of contexts.

A^* is like other graph-searching algorithms in that it can potentially search a huge area of the map. First, it is like Dijkstra's algorithm in that it can be used to find a shortest path (Figure 4.5(a)). Second, it is like BFS in that it can use a heuristic to guide itself (Figure 4.5(b)). Finally, in the simple case, A^* is as fast as BFS.



(a) A^*'s algorithm is able to search for the shortest path in a large-scale environment.

(b) In environments with obstacles, A^*'s algorithm finds a path as good as what Dijkstra's algorithm found in Figure 4.4(a).

Figure 4.5: The A^*'s algorithm search for the shortest path. The pink cell represents the source node; the dark blue cell represents the destination node. The gradient of yellow to blue corresponds to the total cost of path distances (sum of the actual and predicted lengths).

The secret to its success is that it combines the pieces of information that Djkstra's algorithm uses (favoring vertices that are close to the starting point) and information that BFS uses (favoring vertices that are close to the goal). In the standard terminology used when talking about A^*, $g(n)$ represents the cost of the path from the starting point to any vertex n, and $h(n)$ represents the heuristically estimated cost from vertex n to the goal. In the above diagrams illustrated in Figure 4.5, the yellow (h) represents vertices far from the goal and teal (g) represents vertices far from the starting point. A* balances the two as it moves from the starting point to the goal. Each time through the main loop, it examines the vertex n that

has the lowest $f(n) = g(n) + h(n)$.

There are variants of A^* that take into account changes occurring within the VGE and adjust the initial computed path. *Dynamic A** and *Lifelong Planning A** are two examples of the A^*'algorithm variants.

4.1.5 Dynamic A* and Lifelong Planning A*

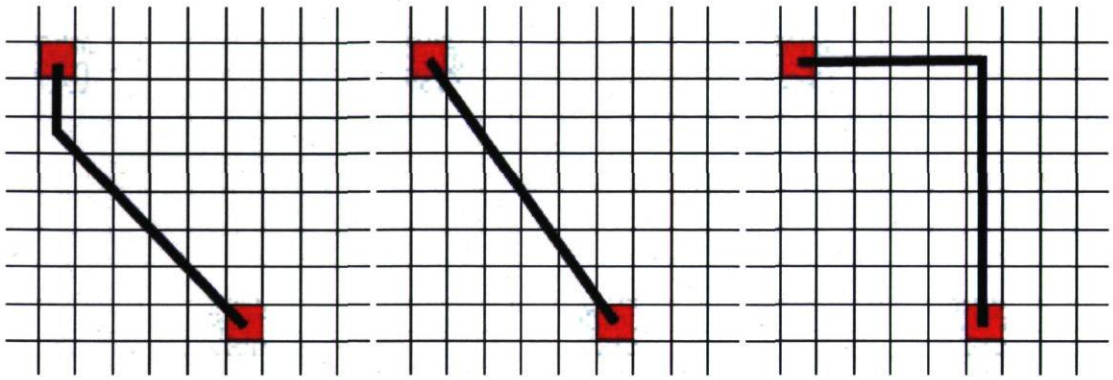
The Dynamic A^* algorithm (also called D^*) solves the path planning problem where a moving agent has to navigate to given goal coordinates in unknown terrain [Ste94]. It makes assumptions about the unknown part of the terrain, for example: that it contains no obstacles, and finds a shortest path from its current coordinates to the goal coordinates under these assumptions. It then follows the path. When it observes new map information (such as previously unknown obstacles), it adds the information to its map and, if necessary, replans a shortest path from its current coordinates to the given goal coordinates. It repeats the process until it reaches the goal coordinates or determines that the goal coordinates cannot be reached. Thus, replanning needs to be fast. Incremental (heuristic) search algorithms speed up searches for sequences of such similar search problems by using experience with the previous search problems to speed up the search for the current one. If the goal coordinates do not change, then all three search algorithms are more efficient than repeated A^* searches.

The *Lifelong Planning A** (also called *LPA**) is intended to be used when the costs of paths are changing [KLF04]. With A^*, the path may be invalidated by changes to the map. *LPA** can re-use previous A^* computations to compute a new path. However, both D^* and *LPA** algorithms require a lot of memory space because they need to save various internal information (data sets, path tree, g values, etc.), in order to handle dynamics (unknown terrain and cost changes) in VGE. When dealing with an environment which is populated with a large number of moving agents, D^* and *LPA** may be considered to manage the agents' path planning capabilities.

4.1.6 Heuristics

As we mentioned already, the heuristic part (the $h(n)$ component) used by the $A*$ algorithm is what differentiates it from *Dijkstra*'s algorithm by guiding the search towards the goal node. If the heuristic function is admissible (meaning it never overestimates the minimum cost to the goal) then $A*$ is also, like *Dijkstra*, guaranteed to find the shortest path. It is also of great advantage to use a heuristic that underestimates the minimum cost as little as

possible, as this will result in examining fewer nodes. An ideal heuristic will always return the actual minimum cost possible to reach the goal. In grid-based environments, the *diagonal distance heuristic* is such a heuristic using *8-adjacency* (see Figure 4.6(a)) and the *Manhattan distance heuristic* is one using *4-adjacency* (see Figure 4.6(c)). A third heuristic which is also commonly used is the *Euclidean* distance heuristic (Figure 4.6(c)).



(a) Diagonal Distance. (b) Euclidean Distance. (c) Manhattan Distance.

Figure 4.6: The computation of a distance between two points.

4.1.7 Discussion on A*'s Use of Heuristics

Using the heuristics concept, we have an interesting situation in that we can decide what we want to get out of A*. At exactly the right point, we can obtain shortest paths really quickly. In a multi-agent geo-simulation involving a large number of moving agents, this property of A^* can be very useful. For example, we may find that in some situations, we would prefer to have a "*good*" path rather than a "*perfect*" path. To shift the balance between $g(n)$ and $h(n)$, we can modify either one. A^*'s ability to vary its behavior based on the heuristic and cost functions can be very useful in multi-agent geo-simulations involving a large number of moving agents. The tradeoff between *speed* and *accuracy* can be exploited to make such geo-simulations faster. Usually, we do not really need the best path between two points; we just need an approximation of such a path.

Suppose our MAGS involves two types of terrain; *flat* and *mountain*, and the movement costs are 1 for flat land and 3 for mountains. A^* searches three times as far along flat land as it does along mountainous land. This is because it is possible that there is a path along flat terrain that goes around the mountains. We can speed up A^*'s search by using 1.5 as the heuristic distance between two map spaces. A^* will then compare 3 to 1.5, and it will not look as bad as comparing 3 to 1. Alternatively, we can speed up A^*'s search by decreasing the heuristic value it searches for paths around mountains. All that must be done is just specifying

to A^* that the movement cost on mountains is 2 instead of 3. Now A^* will search only twice as far along the flat terrain as along mountainous terrain. Either approach gives up ideal paths in order to to get something sufficient faster.

The choice between speed and accuracy does not have to be static. We can choose dynamically based on the CPU speed, the fraction of time going into path finding, the number of cells composing the VGE graph, or units on the map, the importance of the cell or unit, the size of the group, the difficulty level, or any other factor.

4.1.8 Synthesis

As we develop virtual environments that are increasingly interesting, larger in scale and more complex, the ability of agents to find their way around the terrain plays a more important role in their spatial behaviour. Path planning is the art of deciding which route to take, based on and expressed in terms of the current internal representation of the terrain. We presented a short survey of path planning algorithms including *Dijkstra*, *Best-First-Search*, *A**, and *Dynamic A** and *Lifelong Planning A∗*. However, when dealing with large-scale geographic environments, new approaches are required in order to compute paths that take into account the environments's characteristics. These approaches should take into account geometric criteria such as terrain elevation or direction changes when computing paths. They also should take into account characteristics such as path width, topologic criteria such as dead ends. Moreover, these approaches should take into account the qualification of areas crossed by a computed path (i.e. staying on the road for an agent which represents a car, or staying on the sidewalk for an agent representing a pedestrian). When dealing with large-scale environments, these approaches should compute paths with respect to low time and memory use. Hierarchical search is acknowledged as an effective approach to reduce the complexity of path planning in large-scale environments. In the following section, we present some work related to hierarchical path planning algorithms.

4.2 Hierarchical Path Planning

The computational effort required to find a path, using a search algorithm such as A^* [Nil82] or Dijkstra [LRDG90], increases with the size of the search space [BMS04]. As a consequence, path planning on large-scale geographic environments can result in serious performance bottlenecks. However, representing the virtual environment using the hierarchical approach allows a reduction in the size of the search space as well as the problem complexity in

path planning [HB08]. Two recent hierarchical triangulation-based path planning approaches have been proposed by Demyen and collegues [DB06], namely *Triangulation A** and *Triangulation Reduction A** . *TA** makes use of the *Delaunay Triangulation* (DT) technique to build a polygonal representation of the environment. 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 in order to maximise triangle size.

A possible approach to pathfinding is running the A^* algorithm [HNR07] on a tile-based representation of a map representing the environment. Although A^* generates optimal paths, it is computationally expensive, especially for large state spaces. One way to speed up pathfinding is to construct spatial abstractions which retain the topological structure of planar map graphs but contain fewer nodes. A^* often runs considerably faster on such abstractions of the map, while the quality of the generated paths remains good. Recently developed algorithms based on this idea are *Path-Refinement A** (*PRA** , [SB05]), which abstracts cliques to single nodes, and *Triangulation-Reduction A** (*TRA** , [DB06]), which builds a triangulation of the map and builds a compact topological description of the map by collapsing corridors and tree components.

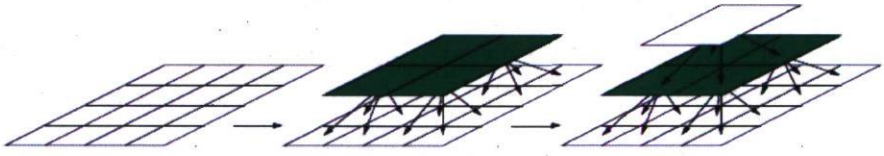


Figure 4.7: Abstracting tiles into a hierarchy [BMS04].

An earlier hierarchical pathfinding algorithm is *HPA** [BMS04]. This method builds abstractions of the search space by segmenting the map into sectors. It assigns entrances along the borders between the sectors, which become nodes in the abstract graph. The example in Figure 4.7 shows a small map. Edges are added between corresponding entrance nodes in adjacent sectors as well as between entrance nodes within the same sector if there is a path between them which lies entirely within the cluster. The weight for an edge is set to be the cost of the shortest path in a lower level of the abstraction. The path corresponding to the edge weight can either be cached or recomputed when it is needed.

It is possible to build more than one level of abstraction for *HPA** as illustrated in Figure 4.8 and Figure 4.9. This is done incrementally, by combining multiple sectors at level $n-1$ into a single sector at level n. All entrance nodes on level $n-1$ which lie along the borders of sectors on level n will become entrance nodes in level n. To perform a pathfinding operation, *HPA** first inserts the start and goal nodes into the abstract graphs. Next, an A^* search is done on some abstract level and the resulting path is refined level by level by

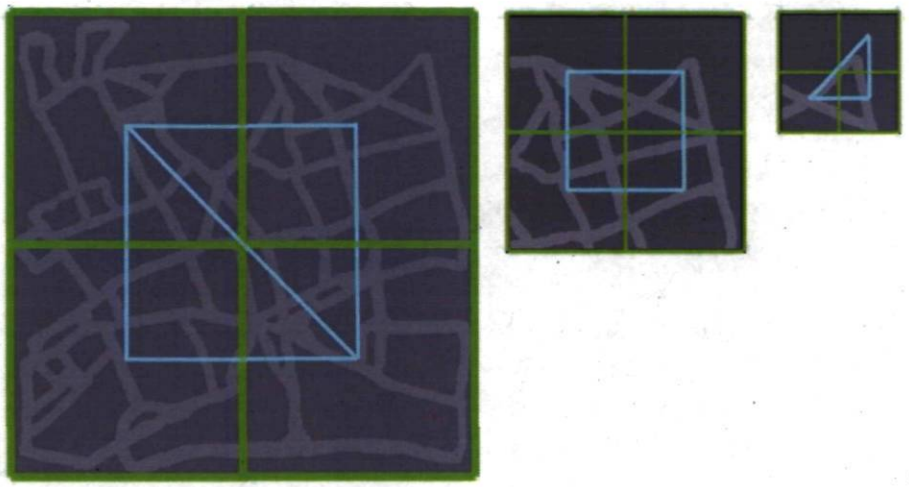
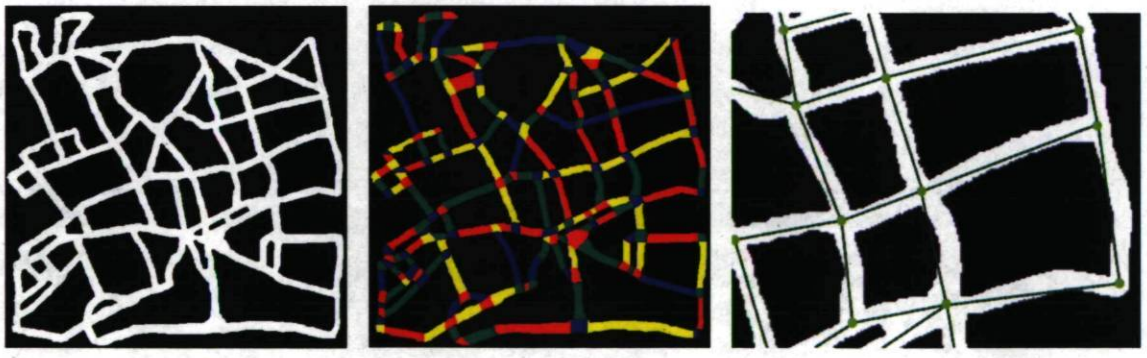


Figure 4.8: Illustration of different levels of hierarchy of a quad-tree applied on an environment as shown in dark green. On the left hand side is shown the highest level, while the subsequent subdivision of the north-east corner is shown on the right hand side. In light blue is shown the new graph structure built from the quad-tree, with cells linked only if there is at least one way to access directly one to another [WL08].

replacing each edge by a lower-level path. This is done until the map level is reached. The paths corresponding to each edge are either retrieved from the cache or recomputed. When the path at the map level is found, smoothing can be done to shorten the path.



(a) . (b) (c)

Figure 4.9: (a) Example of road map extracted from a city street map; (b) zones shown in different colors encoding the subdivided segments from the road map shown in (a); (c) an example of a graph shown in a zoomed in area of (a) [WL08].

To conclude, a hierarchical path planner involves several levels of reasoning. First, it requires an abstract model of the virtual geographic environment, so it can reason at both low levels (i.e. cell level) as well as high levels (groups of cells, regions, zones, etc.). It also involves planning and executing partial plans or paths through the VGE. When dealing with large-scale geographic environments, it would be unreasonable to consider planning every

cell change for the entire path before setting out. Yet this is exactly how traditional search algorithms have approached the task, building a complete plan before starting.

4.3 Spatial Behaviours

In any location, and particularly in a city, virtual humans need to be aware of their environment in order for them to show an intelligent behavior, i.e., act with accordance to their surroundings and react to sudden events that happen close by. Various methods have been studied on how to control crowd behavior by adding semantic data to their virtual environment. Farenc *et al.* presented a method for creating an informed environment, i.e., an environment able to provide pedestrians with information on their location, the position of various objects, the animations they should play, etc. [FBT99, Far01a] (see Section 2.6). Based on its geometry, an environment is hierarchically segmented into entities, such as quarters, blocks, parcels, or junctions. Musse *et al.* classified the environment information into two different types [MBCT98]: (1) *goals* that represent points through which the virtual humans should pass, or where the virtual humans should stop; and (2) *obstacle positions* which indicate where obstacles are situated and allow the pedestrians to avoid collisions with the environment. Farenc introduced the semantic information in the virtual environment description but did not give agents the required tools and mechanisms to access such information in order to reason about it and make decisions while taking into account the virtual environment's characteristics. Moreover, the virtual environment proposed by Farenc is application-specific since it is used for computer animation purposes, which reduces the chances of its generic use in MAGS for the simulation of spatial behaviours.

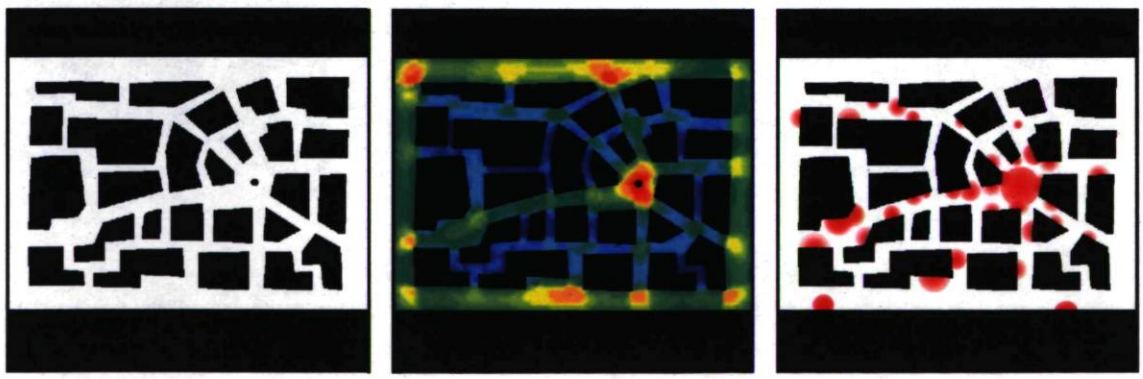


Figure 4.10: Collection of maps used to plan individual pedestrian actions: (left) An example of a collision map. The regions where agents can move are encoded in white and inaccessible regions in black. (center and right) Examples of behaviour maps: Visibility map and Attraction map [TLCC01].

Tecchia *et al.* [TC00, TLCC01, TLC02] developed a segmentation of the space into a 2D grid, composed of 4 different layers. Each layer has a specific task: one takes care of inter-collision detection, another detects collisions with the environment, the third takes care of the behavior, and the last layer manages the callbacks for complex behaviors. The behavior layer, in particular, allows association of a behavior to a cell. For instance, a cell can contain the information: *turn left*, or *wait*. We show in Figure 4.10 examples of such maps. Sung *et al.* [SGC04] developed a painting interface that allows a designer to draw the regions where special events should occur. This interface is based on a layered structure, where each layer contains the regions of a particular situation, saved as a bitmap file, as illustrated in Figure 4.11. Based on these regions, a scalable behavioral model was used to have characters react according to a probabilistic action-selection mechanism. Tecchia's and Sung's approaches suffer from the drawbacks of the grid-based approach and more particularly from the tradeoff between accuracy of the virtual environment description and the memory and CPU usage to allow agents to interact with it. Moreover, the behaviours associated with cells are predefined, which considerably limits the agents' autonomy. Agents should be able to access geometric, topologic and semantic information in order to reason about and make decisions which take into account the characteristics of the virtual environment.

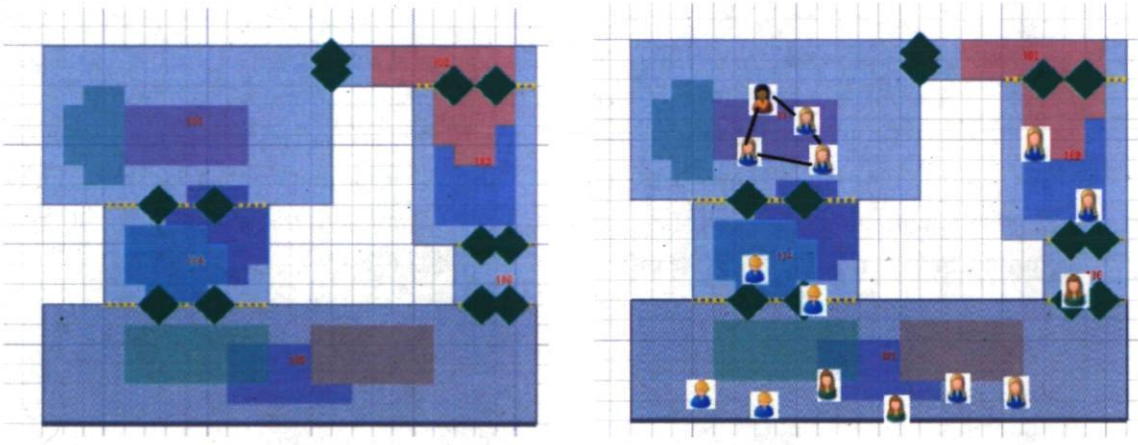


Figure 4.11: Spatial situations can be easily set by drawing directly on the environment. Situation composition can be specified by overlaying regions [SGC04].

Inspired from the work of Lamarche and Donikian [LD04], Shao and Terzopoulos developed an autonomous pedestrian model where virtual humans individually plan their actions, based on a hierarchical collection of maps [ST05, Sha06]. They create: (1) *Topological maps*, which contain nodes and edges. The nodes are bounded volumes in 3D-space, and the edges represent the accessibility between these regions; (2) *A walkable* 2D-surface is deduced from each volume; (3) *Perception maps*, which provide the pedestrians with a sense of the ground height and of the surrounding static and mobile objects; and (4) *Path maps*, which take care of path planning for navigation.

More recently, Da Silveira and Musse [DSM06] introduced a semi-automatic city generation process based on work done by Farenc [Far01a] where semantic information such as population density was used to create the environment. However, this process was limited to a fixed number of different buildings, previously stored in a repository. Moreover, although Da Silveira and Musse exploited semantic data to choose where to put which kinds of buildings, this information was not further used to populate the generated environment or to apply corresponding behaviours.

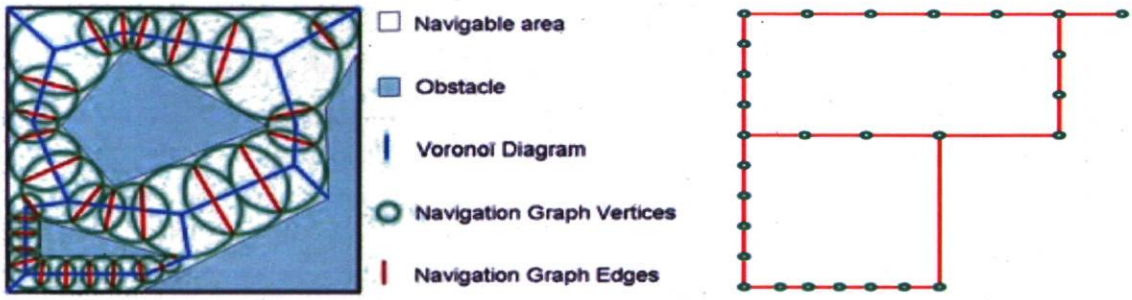


Figure 4.12: left: Geometric representation of a NG in a 2D academic environment. right: NG itself for the same example [PGT07].

Pettré *et al.* [CMY+06, PGT07] proposed a model to support interactive navigation planning and real-time simulation of a very large number of entities moving in a virtual environment. From the environment geometry analysis, Pettré *et al.* compute a structure called *Navigation Graph* as shown in Figure 4.12. Along with this graph-based description of the environment, a set of algorithms is used to answer navigation and path planning queries required by moving agents. Recently, Yersin *et al.* [Yer09, CMY+06] extended the work done by Pettré *et al.* and proposed a model that supports real-time motion planning, navigation, and behaviour for large crowds of virtual humans. However, instead of storing the environmental information in the scene's geometry or in external structures, Yersin directly employs the vertices of the *Navigation Graph* structure.

Although the navigation graph approach allows the support of a large number of moving agents thanks to the graph-based structure, it only provides a geometric description of the geographic environment. This description does not take into account the topologic characteristics of the geographic environment nor the semantic information that qualify the geographic features. Hence, moving agents plan paths without taking into account such important characteristics of the geographic environment. Since these characteristics are missed, agents are not able to reason about their environment and to spatially behave with respect to its description.

4.4 Conclusion

The problem of path planning for moving agents was widely investigated. Algorithms such as *Dijkstra*, *Best First Search*, *A**, *D**, and *Longlife A** have been proposed, but almost all of them consider the problem of path planning for a single moving agent. Moreover, most of these algorithms only consider the distance when computing the path cost, i.e. paths are only optimized under the criterion of length. However, other characteristics of the computed path should be considered such as the path's gradient (slope), the path's width, and the terrain qualification (terrain coverage and use, i.e. sand, water, grass, paved road, etc.) 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. Classic path planners presented in this chapter provide agents with obstacle-free paths between two located positions in the VGE. Such paths do not take into account the environment's characteristics (topologic and semantic) nor the agents' types and capabilities. For example, classic planners assume that all agents are equally capable of reaching most areas in a given map, and any terrain portion which is not traversable by one agent is considered to be not traversable by the other agents. Such assumptions limit the applicability of these planners to solve only a very narrow set of problems: path planning of homogeneous agents in a homogeneous environment. There is a need to address the issue of path planning for agents with different capabilities evolving in complex and large scale geographic environments of various extents.

Spatial behaviors are basically behaviors involving the apprehension of spatial features of the environment in large-scale and complex geographic environments. Several approaches have been proposed to support spatial behaviours of agents evolving within a virtual environment. Although some of these approaches proposed the integration of semantic information in the description of the virtual environment, they did not offer the required tools and algorithms that allow agents access to such information, to reason about it, and to make decisions while taking into account the characteristics of the virtual environment. Moreover, since most of these approaches have been proposed for computer animation purposes, they are usually application-dependent with a limited potential for re-usability in the context of MAGS for spatial behaviour simulation purposes. In addition, since the geographic environment may be large-scale and complex, these approaches are not sufficient to deal with such environments. There is a need for an automated process in order to generate a virtual geographic environment, to integrate semantic information associated with geographic features it may contain, and to abstract its description using the geographic environment's geometric, topologic and semantic characteristics. Finally, in order to preserve the autonomy of situated agents, the environment description and more particularly the semantic information should be accessed and used by agents which reason about it and make decisions with respect to the virtual environment's characteristics.

Chapter 5

Conclusion of the State of the Art

In this state of the art, we presented several approaches used to build virtual environments, to model situated agents, and to support motion planning and path planning of agents within virtual environments. Let us now summarize and discuss this literature review in order to specify the requirements for the simulation of situated agents' spatial behaviours in virtual geographic environments.

5.1 Virtual Geographic Environments

Recent research has produced two fundamental paradigms for modeling geographic environments: the *grid-based* (metric) paradigm and the *topological* paradigm. Grid-based approaches represent environments by evenly-spaced grids [TB96]. Each grid cell may, for example, indicate the presence of an obstacle in the corresponding region of the environment. Topological approaches represent environments by graphs. Nodes in such graphs correspond to distinct situations, places, or landmarks (such as doorways). They are connected by arcs if there exists a direct path between them.

The grid-based approach exhibits orthogonal strengths and weaknesses. Occupancy grids are easy to construct and to maintain even in large-scale environments. Since the intrinsic geometry of a grid directly corresponds to the geometry of the environment, the agent's position within its model can be determined by its position and orientation in the real world which, as shown below, can be determined sufficiently accurately using sensors, in environments of moderate size. As a pleasing consequence, different positions for which sensors measure the same values (i.e., situations that look alike) are naturally disambiguated in grid-based

Grid-based Approach	Vector-based Approach
● easy to build and represent.	● requires low memory complexity (resolution depends on the complexity of the environment)
✗ time and memory consuming when dealing with high resolution.	● provides an accurate localization of real world objects's.
✗ requires accurate localization of the actual objects' position when resolution is not fine enough.	● convenient representation for spatial problem solvers.
✗ poor interface for problem solvers	✗ difficult to construct.

Table 5.1: Advantages and shortcomings of grid-based and topological approaches to represent virtual environments [TB96].

approaches.

On the other hand, grid-based approaches suffer from their enormous space and time complexity. This is because the resolution of a grid must be fine enough to capture every important detail of the world. Compactness is a key advantage of topological representations. Topological maps are usually more compact, since their resolution is determined by the complexity of the environment. Consequently, they permit fast planning, facilitate interfacing to symbolic planners and problem-solvers, and provide more natural interfaces for human instructions. *Table 5.1* summarizes the main advantages and limits of grid-based and topological approaches.

5.2 Situated Agents

The environment is composed of objects which can be perceived and used by the agent and might contain more descriptive semantic information. The agent can perceive the objects of the environment with its senses, and can even consider other agents as potential objects to interact with. The agent then decides what to do using these perceptions and its inner knowledge, then acts on the environment accordingly. The process of perceiving the environment, processing the data to reach a decision, then acting on the environment accordingly is known as the perception-decision-action (PDA) loop. The information stored within the environment enriches the agent's PDA process and can also guide its actions, creating and adapting

animations to allow the agent to interact with the object it is interested in.

In order to perceive the environment, and since the agents and the geographic environment are both virtual, it is possible to allow the agent to directly access the virtual geographic environment's description. This can render agents omniscient and filters might be applied to the data in order to restrict the agent's access to it. But, the agent can also gather information through its synthetic senses, such as vision, hearing or touch, and even through senses not readily available to real humans, such as temperature, pressure and vibration sensors. No approach can be considered to be optimal or correct, since each corresponds to the need of the underlying simulation. The goal of such systems is to allow the agent to gather the information it needs in order to achieve its goals.

After gathering information through perception, the agent can react to that information. The data perceived through the agent's sensors can go through a neural network and directly move the effectors and act on the environment, as is done with stimulus-response systems. Neural networks are well adapted for highly reactive behaviors, but they are difficult to expand and cannot handle completely new and unexpected situations. The agent's actions can also be directed by simple rules reacting to different inputs in rule based systems, in which actions are results of conditions present in the environment. This approach works well as a reactive system since it be easily extended. However, it quickly reaches its limits when the number of rules becomes too high since it is difficult to create non-conflicting rules. State machines can also be used as reactive systems, describing behaviors and the transitions between them. This approach insures temporal coherence between behaviors, which is not guaranteed by other reactive systems. Systems such as HPTS [Don01] and HPTS++ [LD02] can even allow running multiple concurrent behaviors without having to worry about conflicts between them.

The agent can also abstract the information gathered from the environment in order to reason and decide on the best suited actions which will allow it to reach a long term goal. The reasoning process can be achieved through abstract descriptions of the world and the actions possible within it, and the consequences of these actions. This representation can either be done through mathematical operators as it is done with situation calculus and STRIPS, or graphs based on this mathematical representation like GRAPHPLAN [BF95]. Situation calculus formalizes the description of all possible worlds, but suffers from the frame problem which forces the user to assume a closed world environment. STRIPS based approaches, although they are simplified situation calculus, remain very expressive and can perform well in orienting the search towards the goal much more rapidly and with less exploration. It is also possible to represent the actions in hierarchies, and describe different methods achieving the same purpose through hierarchical task networks. This approach allows to represent different resolutions of the same action, but suffers from the complexity of the interconnected

methods and tasks. The choice of the action to undertake can even be influenced by an energy propagation system as it is done in action selection mechanisms. Although, the final selection of the action can oscillate between multiple actions with the same energy, these approaches are nevertheless useful since they can take into account the dynamic changes occurring within the environment. Agents can even formulate plans by emulating the practical reasoning of human beings, according to what they believe is true in the world in order to achieve what they desire through the BDI approach. Plans are constantly refined and re-adapted to the agents evolving desires, but the formalism for such an approach is nonexistent and the compatibility between desires is hard to ensure.

After deciding on what to do, the agent turns again to the environment in order to act on it. Some information stored within the environment will help it to apply its decision. A few approaches store the interaction information within the objects themselves and entirely control the agent's animation during the interaction process. This allows the system to create complex object interactions, but the major drawback is that interactions cannot be stopped once they are started. These approaches do not take into account the agent's position and thus need the agent's position to be constant when the interaction is occurring. It is also possible to build an interaction process that takes into account the object being interacted with, the state of the agent, and the manner in which the agent should undertake the interaction using the PAR architecture. Although generating different interactions depending on multiple dynamic parameters, the actions have to be specified offline and re-created if the target object is changed.

5.3 Motion Planning of Situated Agents

In order to act on the environment, the agent may need to move, navigate, and plan a path in the virtual geographic environment while taking into account both the environment and its own characteristics. Several approaches have been proposed in the field of robotics to address the issue of path planning and navigation in virtual environments. Algorithms such as Dijkstra and A* allow the computation of the shortest path towards a destination position located in the virtual geographic environment. However, most of the research work on path planning used a grid-based representation of the geographic environment which raises the issue of precision and accuracy. Moreover, the virtual geographic environments are usually a two dimensional model. Geographic environments are seldom flat, and ignoring the terrain elevation may affect the quality of the computed path. In addition, elevation may be considered as an obstacle that qualifies the virtual geographic environment (*e.g.* steep slopes). Virtual geographic environments have frequently been modeled from a geometrical perspective. Indeed, the qualification of geographic features have not been addressed. As a consequence,

these virtual geographic environments tend to be monolithic and over-simplified (i.e. only considering free and occupied cells). Actually, a virtual geographic environment description should not only include geometric data, but also be enriched with topologic and semantic information. Since each agent has its own characteristics and capabilities, it should be able to autonomously interact with the virtual geographic environment with respect to these characteristics. Moreover, most of path planning algorithms use the distance in order to compute and to optimize a path within the VGE. However, several other characteristics should be taken into account while planning a path for a situated agent.

For example, simulating a track which securely transports dangerous goods[1] requires planning a path that avoids residential areas. Hence, distance to residential areas becomes the criterion to maximize rather than just minimizing the distance from the origin to the destination. Moreover, transporting wide goods also requires planning a path which will satisfy the minimum path width rather than only optimize the distance. In contrast, emergency response usually requires planning a path that avoids jamming traffic while optimizing the distance. As one can see, such criteria can not be satisfied without an enriched description of the virtual geographic environment.

5.4 Requirements for the Simulation of Spatial Behaviours

Creating a multi-agent geo-simulation involving virtual agents evolving within a virtual geographic environment is indeed a complex task. Many aspects and factors have to be taken into account, at several different levels.

First, since we are interested in simulating agents' spatial behaviours which focus on the interactions occurring between agents and the environment at the individual level, we need an accurate description of the environment as well as of situated agents. Therefore, the data used to build the virtual environment should be ***reliable***. Geographic Information Systems (GIS) provide standard geographic data representing the environment (see Section 2.1). Moreover, the data used to build the virtual environment should be ***accurate***. GIS data in vector format are acknowledged for their *accuracy* and for their capabilities to capture both the *geometric* and the *topologic* characteristics of the geographic environment.

Second, the description of the virtual geographic environment should include ***quantitative data*** such as geometric and topologic data. It should also include ***qualitative information*** which consists of associating semantic information with the geographic features (see Section 2.6).

[1] hazardous materials/dangerous whether it be toxic, radioactive, corrosive, or diagnostic specimens

Third, the virtual geographic environment description should be ***organised*** using a graph-based structure in order to take advantage of efficient algorithms provided by the graph theory (see Section 2.4).

According to our literature review provided in Chapter 2, it becomes clear that in conventional VGE, geo-visualization is typically limited to a two-fold task. First, it creates a display surface on which a subset of digital geographic concepts are projected and symbolized. Second, it provides an interface for the visualization of these graphic symbols. In the former case, the comprehension of what has been visually attended basically relies on the *accuracy* as well as on the *interpretation* of geometric forms. Regarding the *accuracy* aspect, as most VGEs generally use the grid-based approach, they lack precision when representing real geographic environments. Indeed, the grid-based approach, which is based on raster format GIS data, represents the environment using regular geometric forms. Regarding the interpretation aspect of the visualized spatial objects, most of the currently available VGE do not provide users with complete information about these spatial objects. Moreover, these VGE do not allow users to freely integrate additional and complementary information which is usually needed in order to better describe such spatial objects or regions.

In order to provide a meaningful geo-visualization, a VGE should adopt an ***accurate space representation*** technique. It also must allow the ***integration of semantic information*** about specific areas and objects. The concept of VGE and the topological paradigm (used for the spatial representation) provide a way to meet these objectives.

Another important aspect related to the generation of virtual geographic environments for the simulation of spatial behaviours is its complexity. Indeed, the process of building a virtual geographic environment is complex and time- and effort-consuming. It requires various skills including the management and administration of GIS data, the mastering of space decomposition techniques, the geo-visualisation and rendering of the resulting VGE, to name a few. Therefore, this process should be ***automated*** as much as possible according to a well-defined methodology.

In addition, when dealing with a large-scale and complex geographic environment, an ***abstraction process*** should be used in order to optimize its description. In contrast with Thomas's conceptual abstraction [TD03] and Lamarche's topologic abstraction [LD04] that we discussed in Section 2.5 , the virtual geographic environment abstraction process should **rely on *geometric*, *topologic*, and *semantic* characteristics** of the geographic environment. This abstraction process should provide a ***multi-level structure*** of the virtual geographic environment where the highest levels are the most abstracted and the lowest levels are the most detailed.

Moreover, situated agent should be able to take advantage of such an abstracted structure of the geographic environment to support their spatial behaviours. Situated agents should be able to perceive their environment. In addition, they should be able to react to that perception, to extract the information (geometric, topologic, semantic) in order to reason about, to make a decision, and finally to act on the environment. Situated agents should be able to navigate the virtual geographic environments while avoiding obstacles. They should also be able to plan hierarchical paths which take into account their own capabilities as well as the virtual geographic environment's characteristics.

Part II

Contributions

Introduction

In the first part of this thesis, we provided a survey on virtual geographic environments and emphasized the shortcomings of the current approaches for their generation and representation. In addition, we pointed out the need for semantic information in order to enhance the description of virtual geographic environments. We also outlined the importance of semantic information to enable spatial reasoning capabilities of autonomous agents evolving in and interacting with such virtual geographic environments. The second part of this thesis details our contributions and the way to address the above mentioned issues. Our contributions start with our approach for automatically generating informed virtual geographic environments. Then, we present an abstraction-based approach which aims to enhance the informed virtual geographic environments' description and to support large-scale geographic environments. Moreover, we demonstrate how we leverage the geometrically-precise and semantically-enhanced description of geographic environments in order to support spatial reasoning algorithms such as agents' path planning and navigation.

Chapter 6

Automated Generation of Informed Virtual Geographic Environments

6.1 Introduction

In this chapter, we present a complete methodology for the automated generation of a graph-based *Informed Virtual Geographic Environment* (IVGE) (Figure 6.1). Our approach is mainly based on the environment decomposition with links to three-dimensional vector GIS data using *Constrained Delaunay Triangulation* (CDT) techniques. We implemented our IVGE model and produced a software platform that we called *IVGEBuilder* which provides users with a user-friendly interface to easily and automatically generate IVGE.

The rest of the chapter is organized as follows: in Section 6.2, we present our methodology for creating informed virtual geographic environments. Section 6.3 highlights the properties of the IVGE. Section 6.4 introduces the *IVGE-Builder*'s architecture and design characteristics. Section 6.5 illustrates a case study which shows how we leverage the geometrically-precise and semantically-enhanced description of IVGE in the field of wireless communication for communications analysis purposes. Finally, Section 6.6 discusses our automated generation of IVGE while Section 6.7 concludes this chapter.

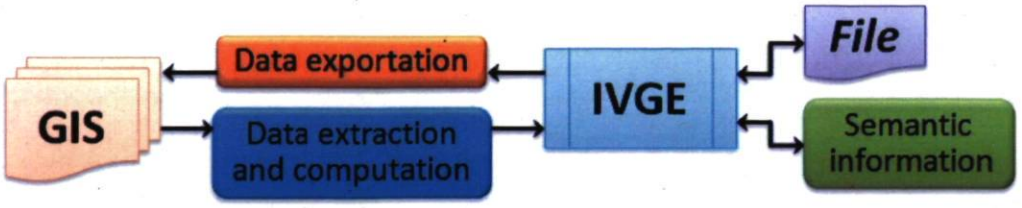


Figure 6.1: Global architecture for IVGE management: in blue, GIS data extraction and associated processes; in orange, IVGE data exportation to GIS format; in green, IVGE's description enhancement using semantic information.

6.2 Generation of Informed VGE

We propose an automated approach to compute the IVGE data directly from vector GIS data. This approach is based on four stages which are detailed in this section (Figure 6.2): *input data selection*, *spatial decomposition*, *maps unification*, and finally the generation of the *informed topologic graph*.

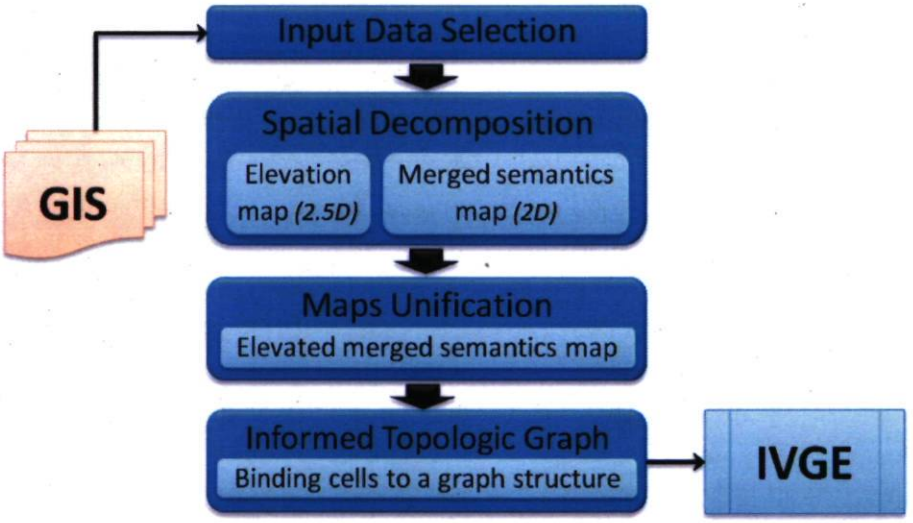


Figure 6.2: The four stages to obtain an IVGE from GIS data. All the stages are automatic but the first.

6.2.1 GIS Input Data Selection

The first step of our approach is the only one requiring human intervention. It consists of selecting the different vector data sets which are used to build the IVGE. The only restriction concerning these data sets is that they must respect the same scale. The input data can be

organized into two categories. First, *elevation layers* contain geographical marks indicating absolute terrain elevations. As we consider 2.5*D* IVGE, a given coordinate cannot have two different elevations, making it impossible to represent tunnels for example. Multiple elevation layers can be specified, and if this limitation is respected, the model can merge them automatically. Second, *semantic layers* are used to qualify various types of data in space. As shown in Figure 6.3 and Figure 6.4, each layer indicates the physical or virtual limits of a given set of features with identical semantics in the geographic environment, such as roads or buildings. The limits can overlap between two layers, and our model is able to merge the information.

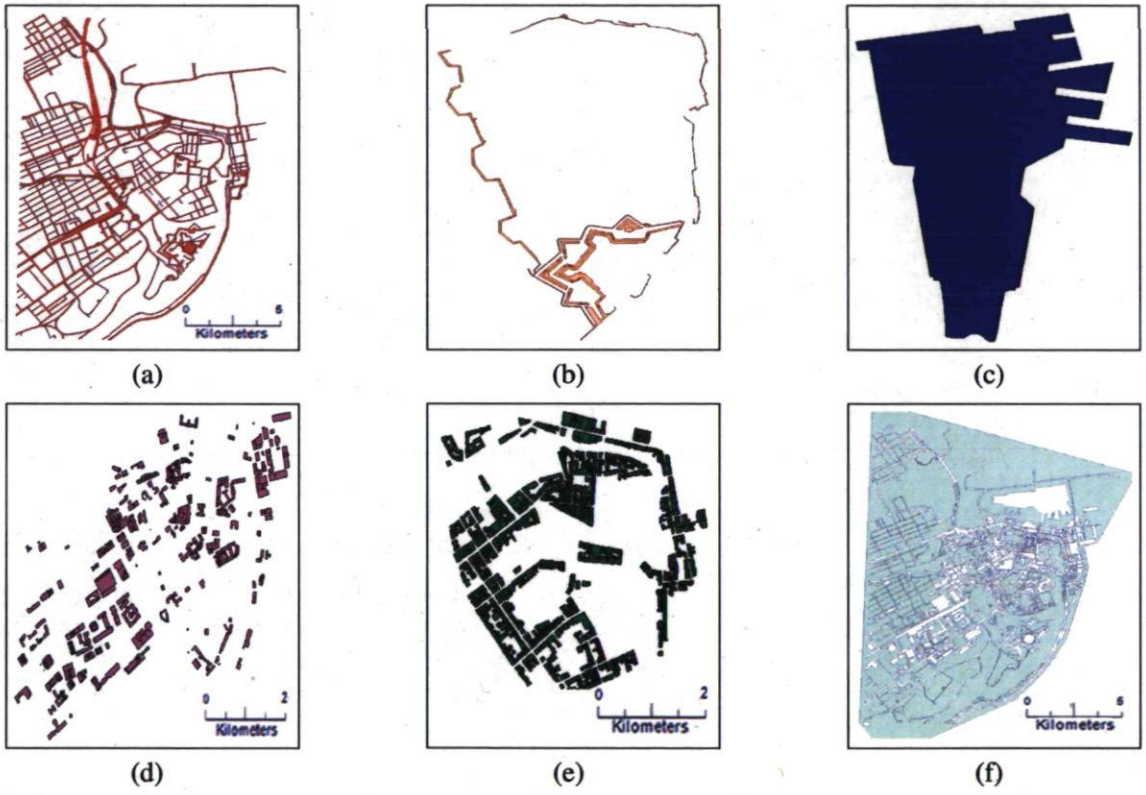
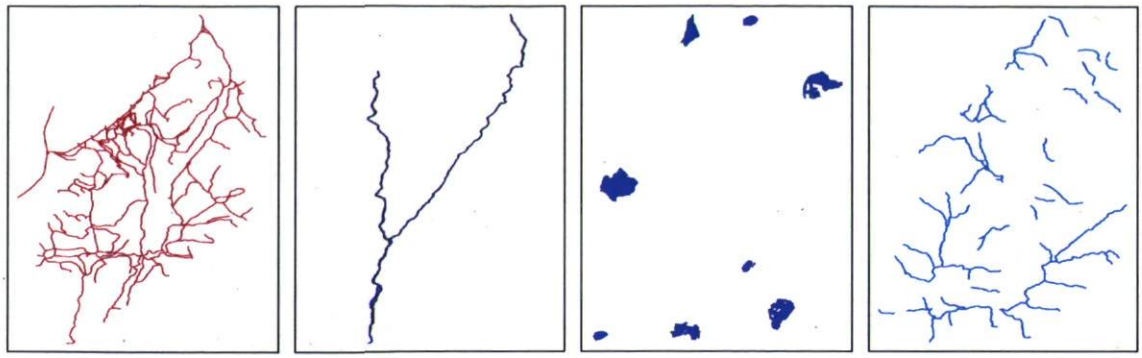


Figure 6.3: Various semantic layers related to Quebec City (Canada): (a) road network; (b) old city wall; (c) marina; (d) governmental buildings; (e) houses

6.2.2 Spatial Decomposition

The second step consists of obtaining an exact spatial decomposition of the input data into cells. This process is entirely automatic using Delaunay triangulation, and can be divided into two parts in relation to the previous phase.

First, an elevation map is computed, corresponding to the triangulation of the elevation



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

Figure 6.4: Various semantic layers related to Montmorency experimental forest: (a) pedestrian walkway network ; (b) river; (c) lakes, (d) water streams.

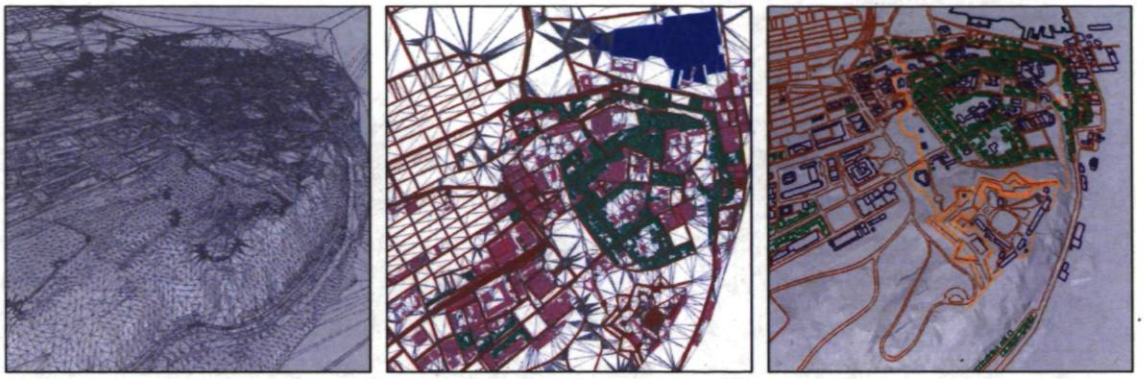
layers. All the elevation points of the layers are injected into a 2D triangulation, the elevation being considered as an attribute of each node. This process produces an environment subdivision composed of connected triangles as shown in Figure 6.5(a) and Figure 6.6(a). Such a subdivision provides information about coplanar areas: the elevation of any point inside a triangle can be deduced by using the elevation of the three original data points to form a plane.

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 using an additional attribute for each semantic layer. The obtained map is then a constrained triangulation merging all input semantics (Figure 6.5(b), Figure 6.6(b)). Each constraint represents as many semantics as the number of input layers containing it.

6.2.3 Merging Elevation and Semantics Layers

The third step to obtain our IVGE consists of unifying the two maps previously obtained. This phase can be depicted as mapping the $2D$ merged semantic map (Figure 6.5(b), Figure 6.6(b)) onto the $2.5D$ elevation map (Figure 6.5(a), Figure 6.6(a)) in order to obtain the final $2.5D$ elevated merged semantics map (Figure 6.5(c), Figure 6.6(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 into the merged semantics triangulation, creating new triangles. This first process can be dropped if the elevation precision is not important. Then, a second process elevates



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

Figure 6.5: The two processed maps (*a*, *b*) and the unified map (*c*). The semantic colours are the same as in figure 6.3, plus red which represents semantics overlapping, and the grey lines which represent triangulation segments.

(a) Triangulated elevation (3*D*). (b) Merged semantics (2*D*). (c) Fused map (3*D*).

Figure 6.6: Spatial decomposition and semantic merging(*a*, *b*) and the fused map (*c*) of the *Montmrency* forest.

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. When P is outside the convex hull of the elevation map then no triangle can be found and the elevation cannot be directly deduced. In this case, we use the average height of the points of the convex hull which are visible from P.

6.2.4 Informed Topologic Graph

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 an *Informed Topologic Graph* (ITG), where each node corresponds to the map's triangles, and each arc corresponds to the adjacency relations between these triangles. Then, common graph algorithms can be applied to this topological graph, and graph traversal algorithms in particular. One of these algorithms retrieves the node, and therefore 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 from the $2.5D$ triangle, and the semantics from its additional attributes. Several other algorithms can be applied, such as path planning or graph abstraction, but they are out of the scope of this chapter and will not be detailed here.

6.3 Properties and Capabilities of Informed VGE

The subdivision of space into convex cells allows us to preserve the original geometric definition of the geographic environment, unlike the grid-based representations that discretise the environment. Furthermore, the proposed data reorganization produces triangles that feature good properties: convexity which facilitates the geometric calculations; support of heterogeneous geometric constraints (points, segments, polygons); Since each constraint is linked to its nearest neighbor, it is easy to compute the widths of the bottlenecks in the virtual geographic environments. The width computation corresponds to the minimum of borders' width that are not qualified as obstacle (Figure 6.7).

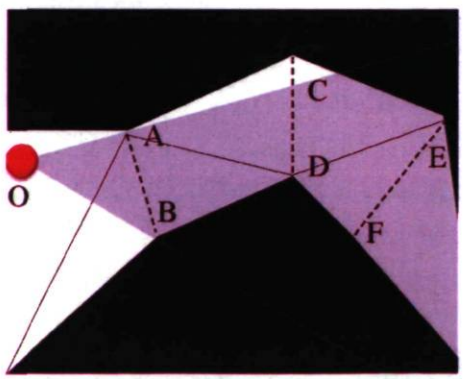


Figure 6.7: Computation of the widths of the bottlenecks in the virtual geographic environment using the widths of (A,B), (C,D) and (E,F) borders which are not obstacles.

The subdivision of space into convex cells also allows us to extract an informed topo-

logic graph of the environment featuring relatively few nodes compared to grid-type representations. Additionally, the triangulation is not dependent on a fixed spatial scale for the environment, but only on its complexity (number of constrained segments). It should also be emphasized that curved geometries will produce a lot of triangles since they are represented by a large number of constrained segments. However, since the produced triangulation is represented as a graph, it is possible to abstract it in order to reduce the number of elements. All these properties are of interest to address the issue of path planning, navigation and line-of-sight visual perception computation.

6.3.1 Support of Agents' Motion Planning

The convex shape of cells generated by the spatial decomposition facilitates the agents' navigation. Indeed, all points belonging to the same convex cell can be connected by a straight obstacle-free path. In addition, since semantic information is associated with each cell's boundary, the agent does not have to compute the distance to obstacles (walls, roads) while navigating in the virtual environment. Indeed, obstacle borders are filtered by the path planning algorithm (see Section 8.2). Moreover, since the topologic properties of cells are computed during their generation, it is easy for an agent to detect and avoid a dead end cell. Finally, the spatial decomposition process generates convex cells and computes their geometric properties including surface, center position, normal vector, and width. Using the width attribute, it is for example easy for an agent to detect and avoid cells whose widths are less than the agent's width.

6.3.2 Accurate Localization of Agents

The issue of the accurate localization of an agent may be thought of as a search for the cell which contains this agent. Once this cell is found, detecting and avoiding obstacles is easy to do as it is based on a simple exploration of the cell's neighborhood. This kind of search is improved by using the graph structure of the IVGE's description. The search algorithm starts with a random selection of a cell (let us note it $cell_{crt}$) from the IVGE. Then, the borders surrounding $cell_{crt}$ are explored. For each border (let us note it b_{crt}), the algorithm proceeds as follows:

1. ***Step 1***: compute the scalar product between the normal to this border (let us note it $\vec{b_{crt}}$) oriented to the inside of the cell and the relative position of the agent compared to the border.

2. ***Step 2***: if this scalar product is positive, then the agent is within $cell_{crt}$; return $cell_{crt}$ and exit.

3. ***Step 3***: if this scalar product is negative, then the agent is not within $cell_{crt}$. The cell sharing b_{crt} with $cell_{crt}$ becomes the new current cell. Proceed with ***Step 1***.

Figure 6.8 illustrates a simplified step-by-step example of agent localization in IVGE. On average, the complexity of such an algorithm based on the number n of cells is $O(\sqrt{n})$. However, it is possible to exploit the spatial and temporal continuity of the simulation in order to keep track of the agent's last visited cell to locate it effectively. The complexity of such an enhanced agent localization algorithm becomes then $O(1)$ as we assume that the spatio-temporal continuity is respected.

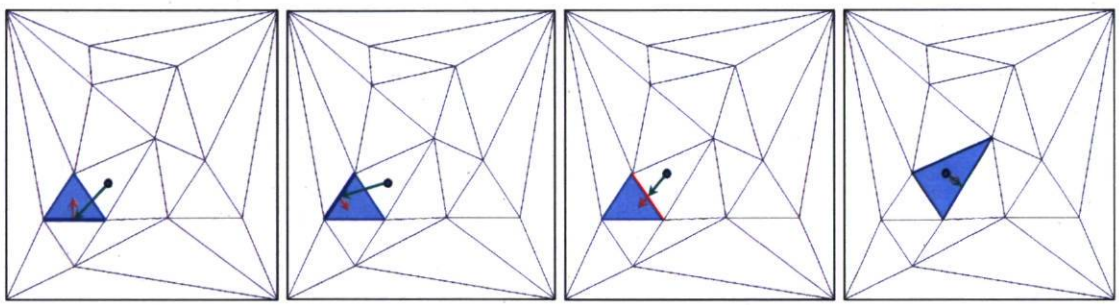


Figure 6.8: Accurate localization of agents: (a, n) a random cell is selected and each of its borders is analyzed; (c) the scalar product is negative, the cell sharing this border (highlighted in red) becomes the current cell; (d) the scalar product is positive, the agent is within the current cell.

6.3.3 Static Obstacle Detection

The IVGE's description provides agents with geometric, topologic, and semantic information regarding their surroundings. This information is associated with cells (geometric) and their borders (topologic, semantic). Once the agent is localized within a cell (using the algorithm presented in Section 6.3.2), it becomes easy to compute its distance to the closest obstacles. To do this, the agent simply computes the distance from its position to borders whose semantics are qualified as *obstacles* such as wall, building, river, etc.

6.3.4 Line of Sight's Computation

The spatial subdivision provides a structure of convex cells which facilitates and accelerates the calculation of ray tracing in three dimensions. We define the radius α using the following information: the position of the origin p; the direction vector $\vec{d}$; and the maximum distance considered. Let $Get_{free}(Cell)$ and $Get_{constrained}(Cell)$ be two functions returning respectively the list of free (S_{free}) and constrained (S_{const}) borders bounding the convex cell $Cell$. Let $N(Cell,b)$ be a function returning the normal vector to the border b which belongs to the cell $Cell$ and directed towards the inside of the cell. Finally let us note $\wp(\beta)$ the $2X2$ rotation matrix of $\vec{d}$. The test checking if there is an intersection between the ray and the border b, which links the vertices I and J, is performed using the following expression:

$$(\vec{d}\cdot N(Cell,b)\leq 0)\wedge(((\tfrac{I+J}{2}-p)\times\wp(\tfrac{\pi}{2}\cdot(I-p))\times((\tfrac{I+J}{2}-p)\times\wp(\tfrac{\pi}{2}\cdot(J-p)))\leq 0$$

Let us denote $\theta=(\frac{I+J}{2}-p)\times\wp(\frac{\pi}{2})$, the previous expression becomes:

$$(\vec{d}\cdot N(Cell,b)\leq 0)\wedge((\theta\cdot(I-p))\times(\theta\cdot(J-p))\leq)0$$

The line of sight computation algorithm proceeds as follows:

- ***Step 1***: the cell $Cell$ containing the source of the line of sight vector $\vec{LoS}$ is determined (using the algorithm presented in Section 6.3.2).
- ***Step 2***: an intersection test is performed between $\vec{LoS}$ and each border b of $Cell$.
- ***Step 3***: compute $S_{free}(Cell)$ using $Get_{free}(Cell)$ and $S_{const}(Cell)$ using $Get_{constrained}(Cell)$.
- ***Step 4***: if no intersection is found with borders from $S_{free}(Cell)$, then $\vec{LoS}$ must intersect with a border from $S_{const}(Cell)$.
- ***Step 5***: the border b is pushed back to the list of borders crossed by $\vec{LoS}$.
- ***Step 6***: the cell $Cell$ is pushed back to the list of cells crossed by $\vec{LoS}$.
- ***Step 7***: the cell sharing the border b which intersects with $\vec{LoS}$ becomes the current cell. Proceed to Step 2.

The line of sight algorithm, due to its low computational cost, can be extensively used in MAGS involving a large number of agents evolving in a complex IVGE. It allows, for example, to determine whether two agents are mutually visible, as we will detail in Section 8.3, or even to obtain the closest obstacle given a direction and a position of an agent located in the IVGE.

6.3.5 Synthesis

Our automated approach for the generation of IVGE:

1. provides a precise description of virtual geographic environments enriched with semantics;
2. leverages the properties of Delaunay triangulations in order to automatically adjust the cells' size according to obstacles' density within the geographic environment;
3. proposes a data structure which allows an agent to easily and efficiently query the IVGE's description about obstacle and dead-end areas;
4. allows an agent to precisely compute distances to obstacles within its surroundings; and
5. enables a fast and precise line of sight computation.

It is important to mention that the memory used by our approach is only dependent on the geometric complexity of the geographic environment rather than on its scale. However, the localization of an agent within the IVGE remains less efficient than grid-based approaches. In order to optimize the agent localization process, we can use the spatio-temporal continuum of agent displacement which allows us to keep track of the cells that have been visited by the agent. Using the last cell visited by the agent, its direction, its speed, we are able to reduce the search space in order to query a subset of cells, in which the agent is supposed to be, instead of the entire set of cells representing the virtual geographic environment. The complexity of the agent localization's process is thus reduced and becomes comparable to grid-based approaches.

6.4 IVGEBuilder: Architecture and Implementation

In order to develop the solution outlined in the preceding sections, we elaborated an architecture and adopted an incremental development approach that allowes us to carefully develop

and integrate the different components required by the project. The first main phase of this development project aimed at automatically creating the IVGE from GIS data. Hence, we needed a suite of software to create the IVGE (Figure 6.9).

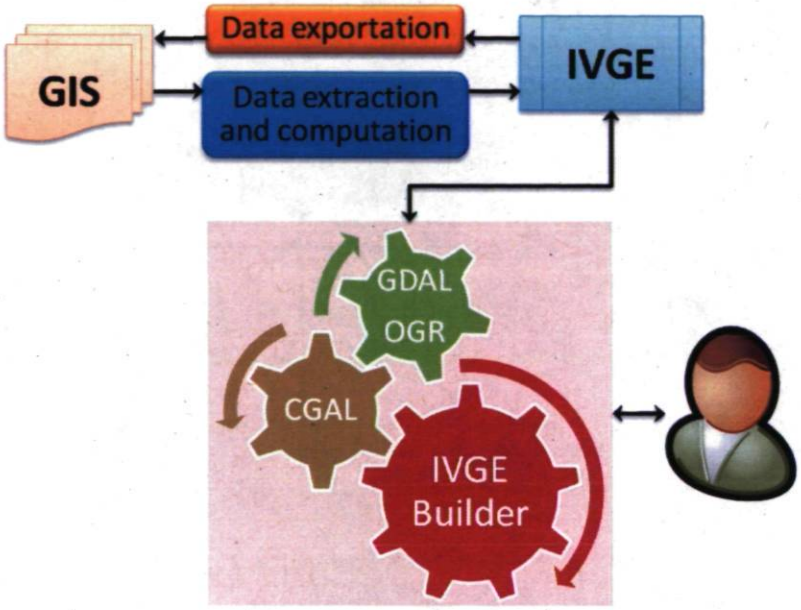


Figure 6.9: Creation of the IVGE from GIS data.

We developed a software application that we called *IVGE-Builder* which allows us to automatically parse and extract various kinds of vector data from GIS files and to process them in order to create a 3D Virtual Geographic Environment (VGE). IVGE-Builder also allows us to enrich the VGE with semantic information, leading to the creation of an Informed VGE (IVGE). Moreover, the description of the IVGE is structured using binary data and can be re-exported in GIS data format in order to enable MAGS users to manipulate it, using standard geographic information systems. As illustrated in Figure 6.10, this tool is based on three software packages: a) *EnvironmentLoader*; b) *TopologicEnvironment*; and c) *MicrosimUtilities*.

MicroSimUtilities is a dependency-free package providing high-level concepts necessary to micro-simulation of multi-agent systems such as geometric computation, event scheduling, and semantic information definition.

TopologicEnvironment is a package dependent on MicroSimUtilities which allows the topologic management of a geographic environment. This package allows us to store and manipulate geometric, topologic and semantic information representing the simulation environment. It provides the object-oriented data structure which defines the graph-based description of the virtual environment as well as the semantic information associated with its geographic features.

EnvironmentLoader is a package which depends on *TopologicEnvironment*, and so, on *MicroSimUtilities*. It imports and exports vector GIS data and uses two external open-source libraries: a) GDAL/OGR [Ope08] and CGAL [CGA] software libraries. *GDAL/OGR* is a *C++* open source library providing read (and sometimes write) access to a wide variety of vector file formats. *CGAL*, standing for *Computational Geometry Algorithms Library*, is an open-source library providing geometrical algorithms such as Delaunay triangulation.

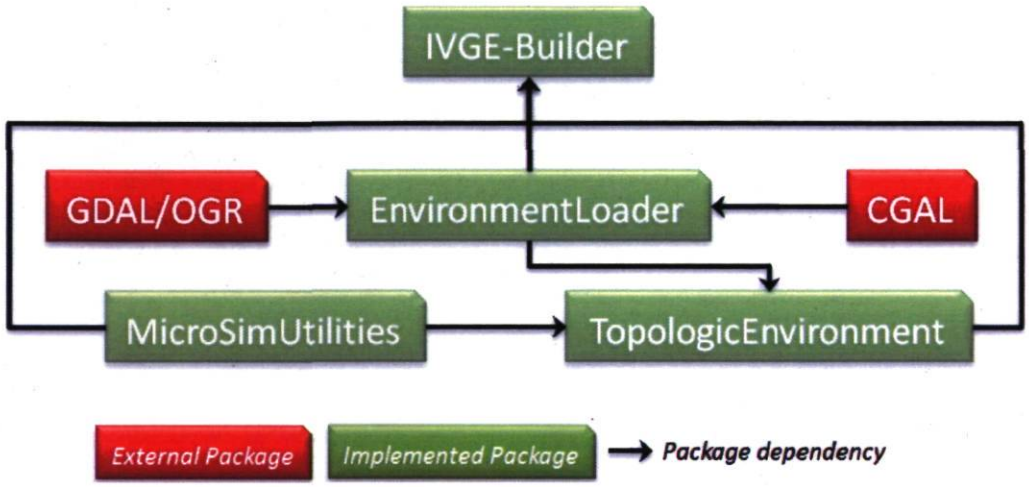


Figure 6.10: Software Architecture of IVGE-Builder.

6.5 Case Study: Analysis of Radio Communication Attenuation Using IVGE

Now that we have presented our automated approach for the generation of geometrically-precise and semantically-informed virtual geographic environments and have outlined the advantages of such an IVGE, we illustrate in this section the use of our approach in the field of wireless networks for the purpose of the analysis of communications' attenuation. We show how we leverage our informed VGE description in the analysis of the effects of terrain shape on communication systems and more particularly on radio signal attenuation between the transmitter and receiver antennas.

6.5.1 Background

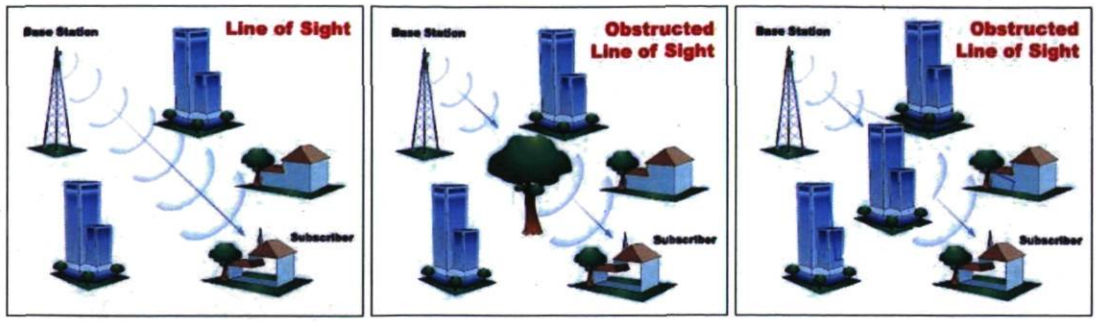
Rapid advances in wireless communications have made mobile data applications a high-growth area of development. So far, most applications only focus on geographic data collection and access using geographic information systems. However, many emergency management applications need such geographic data in order to ensure that field workers and command center operators collaborate under acceptable operating conditions [GM03]. Under emergency conditions, emergency systems need to quickly establish an *ad hoc*, ground-level network of radio stations mounted on temporary command centers, vehicles, or temporary masts, interacting with moving field operators using mobile devices [Gra04].

Radio Frequency (RF) communication does have some limitations that must be considered. The maximum line-of-sight range between two shoulder-height devices is limited to $12km$ considering the curvature of the earth but not considering refraction of radio waves. The actual range may be considerably less depending on transmission power and receiver sensitivity, and the radio signal can be attenuated or degraded due to obstruction resulting from its interactions with features on its transmission path. In an urban environment, possible obstructions include buildings, trees, and bridges for example (Figure 6.11).

The difficulty of evaluating an ad hoc radio network with hundreds of nodes and various levels of mobility operating in a complex geographic environment (i.e. rugged terrain, dense foliage, buildings, etc.) motivated us to use our IVGE model to analyze communication attenuation. In order to analyze the communication attenuation in real geographic environments, we propose to virtually reproduce the actual geographic environment using our automated IVGE generation approach and leverage its enriched description to precisely compute the radio transmission's attenuation. The radio transmission is computed using the line of sight between two points located in the IVGE.

6.5.2 Problem Formulation

Signal specialists currently have limited capabilities for predicting radio performance in complex geographic environments that may involve jammers and may have combinations of open terrain and obstructed locations. Foliated areas, buildings, or terrain may cause obstructions to the radio line-of-sight path. Therefore, there is a need for a communication analysis tool that helps fill this void. We propose to use our geometrically-precise and semantically-enriched IVGE to build a tool for the analysis of radio communications attenuation. This tool is an easy to use, stand-alone, GUI-based application that runs on a PC and provides a rich set of functionalities to aid the user to compute the total path loss of a given operational scenario



(a) (b) (c)

Figure 6.11: Radio signal propagation: (a) an obstruction-free propagation; (b) propagation obstructed by vegetation and foliage; and (c) propagation obstructed by buildings.

that is directly coupled to an operational area using reliable GIS data.

This tool enables the user to plan radio deployments and determine link connectivity using actual radio parameters, taking into account the presence of obstacles, while accounting for the excess attenuation due to terrain, foliage (vegetation), and building obstructions. To compute the total path loss, our tool uses the following parameters: height of transmitter antenna (meters), height of receive antenna (meters), transmitter antenna position (x, y, z), receiver antenna position (x, y, z), and frequency of operation (GHz).

6.5.3 Proposed Approach

Planning communications links requires the ability to assess the performance of each link in the presence of a number of degrading factors. In addition to the normal free space signal attenuation loss, other losses reduce the signal level. These additional losses can be caused by obstructions such as buildings and vegetation (foliage). Analyzing obstruction losses can be difficult because of the geometric, topologic, and semantic characteristics of the geographic environment and the need for path loss models.

We use a ray-tracing approach which determines the path that a radio signal takes to arrive at the receiver's position from a given transmitter within our 3D IVGE. The ray-tracer implemented in the tool uses optimized algorithms for detecting direct (i.e., line of sight) paths (Figure 6.12). An analytical ray-tracing technique [LYLN07] is used rather than the approximate technique, for example, as used in [SR94]. For the analytical approach, the transmitter and receiver are each modelled as infinitesimally small points such that paths are computed precisely and cannot be duplicated or missed as sometimes could happen in the

approximate approach. Obviously, the analytical technique is more precise and reliable. Our approach computes the total source-to-destination path length and then determines whether the vector defined by the source and destination points (locations in the IVGE) passes through an obstruction area. Doing so, it is able to compute the total path loss between transmitter and receiver antennas. One of three cases may occur: 1) *Obstruction-Free Path*: Vector does not penetrate any obstruction; 2) *Obstruction Block Penetration*: Vector penetrates one or more foliage obstruction blocks and/or one or more building obstruction blocks; and 3) *Ground Penetration*: Vector penetrates the ground (earth) once or several times.

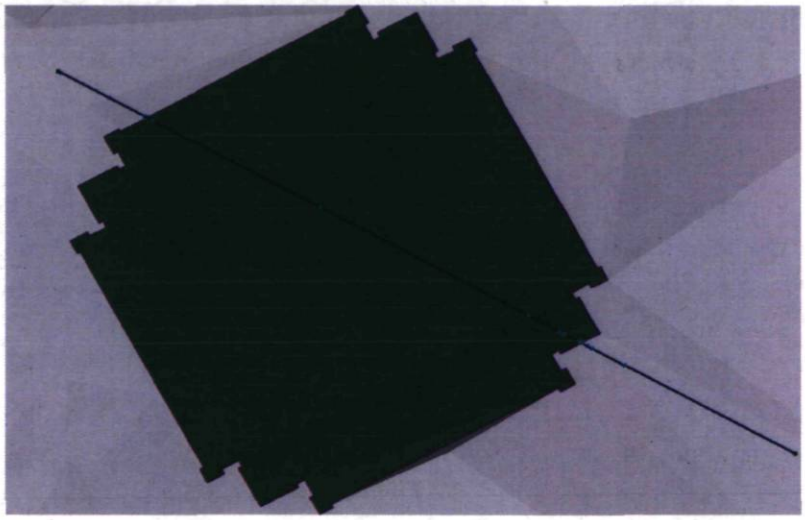


Figure 6.12: Computation of the ray tracing: the radio signal propagation path (blue), free space (grey), building (green), and vertices representing the intersection locations (light blue).

The computations performed by our tool to quantify the path attenuation for each of the three cases defined above are based on the following mathematical models described in [CSSM03].

Plane-Earth Attenuation Model Let L_p be the path attenuation using the plane-Earth model (dB):

$$L_p = 40Log(D) - 20Log(H_t) - 20Log(H_r) \tag{6.1}$$

where D is the total source-to-destination path length (meters), and H_t and H_r are the heights of the transmitter and receiver antennae above ground level, respectively (meters).

Free-Space Attenuation Model Let L_{fs} be the path attenuation using the Free-Space model (dB):

$$L_{fs} = 32.45 + 20Log(D) + 20Log(f) \tag{6.2}$$

where D is the total source-to-destination path length (meters) and f is the RF frequency (GHz).

Obstruction Block Penetration Model Let L_B be the path attenuation term due to propagation through a building obstruction block (dB) [SCM03]:

$$L_B = K_1(0.6)^f + K_2 D_B \tag{6.3}$$

where f is the RF frequency (GHz), K_1 is a constant used to map the first expression above to building penetration data reported in [CSSM03]. $K_1 = 35$; K_2 is a constant to account for the attenuation (per meter) of the signal within the building. $K_2 = 1$ (dB/m); and D_B is the distance that the signal propagates through the building (meters).

The first term in Equation 6.3 accounts for the penetration into and out of the building by the signal and was derived from data reported in [CSSM03] using a regression analysis technique. The second term in the equation above accounts for the attenuation through the building and is based on data reported by Willassen in [Wil98].

Foliage obstruction Attenuation Model If the penetration is through a foliage obstruction block, the tool computes an excess path attenuation term called L_f using the Wiessberger model [Wei82] as follows:

$$L_f = 1.33 f^{0.284}.D_f^{0.588}, 14 < D_f \tag{6.4}$$

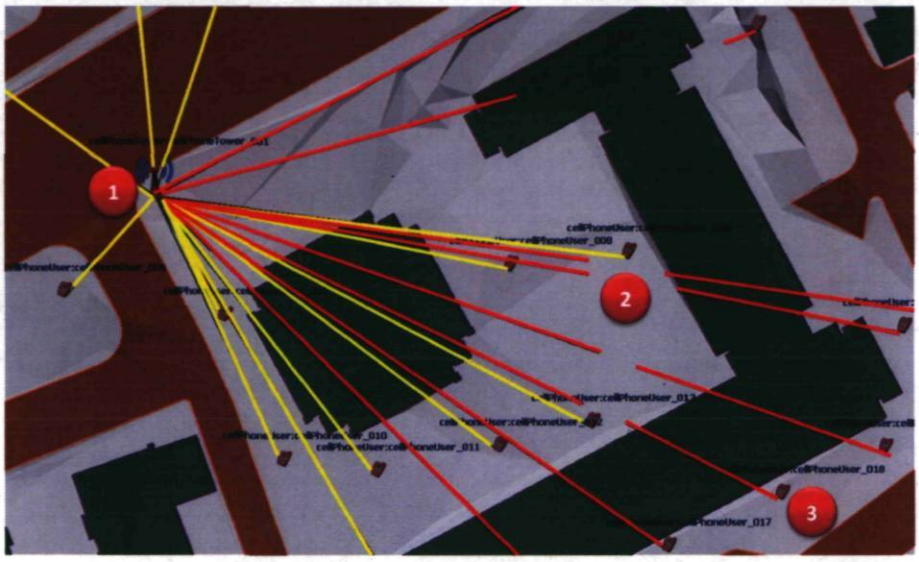
$$L_f = 0.45 f^{0.284}.D_f^{1.0}, \ 0 \leq D_f \leq 14m \tag{6.5}$$

where D_f is the distance that the signal propagates through the foliage obstruction (meters) and f is the RF frequency (GHz).

The term 'excess attenuation' refers to the additional attenuation above the basic transmission loss, for a given path length, in the absence of foliage. Our analysis approach applies equations 6.1 to 6.5 to calculate the basic transmission loss for the link. Thus, the total path attenuation for the obstruction penetration case called L_{total} is calculated as follows:

$$L_{total} = max(L_p, L_{fs})_{Deff} + sum(L_f) + sum(L_B) \tag{6.6}$$

where $max(L_p, Lfs)_{Deff}$ means that this term is calculated at $Deff$; $Deff$ is the effective path length over which the *Plane-Earth* and *Free-Space* path attenuation model is applied and is equal to the total path length minus the sum of the building obstruction block path length segments or sum(DB); $sum(L_f)$ is the sum of the excess attenuation terms in dB due to signal propagation through the foliage obstruction block(s) (dB); and $sum(L_B)$ is the sum of the path attenuation terms due to propagation through the building obstruction block(s) (dB).



(a)

Figure 6.13: Simulation of radio communications' attenuation; yellow lines correspond to obstacle-free line-of-sight radio signal propagation; red lines correspond to obstructed line-of-sight; (1) represents the transmitter antenna implemented using the agent paradigm; (2) an example of a plane-earth obstruction; (3) an example of a block-penetration obstruction.

Figure 6.13 and Figure 6.14 illustrate the agent-based simulation tool that we developed in order to implement our approach to analyse the radio signal attenuation in informed virtual geographic environments. Figure 6.13(a) presents a snapshot of the simulation at time t_0 with an agent representing a transmitter antenna and several agents representing receiver antennae.

6.5.4 Discussion and Conclusion

Using reliable GIS data along with the line of sight algorithm (ray tracing feature) provided by our IVGE allows the system to compute the exact locations of intersections occurring between the radio signal propagation path and the terrain shape (Figure 6.13). Our IVGE also allows us to collect the list of cells crossed by the radio signal propagation path (Figure 6.14). In

addition, using the semantic information associated with these cells, we are able to determine which analytical model to apply in order to precisely compute the path loss.

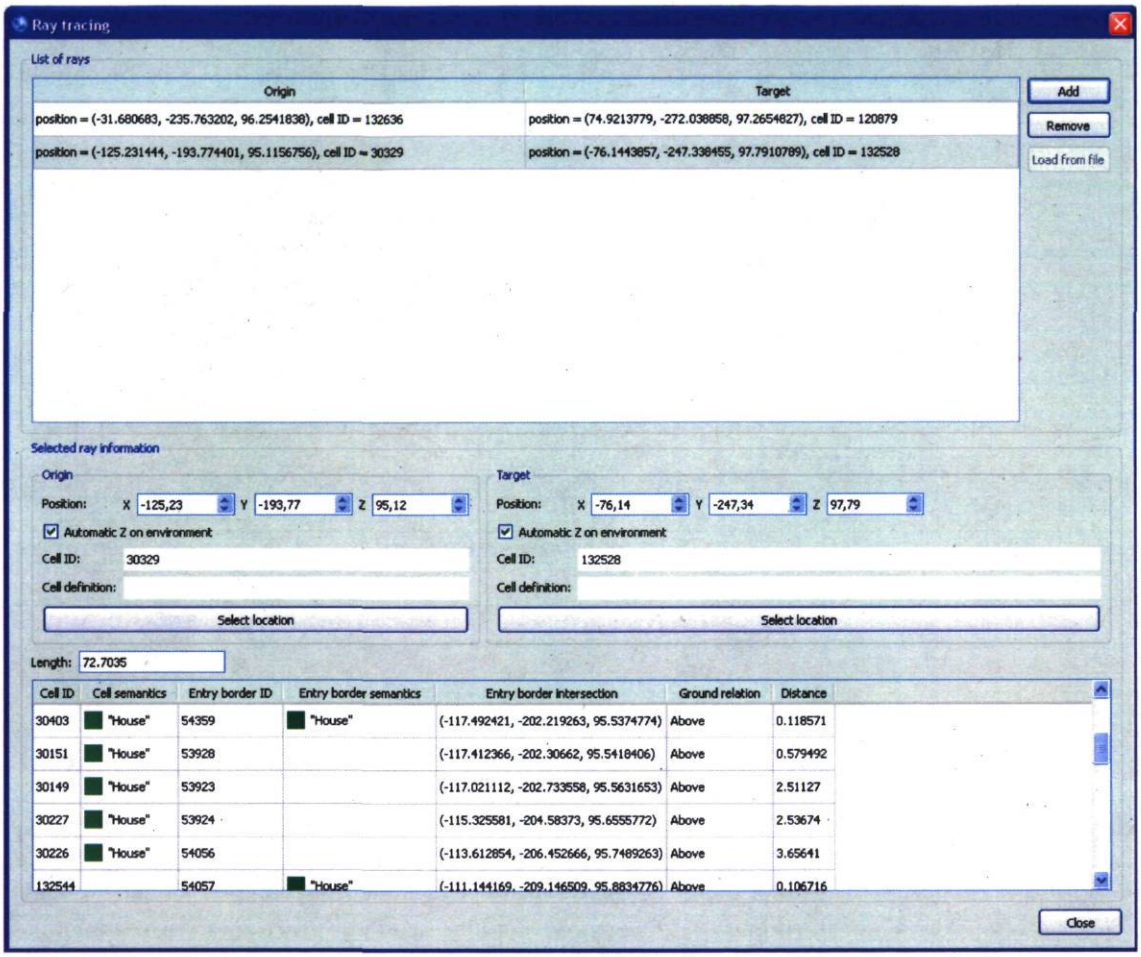


Figure 6.14: The graphic user-interface for the computation of the radio signal line-of-sight; Cells and borders traversed by the line of sight are reported along with their geometric, topologic, and semantic characteristics.

We have shown a tool that leverages the enriched description of the IVGE and computes the radio signal attenuation due to buildings, foliage and field obstructions. However, other phenomena can also degrade the radio signal transmission. Examples of such phenomena include transmitter power, receiver sensitivity, and radio signal's absorption, reflection, and scattering from interaction with features on or near its transmission path. These phenomena should be taken into account when computing the radio signal attenuation in order to predict anticipated communications network connectivity and performance data. To this end, the agent models representing the transmitters and receivers antennae should be improved in order to take into account the antenna sensitivity. Moreover, the computation of the signal propagation should also take in account absorption, reflection and scattering phenomena.

In the future, we propose to extend our tool in order to integrate an advanced ray-tracing process [Mat05] which combines both the geometric optics and Keller's [Kel62] geometric theory of diffraction (GTD). Moreover, we propose to include the uniform theory of diffraction (UTD) [KP74] extension to GTD which removes the inaccuracies close to the incident and reflection boundaries.

To conclude, our geometrically-precise and semantically-enhanced IVGE enables us to provide wireless network planners with a tool for the analysis of the communications' attenuation. In contrast with mathematical models which only approximate the radio signal attenuation based on a coarse-grained qualification of the geographic environment: *urban*, *suburban* and *rural*, we compute more precisely the radio signal propagation path and qualify obstructions in order to apply the appropriate analytical model.

6.6 Discussion

Our approach goes beyond classical grid-based descriptions of virtual geographic environments by combining the semantic information and the vector-based representations (Figure 6.15). Indeed, as shown in *Table 6.1*, our topological method combines the advantages of grids and vector layers, and avoids their respective drawbacks. Moreover, this data extraction method is fully automated, being able to directly process GIS vector data. Thanks to the automatic extraction method that we showed, our system handles the IVGE construction directly from a specified set of vector format GIS files.

The performances of the extraction process are very good, being able to process an area such as the center part of Quebec-city, with one elevation map containing 37880 vertices, and several semantic layers containing 22286 polygons, 75739 lines and, 126650 vertices in less than 5 seconds on a standard computer (Intel Core 2 Duo processor 2.13*Ghz*, 1*Go* RAM). The resulting unified map approximately contains 122,000 triangles. Besides, the necessary time to obtain the triangle corresponding to a given coordinate is negligible.

6.7 Conclusion

To conclude, we presented in this chapter a four-stage automated approach for the generation of geometrically-precise and semantically-informed VGEs. This approach is based on vector format GIS data representing a real geographic environment. These data are processed in

order to build a precise informed graph enriched with semantic information. The geometric versatility of triangulation as a tool for precise decomposition of complex geographic environments, combined with the integration of semantic information, make our approach an original model for IVGE.

Interest in the effects of terrain shape on communication systems has increased in recent years because of the widespread deployment of wireless broadband systems and demands for mobile high data rates access. Such an interest motivated us to apply our IVGE model in order to analyse the communications' attenuation and to precisely compute the total loss of the radio signal power depending on the qualification of the traversed area (vegetation, building, field, free space). Moreover, the prototype system (IVGE-Builder) we developed proved that the application of the proposed model based on reliable vectorial GIS data sources is possible and automatically provides a semantically-enriched and geometrically-precise 2.5*D* IVGE description.

The informed graph still needs enhancements particularly regarding its size and structure when dealing with large-scale and complex geographic environments and regarding the qualification of the terrain's elevation. In the following chapter, we describe how to improve IVGE by adding new information to the environment description in order to deal with complex and large-scale geographic environments using an abstraction process.

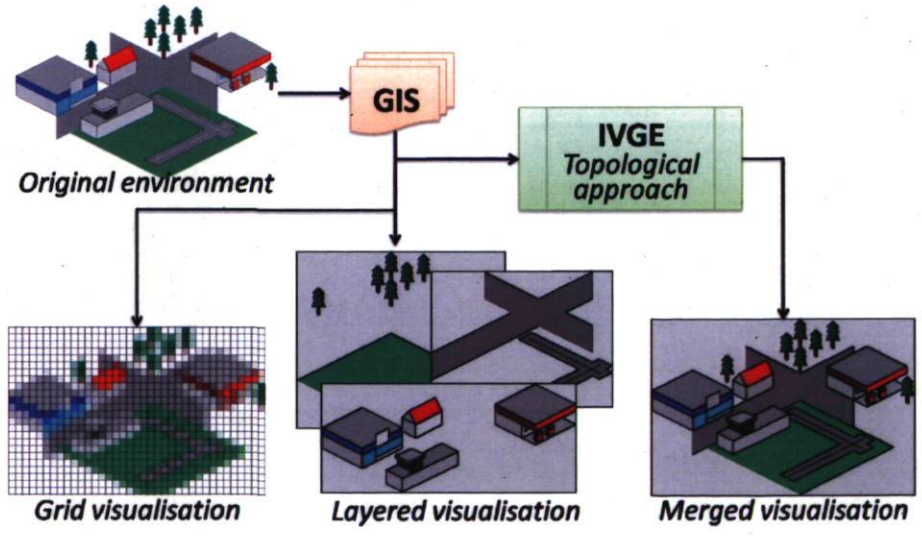


Figure 6.15: The proposed graph-based IVGE model merges semantic information and preserves geometrical precision. In contrast with the grid-based approach on the left hand side of the figure, the topologic approach enables both layered and merged visualisation capabilities.

Grid	Vector layer	Topological
✔ Easy to fill ✔ Semantics merging ✔ Easy to examine --- ✘ Constant scale ✘ Problem of precise localization ✘ Balance precision/ memory complexity	✔ Adaptable scale ✔ Memory complexity depends on geometry ✔ Arbitrarily complex and precise representation of features --- ✘ Hard to access and to manipulate ✘ Very hard to merge spatially overlapping semantic data	✔ Adaptable scale ✔ Memory complexity depends on geometry ✔ Exact representation of input data (accuracy) ✔ Multiple semantics merging ✔ Easy to examine ✔ Graphs algorithms and improvements

Table 6.1: Advantages and drawbacks of the existing approaches compared to our topological model.

Chapter 7

Abstraction of Informed Virtual Geographic Environments

7.1 Introduction

The IVGE description resulting from the exact spatial subdivision and semantics merging presented in Chapter 6 provides a geometrically-precise and semantically-informed virtual geographic environment. However, this description needs to be enhanced in order to enable its efficient exploitation in MAGS which involve spatially-reasoning agents evolving in and interacting with a large-scale and complex virtual geographic environment. As stated in Section 2.5 and discussed in Section 5.4, the description of such large-scale and complex geographic environment needs to be enhanced and improved using an abstraction process.

In this chapter, we describe an abstract process which enhances the description of the IVGE. Figure 7.1 illustrates the way we extend our IVGE model: the left hand side of Figure 7.1 shows the four stages to build an IVGE as described in Chapter 6,and the right hand side shows the proposed enhancement techniques.

Section 7.2 presents the first enhancement which is related to the qualification of terrain. We propose a novel approach of information extrapolation using a one-time spatial reasoning process based on a geometric abstraction. This approach can be used to fix input elevation errors, as well as to create new qualitative data relative to elevation variations. These data are stored as additional semantics bound to the graph nodes, which can subsequently be used for spatial reasoning. Section 7.3 introduces the second enhancement which optimises the size of the informed graph structure using a topological abstraction process. This process

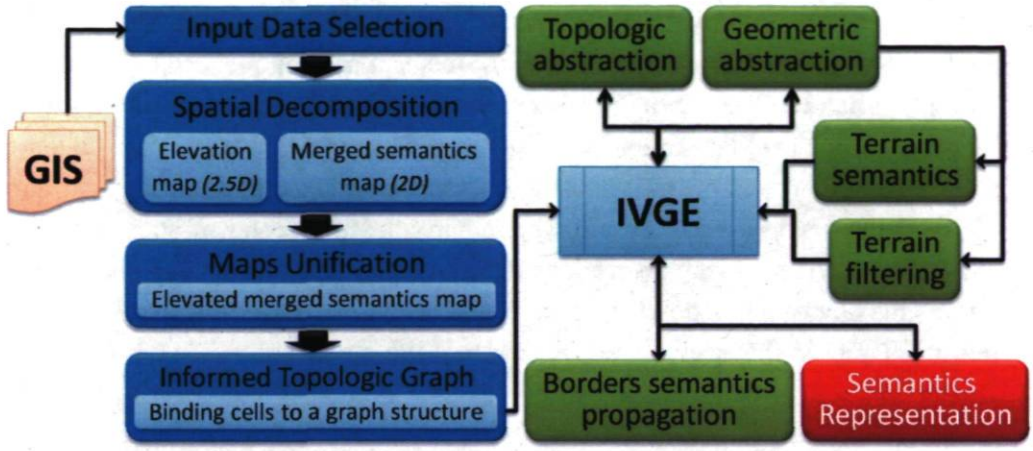


Figure 7.1: Global architecture for IVGE management: in blue, GIS data extraction and associated processes; in orange, IVGE data exportation to GIS format; in green and red, optional stages for IVGE information enhancement; in red, the enhancement of the semantic information representation.

aims at building an hierarchical topologic graph structure in order to deal with large-scale virtual geographic environments. Section 7.4 details the third enhancement technique which propagates qualitative input information from the arcs of the graph to the nodes, which allows deduction of the internal parts of features such as buildings or roads in addition to their boundaries. Moreover, this technique uses Conceptual Graphs (CG) [SW86], a standard formalism for the representation of semantic information. Section 7.5 presents a case study illustrating a semantic abstraction of an urban IVGE. Section 7.6 details a case study showing a holonic-based abstraction model which combines the semantic and the topologic abstraction processes. Section 7.7 presents a software application that we called *IVGE-Viewer* which aims to provide MAGS users with a set of functionalities in order to easily visualise IVGE, to manage semantic information, to manipulate IVGE instances, and to abstract the IVGE's description. IVGE-Viewer is built up on the *IVGE-Builder* package we described in Section 6.4. Finally, Section 7.8 closes with remarks on the enhancements of the IVGE model.

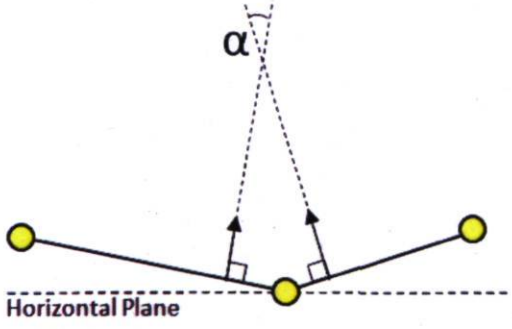
7.2 Geometric abstraction for the qualification of terrain

Spatial decomposition subdivides the environment into convex cells. Such cells encapsulate various quantitative geometric data which are suitable for precise computations. Since geographic environments are seldom flat, it is important to consider the terrain's elevation and shape. While elevation data are stored in a quantitative way which is suitable for exact calculations, spatial reasoning often needs to manipulate qualitative information. Indeed, when considering a slope, it is obviously simpler and faster to qualify it using an attribute with

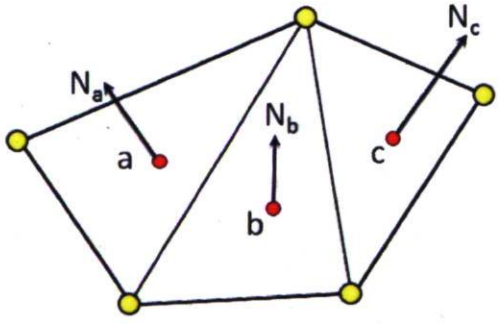
ordinal values such as *gentle* and *steep* rather than using numerical values. However, when dealing with large scale geographic environments, handling the terrain's elevation, including its light variations, may be a complex task. To this end, we propose an abstraction process that uses geometric data to extract the average terrain's elevation information from spatial areas. The objectives of this *Geometric Abstraction* are threefold. First, it aims to reduce the amount of data used to describe the environment. Second, it helps for the detection of anomalies, deviations, and aberrations in elevation data. Third, the geometric abstraction enhances the environment description by integrating qualitative information characterising the terrain shape. In this section, we first present the algorithm which computes the geometric abstraction. Then, we describe two processes which use the geometric abstraction, namely *Filtering elevation anomalies* and *Extracting elevation semantics*.

7.2.1 Geometric Abstraction Algorithm

As presented in the previous chapter, the geographic environment is subdivided into cells of different shapes and sizes. The algorithm takes advantage of the graph structure obtained from the IVGE extraction process (see Section 6.2.4). A *cell* corresponds to a node in the topological graph. A node represents a triangle generated by the *CDT* spatial decomposition technique. A cell is characterized by its boundaries, its neighbouring cells, its surface as well as its normal vector which is a vector perpendicular to its plane (Figure 7.2(a)).



(a) Two adjacent cells with α the angle between their normal vectors.



(b) Adjacent cells with normal vectors perpendicular to their respective planes.

Figure 7.2: (*a*) illustrates the coplanarity test for two cells and (*b*) shows unit normal vectors $\vec{N_a}$, $\vec{N_b}$, $\vec{N_c}$ for cells *a*, *b*, and *c* respectively.

Now we introduce the notion of a group, which is a collection of adjacent cells. The grouping strategy is based on a coplanarity criterion which is assessed by computing the difference between the *normal vectors* of two neighbouring cells or groups of cells (Figure 7.2(b)). Since a group is basically composed of adjacent cells it is obvious to characterise

a group by its boundaries, its neighbouring groups, its surface, as well as its normal vector. However, the normal vector of a group must rely on an interpretation of the normal vectors of its composing cells. In order to compute the normal vector of a group, we adopt the *area-weight normal vector* [CW05] which takes into account the unit normal vectors of its composing cells as well as their respective surfaces. Let S_c denote the surface area of a cell c and $\overrightarrow{N_c}$ be its unit normal vector. The area-weight normal vector $\overrightarrow{N_G}$ of a group G is computed as follows:

$$\overrightarrow{N_G} = \sum_{c \in G} \left(S_c \,.\, \overrightarrow{N_c} \right) / \sum_{c \in G} S_c \tag{7.1}$$

The geometric abstraction algorithm uses two input parameters: 1) a set of *starting cells* which act as access points to the graph structure, and 2) a Δ parameter which corresponds to the maximal allowed difference between cells' gradients. Two adjacent cells are considered coplanar, and hence grouped, when the angle between their normal vectors (α in Figure 7.2(a)) is lesser than Δ. The recursive geometric abstraction algorithm (1) is composed of five steps:

1. For each cell c of the *starting cells*, create a new group G and do step 2.
2. For each neighbouring group or cell n of G, if the neighbour has already been processed, do step 3, else do step 5.
3. If angle $\left(\overrightarrow{N_G}, \overrightarrow{N_n}\right) \leq \Delta$ then do step 4. Otherwise do step 5.
4. Merge n in group G, then evaluate $\overrightarrow{N_G}$ using equation (7.1). Do step 2 again for G.
5. If n is an unprocessed cell, create a new group G with n and do step 2.

The algorithm starts by visiting all the cells of the virtual environment (line 1 of *Algorithm 1*). For each visited cell *crt_cell*, a new group *crt_grp* is created and the cell is registered as a member (line 2). The area-weighted normal vector of *crt_grp* is computed using equation (7.1). Besides, the algorithm tests the coplanarity of *crt_cell* with its neighbouring cells (*nxt_cell* belonging to *nxt_grp*) using equation (7.1) and to decide whether to include these neighbours in the group *crt_grp* to which *crt_cell* belongs. Next, the algorithm explores and processes the neighbouring cells of *crt_cell* (line 4). For each neighbour, if it is visited for the first time, a new group is created and the neighbour cell is registered as its first member.

Afterwards, the algorithm computes the angles resulting from the merging of *crt_grp* and *nxt_grp* groups. The area-weighted normal vector resulting from the integration of *crt_grp*'s elements in the *nxt_grp* group is computed. The algorithm goes on by computing the angle between the new (after the merge) and the previous (before the merge) area-weighted normal

Algorithm 1 GeometricAbstraction

PARAMETERS: ***start_cells*** the starting cells; ***crt_cell*** the current cell; Δ the gradient difference threshold.

REQUIRE: ***neighbours_cells*** the container of neighbour cells; ***crt_grp*** the group assigned to the current cell; ***nxt_grp*** the group assigned to the neighbour cell.

```
1:  FOR ALL crt_cell ∈ start_cells DO
2:      crt_grp ← BUILDNEWGROUP(crt_cell)
3:      neighbours_cont ← NEIGHBOURS(crt_cell)
4:      FOR ALL nxt_cell ∈ neighbours_cont DO
5:          nxt_grp ← BUILDNEWGROUP(nxt_cell)
6:          coplan_value ← COMPUTECOPLANARITY(crt_grp,nxt_grp)
7:          IF coplan_value ≤ Δ THEN
8:              crt_grp ← MERGEGROUPS(crt_grp,nxt_grp)
9:          END IF
10:     END FOR
11: END FOR
```

vectors. The angle is given by the scalar product of the two normalised vectors $\vec{N}_{crt\_grp}$ and $\vec{N}_{nxt\_grp}$. If this angle respects the input parameter Δ (line 9), then merging is performed (line 10).

In the proposed algorithm, the geometric abstraction produces coherent groups whose cells are coplanar and with respect to the Δ threshold. The geometric abstraction process abstracts a higher-order topologic graph and produces a new graph with fewer nodes which helps to enhance performance of spatial reasoning mechanisms.

The analysis of the resulting groups helps to identify anomalies in elevation data. Such anomalies need to be fixed in order to build a realistic virtual geographic environment. Furthermore, the average terrain slope which characterises each group is a quantitative datum described using area-weighted normal vectors. Such quantitative data are too precise to be used by qualitative spatial reasoning. Hence, a qualification process would greatly simplify spatial reasoning mechanisms. Thus the geometric abstraction can improve *IVGE* by filtering the elevation anomalies, qualifying the terrain slope using semantics and integrating such semantics in the description of the geographic environment.

7.2.2 Filtering elevation anomalies

Analysis of the geometric abstraction may reveal an isolated group which is totally surrounded by another single coherent group. These groups are characterised by a large difference between their respective area-weighted normal vectors. Such isolated groups are often characterised by their small surface areas and can usually be considered as anomalies, deviations, or aberrations in the initial elevation data. MAGS users may verify if such groups correspond to real pits or depressions, or substantial mounds or heaps on the landscape. The geometric abstraction process helps to identify them and can help to automatically filter such anomalies using a two phase process. First, isolated groups are identified (Figure 7.3(a)). The identification of isolated groups is based on two key parameters: 1) the ratio between the surface areas of the surrounded and surrounding groups, and 2) the difference between the area-weighted normal vectors of the surrounded and surrounding groups. Second, these isolated groups are adjusted to the average level of elevation of the surrounding ones (Figure 7.3(b)). The lowest and the highest elevations (*low_elev*, *high_elev*) of the surrounding group (*surrounding_grp*) are computed. Then, the elevation of all the vertices of the isolated group (*isolated_grp*) are adjusted using the average between *lowest_elevation* and *highest_elevation*. As a consequence, we obtain more coherent groups in which anomalies of elevation data are corrected.

(a) Identifying elevation anomalies. Two isolated groups (in red) and angles (α1 and α2) resulting from the difference between the area-weighted normal vectors

(b) Fixing elevation anomalies. Nodes in isolated groups are adjusted to the average elevation level.

Figure 7.3: Profile section of anomalous *Isolated Groups* (red colour) adjusted to the average elevation of the surrounding ones (yellow colour).

7.2.3 Qualification of terrain shape

The geometric abstraction algorithm computes quantitative geometric data which precisely describe the terrain However, handling and exploiting quantitative data is a complex task as the range of values may be too large and calculations or analysis methods may be too costly.

Therefore, we propose to interpret the quantitative data representing the terrain shape by qualifying the terrain characteristics. Semantic labels, which are called *the shape semantics*, are associated to quantitative intervals of values that represent the terrain's shape. In order to obtain the shape's semantics we propose a two-step process taking advantage of the geometric abstraction: 1) calculation of the inclination, or the angle α between the weighted normal vector $\vec{N_g}$ of a group *grp* and the horizontal plane; and 2) assigning to each discrete value a semantic category which qualifies it. The discretisation process can be done in two ways: a *customised* and an *automated* approach.

The *customised approach* requires that the user provides a complete specification of the discretisation to qualify the range of slopes. Indeed, the user needs to specify a list of inclination intervals as well as their associated semantic labels. The algorithm iterates over the groups obtained by the geometric abstraction. For each group *grp*, it calculates the inclination value *I*. Then, this process checks the interval bounds and determines in which one the inclination value *I* falls. Finally, the customised discretisation extracts the semantic shape label from the selected slope interval and assigns it to the group *grp*. For example, let us consider the following inclination interval and the associated semantic label : $\{([10°, 20°], \textit{gentle slope}), ([20°, 25°], \textit{steep slope})\}$. Such a customised specification associates the semantic label "*gentle slope*" to inclination values included in the interval [10°, 20°] and the semantic label "*steep slope*" to inclination values included in the interval [20°, 25°].

The *automated approach* only relies on a list of semantic shape labels representing the slope qualifications. Let N be the number of elements of this list, and T be the total number of groups obtained by the geometric abstraction algorithm. First, the automated discretisation orders groups based on their terrain inclination. Then, it iterates over the ordered groups and associates a uniform number of groups, T/N, to each semantic label from the *semantic set*, each T/N processed groups. For example, let us consider the following semantic slope labels: $\{gentle, medium, steep\}$, and an ordered set S of groups denoted as follows: $S = \{gr_i | i \in \{1,2,..,6\}\}$ with the following respective slope values: {5°, 10°, 15°, 20°, 25°, 30°}. For every 2 groups (as $T = 6$ and $N = 3$, $\frac{T}{N} = 2$), the automated discretisation assigns a new semantic slope label: $\{gentle, gentle, medium, medium, steep, steep\}$.

Let us compare these two discretisation approaches. On the one hand, the *custumized discretisation process* allows one to freely specify the qualification of the slopes, choosing ranges that match the problem domain. However, qualifications resulting from such a flexible approach deeply rely on the correctness of the interval bounds' values. Therefore, the customised discretisation method requires to have a good knowledge of the terrain characteristics in order to guarantee a valid specification of inclination intervals. On the other hand, the *automated discretisation process* is also able to qualify slopes without the need to spec-

ify interval bounds. This method also guarantees that all the specified semantic attributes will be assigned to the groups without a prior knowledge of the environment characteristics. However, the resulting intervals may have no relation to the problem domain.

7.2.4 Improving the geometric abstraction

Thanks to the extraction of slope semantics, terrain shape is qualified using semantic attributes and associated with groups and with their cells. Because of the nature of the classification intervals, adjacent groups with different area-weighted normal vectors may obtain the same semantic slope label. In order to improve the results provided by the geometric abstraction, we propose a process that merges adjacent groups which share the same semantic slope. This process starts by iterating over groups. Every time it finds a set of adjacent groups sharing an identical semantic slope, it creates a new group. Next, cells composing the adjacent groups are registered as members of the new group. Finally, the area-weighted normal vector is computed for the new group. Hence, this process guarantees that every group is only surrounded by groups which have different semantic slopes.

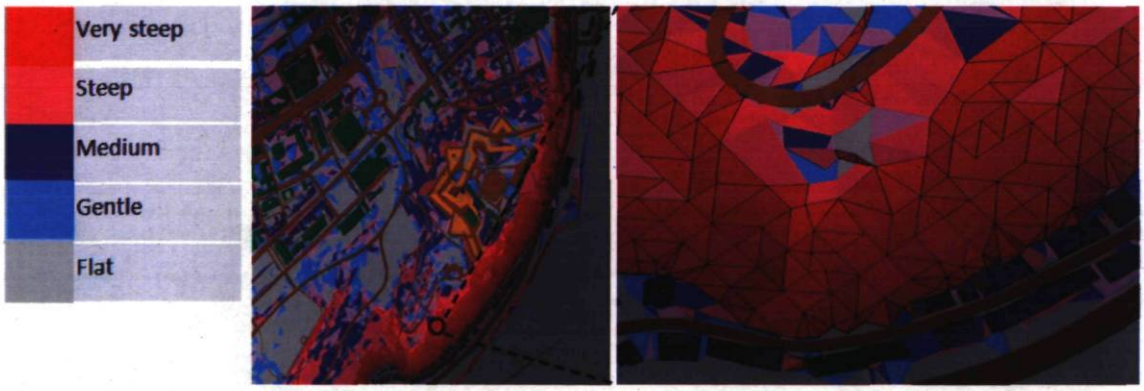


Figure 7.4: Extraction of slope semantics. The figure at the left illustrates an overview of the initial geometric abstraction, and the zoomed-in figure on the right shows adjacent groups with identical semantic slopes.

In this section, we proposed a geometric abstraction process as well as three heuristics which take advantage of this process. The geometric abstraction is built using a graph traversal algorithm. It groups cells based on their area-weighted normal vectors. The objectives of the geometric abstraction are threefold. First, it qualifies the terrain shape for geographic environments to simplify spatial reasoning mechanisms. Second, it helps to identify and fix elevation anomalies in the initial GIS data. Third, it enriches the description of geographic environments by integrating terrain semantics.

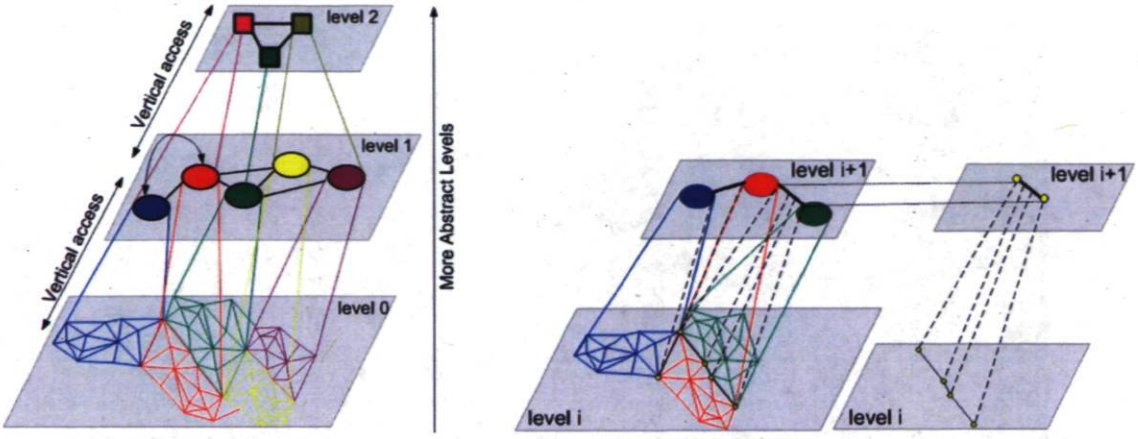
7.3 Topological abstraction for Large-Scale Geographic Environments

In Chapter 6.2, we presented our work on the generation of informed virtual geographic environments using an exact spatial decomposition scheme which subdivides the environment into convex cells organized in a topological graph structure. However, inside large scale and complex geographic environments (such as a city for example), such topological graphs can become very large. The size of such a topological graph has a direct effect on paths' computation time for path-finding. In order to optimise the performance of path computation, we need to reduce the size of the topological graph representing the IVGE. The aim of the topological abstraction is to provide a compact representation of the topological graph that is suitable for situated reasoning and enables fast path planning. However, in contrast to the geometric abstraction which only enhances the description of the IVGE with terrain semantics, the topological abstraction extends the topological graph with new layers. In each layer (except for the initial layer which is called level 0), a node corresponds to a single or a group of nodes in the immediate lower level (Figure 7.5(a)). The topological 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* in which cells are fused into groups and edges are abstracted in boundaries (Figure 7.5(b)). To do so, convex hulls are computed for every node of the topological 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 topological abstraction finally performs groupings of a set of connected nodes if and only if the group ratio is equal or close to one depending on the problem domain. Let C be the convexity rate and $CH(gr)$ be the convex hull of the polygon corresponding to gr. C is computed as follows:

$$C(gr) = \frac{Surface(gr)}{Surface(CH(gr))} \quad and \quad 0 < C(gr) \leq 1 \tag{7.2}$$

The convex property of each group's hull needs to be preserved after the topological abstraction. This ensures that an entity can move freely inside a given cell (or group of cells), and that there exists a straight path linking edges belonging to the same cell (or group of cells).

Figure 7.6 illustrates an example of the topological abstraction process and the way it reduces the number of cells representing the environment. In Figure 7.6(a), we present the initial vector format GIS data of a complex building. Figure 7.6(b) depicts the initial exact spatial decomposition (Section 6.2.2) which yields 63 triangular cells. Figure 7.6(c) presents 28 convex polygons generated by the topological abstraction algorithm. The *abstraction rate* of the number of cells representing the environment is around 55%. This rate is computed using the ratio of initial number of cells produced by the space decomposition techniques (63)



(a) Topologic abstraction and convex cells grouping.

(b) Abstraction of the cells' edges into groups' boundaries.

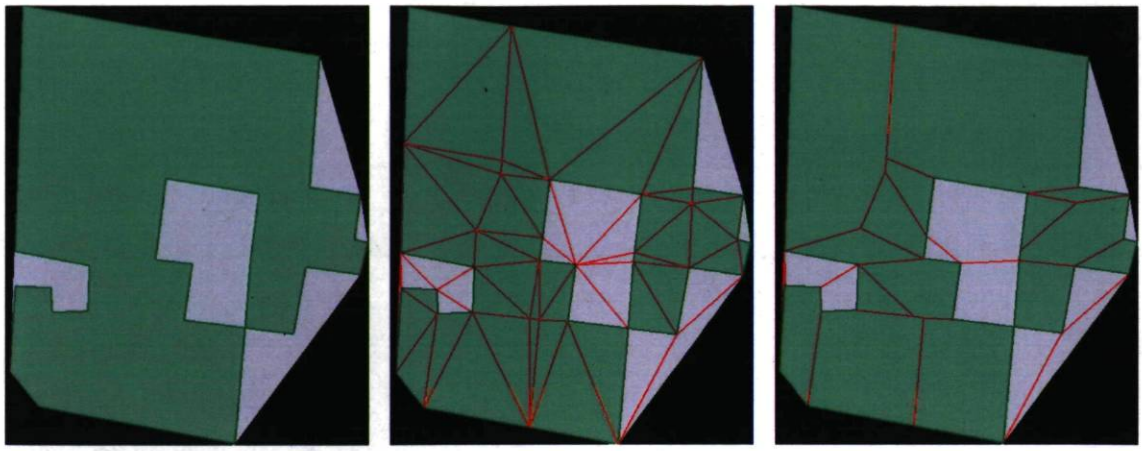
Figure 7.5: The topological graph extraction from space decomposition and extension into different levels using the topological abstraction.

by the number of convex polygons (28) obtained using the topologic abstraction technique with a convexity rate equal to 1

To conclude, we described in this section a topologic abstraction process in order to enhance the performance of the exploration of the IVGE's description. This process aims to simplify large informed graphs corresponding to large-scale and complex geographic environments. Our topologic abstraction approach reduces the number of convex cells by overlaying the informed graph with a topologically abstracted graph. The resulting IVGE is hence based on a hierarchical graph whose lowest level corresponds to the informed graph initially produced by the spatial decomposition. In the following section, we show how we use a well-known knowledge representation formalism to represent the semantic information in order to further enhance the IVGE description with respect to agents' and the environment's characteristics.

7.4 Semantic Information Representation and Propagation

Two kinds of information can be stored in the description of an IVGE. Quantitative data are stored as numerical values which are generally used to depict geometric properties (like a path's width of *2* meters) or statistical values (like a density of *2.5* persons per square meter). Qualitative data are introduced as identifiers which can range from a word with a given semantics, called a *label*, to a reference to an external database or to a specific knowledge representation. Such semantic information can be used to qualify an area (like a *road* or a



(a) (b) (c)

Figure 7.6: Illustration of the topological abstraction process with a strict convex property ($C(gr) = 1$); (a) the GIS data of a complex building; (b) the exact space decomposition using CDT techniques (63 triangular cells) ; (c) the topological abstraction (28 convex polygons)

building) or to interpret a quantitative value (like a *narrow* passage or a *crowded* place). An advantage of interpreting quantitative data is to reduce a potentially infinite set of inputs to a discrete set of values, which is particularly useful to condense information in successive abstraction levels to be used for reasoning purposes. Furthermore, the semantic information enhances the description of the IVGE, which in turn extends the agents' knowledge about their environment. However, the integration of the semantic information raises the issue of its representation. Therefore, we need a standard formalism that allows for precisely representing the semantic information which qualifies space and which is computationally tractable in order to be used by spatial reasoning algorithms used by agents.

In the following sub-section, we present the *Conceptual Graphs* (CGs) formalism which we use to represent the semantic information in the IVGE. Next, we introduce the semantic abstraction process which takes advantage of both the CGs representation and the hierarchical topological graph. This semantic abstraction aims to extract a specialised view of the IVGE which takes into account the characteristics of the involved agent archetypes, the performed actions, and the locations where these actions occur. Figure 7.7 presents the way we extend our IVGE model in order to introduce spatial semantics represented by the CGs formalism in order to support spatial reasoning algorithms.

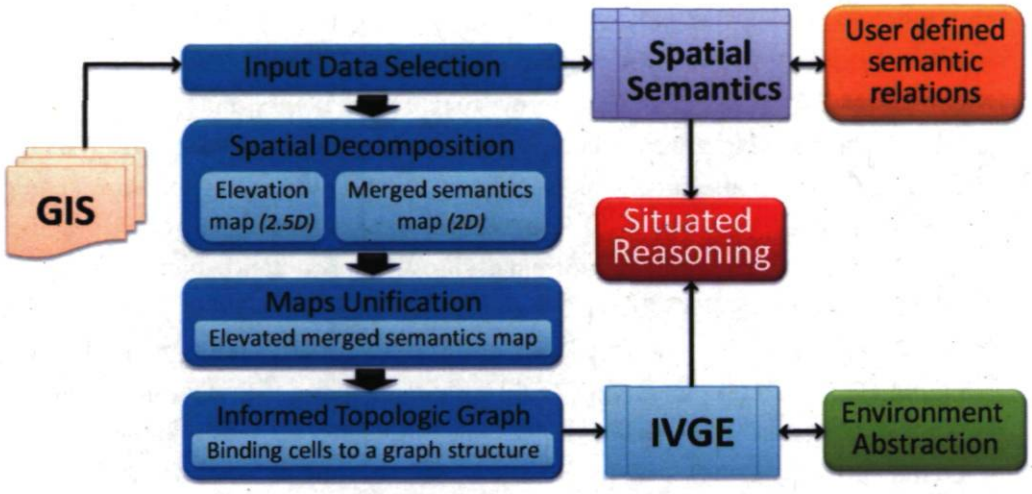


Figure 7.7: Global architecture for IVGE management: in blue, GIS data extraction and associated processes; in green, geometric, topologic and semantic abstraction processes of IVGE; and in orange used-defined spatial semantics represented using CGs.

7.4.1 Representation of semantic information using conceptual graphs

Several knowledge representation techniques can be used to structure semantic information and to represent knowledge in general such as *frames* [Min74], *rules* [LSV95] (also called *If-Then* rules), *tagging* [Var10], and *semantic networks* [Sow87] which have originated from theories of human information processing. Since knowledge is used to achieve intelligent behavior, the fundamental goal of knowledge representation is to represent knowledge in a manner that facilitates inferencing (i.e. drawing conclusions) from knowledge. In order to select a knowledge representation (and a knowledge representation system to logically interpret sentences in order to derive inferences from them), we have to consider the expressivity of the knowledge representation. The more expressive a knowledge representation technique is, the easier (and more compact) we can describe and qualify geographic features which characterise IVGE. Various artificial languages and notations have been proposed to represent knowledge. They are typically based on logic and mathematics, and can be easily parsed for machine processing. However, Sowas's *Conceptual Graphs* [Sow84] are widely considered an advanced standard[1] logical notation for logic based on existential graphs proposed by Charles Sanders Peirce and on semantic networks.

We chose to use CGs in order to represent spatial semantic information for several reasons. First, CGs are known to express meaning in a form that is logically precise and com-

[1]Conceptual Graph Interchange Format (CGIF) has been standardized in the ISO standard for Common Logic (ISO 24707) (2007).

putationally tractable [Sow84, Sow99]. Second, CGs are proved to be efficient for spatial semantic information (spatial semantics) and geographic knowledge representation [KKK04, KKK05, HM07]. Third, with their direct mapping to natural language, CGs serve as an intermediate language to translate computer-oriented formalisms to and from natural language. This will help MAGS designers and analysts express and interpret spatial semantics and space qualification in a formalism that is close to natural language. With their graphic representation, CGs serve as a readable, but formal design and specification language. CGs are basically graphs which are analogous to the description of our IVGE model. As a consequence, nodes from CGs may be easily associated with nodes of the IVGE's description. Exploring and manipulating CGs also take advantage of the efficiency of algorithms from graph theory. Fourth, CGs provide extensible means to capture and represent real-world knowledge and have been implemented in a variety of applications for information retrieval [HD04], natural language processing [CC01], and qualitative simulations [HM07]. However, our proposal to use CGs to represent spatial semantics in virtual geographic environments and to reason about them is an innovative issue. Indeed, virtual geographic environments integrating semantic information expressed using a standard formalism did not exist.

Syntactically, a conceptual graph is a network of concept nodes linked by relation nodes. Concept nodes are represented by the notation *[Concept Type: Concept instance]* and relation nodes by *(Relationship-Name)*. A concept instance can be either a value, a set of values or even a CG. The formalism can be represented in either graphical or character-based notations. In the graphical notation, concepts are represented by rectangles, relations by circles and the links between concept nodes and relation nodes by arrows. The most abstract concept type is called the *universal type* (or simply *Universal*) denoted by the symbol $\perp$.

A MAGS usually involves a large number of situated agents of different types (human, animal, static, mobile, etc.) performing various actions (moving, perceiving, etc.) in virtual geographic spaces of various extents. Using CGs greatly simplifies the representation of complex situated interactions occurring at different locations and involving various agents of different types. In order to create models for MAGS we consider three fundamental abstract concepts: 1) *agents*; 2) *actions*; and 3) *locations*. Taking advantage of the abstraction capabilities of the CGs formalism (through the *Concept Type Lattice* (CTL)[2]) instead of representing different situated interactions of various agents in distinct locations, we are able to represent *abstract actions* performed by *agent archetypes* in *abstract locations*. Moreover, we first need to specify and characterise each of the abstract concepts. The concept type lattice enables us to specialise each abstract concept in order to represent situated behaviours such as path planning of agents in space. Figure 7.8 presents the first level of the concept type lattice refining the *agent*, *action*, and *location* concepts. Figures 7.9(a),7.9(b), and 7.9(c) present the

[2]Concept types are organised in a hierarchy according to levels of generality. However, this hierarchy is not a tree, since some concept types may have more than one immediate supertype [Sow99].

expansion of the concept type lattice presented in Figure 7.8. Figure 7.9(a) illustrates some situated actions that can be performed by agents in the IVGE such as *sailing* for maritime vehicles, *rolling* for terrestrial vehicles, *walking* for humans, and *accessing* for humans to enter or exit buildings (we assume that buildings are not navigable locations from the perspective of outdoor navigation). 7.9(b) depicts how the *location* concept may be specialized into *Navigable* and *Not Navigable* concepts. The *Navigable* concept may also be specialised into *Terrestrial Vehicle Navigable*, *Pedestrian Navigable*, *Marine Vehicle Navigable*, and *Bike Navigable* which are dedicated navigable areas with respect to agent archetypes and environmental characteristics as specified by the *elementary semantics*. Figure 7.9(c) illustrates a few agent archetypes that are relevant to our geo-simulation including *pedestrians*, *cars*, *trucks*, and *bikes*.

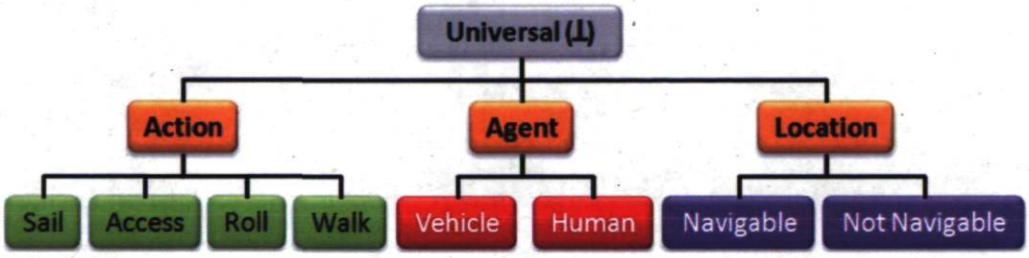
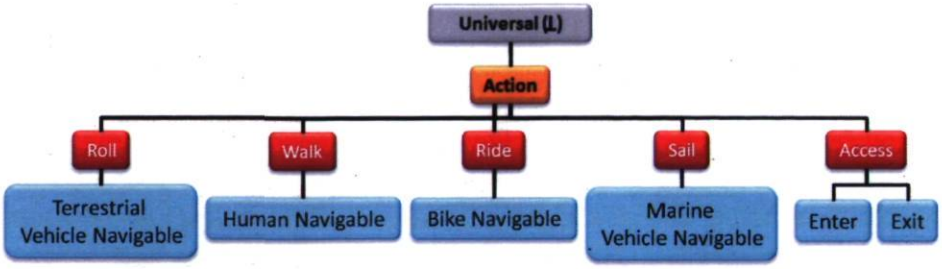


Figure 7.8: Illustration of the *action*, *agent* and *location* concepts using a concept type lattice.

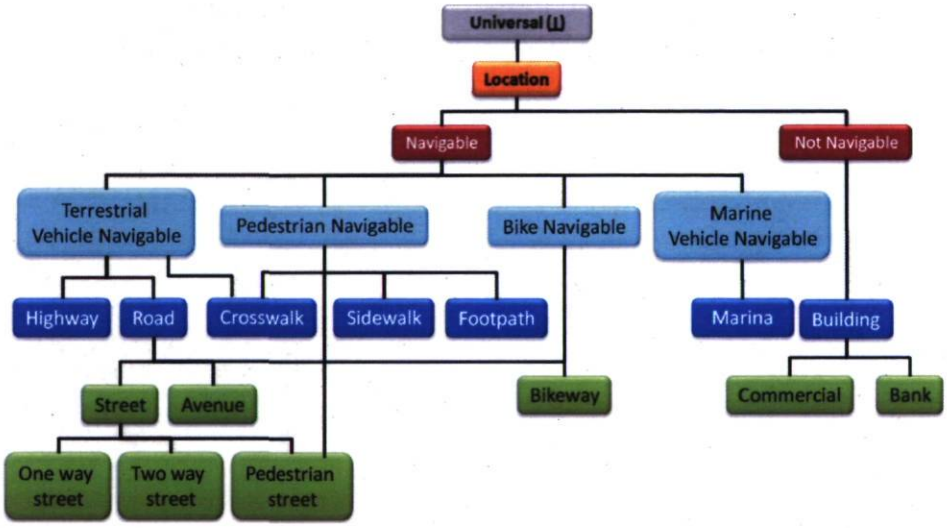
In order to show how powerful such a representation may be, let us consider the following example. We want to build a MAGS simulating the navigation of three human agents (a man, a woman, and a child), two bike riders (a man and a woman), and three vehicles (a car, a bus, and a boat) in a coastal city. The navigation behaviours of these different agent archetypes must respect the following constraints (or rules): 1) *pedestrian* agents can only move on *sidewalks*, on *pedestrian streets*, and eventually on *crosswalks* if needed; 2) *vehicles* can move on *roads* and *highways*; 3) *boats sail* on the *river* and stop at the *harbour port*; and 4) *bikes* move on *bikeways*, *roads*, and *streets* but not on *pedestrian streets*. Using standard programming languages, it might be difficult to represent or develop the functions related to such simple navigation rules which take into account both the agents' and the locations' characteristics. However, the representation of these navigation rules becomes an easy task when using CGs and our defined concept type lattice. Here are their expressions in CGs:

[PEDESTRIAN:*p]<-(agnt)<-[WALK:*w1]->(loc)->[PEDESTRIAN NAVIGABLE:*pn]
[VEHICLE:*v]<-(agnt)<-[ROLL:*r1]->(loc)->[TERRESTRIAL NAVIGABLE:tn]
[BOAT:*bo]<-(agnt)<-[SAIL:*s1]->(loc)->[MARINE NAVIGABLE:*mn]
[BIKE:*bi]<-(agnt)<-[RIDE:*r2]->(loc)->[BIKE NAVIGABLE:*bn]

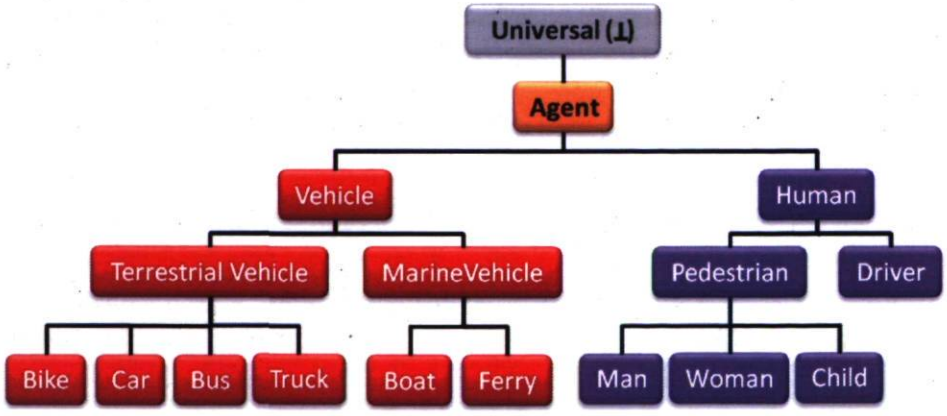
The arrows indicate the expected direction for reading the graph. For instance, the first example may be read: *an agent *p which is a "pedestrian" walks on a location *pn which*



(a) A specialisation of the concept *action*.



(b) A specialisation of the concept *location* including the *input semantics* (dark blue).



(c) A specialisation of the concept *agent*.

Figure 7.9: an example of a conceptual description of agents archetypes, actions performed, and locations situated in a geographic environment

is "pedestrian navigable". Since this expression involves the concepts *Pedestrian*, *Walk* and *Pedestrian Navigable*, this rule remains valid for every sub-type of these concepts. Therefore, thanks to CGs and the concept type lattice, there is no need to specify the navigation

rules for men, women, and children if they act as pedestrians in locations such as *pedestrian streets*, *sidewalks*, or *crosswalk*. Indeed, these agent archetypes are subtypes of the *Pedestrian* concept and *pedestrian streets*, *sidewalks*, and *crosswalks* are subtypes of the *Pedestrian Navigable* concept. To conclude, CGs offer a powerful formalism to easily describe different concepts involved in MAGS including agents, actions, and environments. It is important to mention that only elementary semantics are spatially situated in our IVGE. The remaining abstract and specialised semantics are conceptual semantics used by the semantic abstraction process described in Section 7.4.3.

7.4.2 Propagation of input semantics

The *IVGE* obtained as a result of the *GIS* importation technique emphasizes qualified areas by defining the semantics of their boundaries. But these informed boundaries are difficult to exploit when dealing with the semantics associated with a position, as for example if we want to check if a point location is inside a building. This is why we propose to enhance the information provided by the *IVGE* by spreading the boundaries' semantics to the cells. Three related processes are necessary, and are explained in the following subsections: *graph analysis*, *resolution of potential conflicts*, and *semantic assignment*.

Graph analysis

The graph analysis is a traversal algorithm which explores the environmental graph while qualifying the cells with a given semantic label. This algorithm is applied to the entire graph for each semantic label to be propagated. While exploring the graph, the algorithm collects three kinds of cells which are stored in three container structures for future use: *Inside* cells are within an area delimited by borders associated with the propagated semantic label. *Outside* cells are outside any area defined by the propagated semantic label. *Conflict* cells are qualified as both inside and outside by the algorithm.

Three parameters influence the traversal: 1) the semantic label *sem* to propagate. 2) a set of *starting cells*, indicating where to start the exploration of the graph; a set is provided instead of a single cell in order to manage disconnected graphs. 3) a boolean value $start_{in}$ indicating whether the semantic must be assigned to the starting cells or not. Let C and NC be two collections of cells representing the *Inside* and the *Outside* of a geographic feature. The recursive algorithm is composed of four steps:

- **Step 1:** For each cell c in *starting cells*, proceed to *step 2*;

- **Step 2:** If $start_{in}$ let C be a collection of cells called *Inside* and NC be a collection of cells called *Outside*; else let C be *Outside* and NC be *Inside*.
  Proceed to *step 3*.
- **Step 3:** If c is in NC, transfer c to $Conflict$. If c is in $Conflict$ or in C, Proceed to *step 1*.
  Put c in C, then for each cell n neighbouring c through a border b proceed to *step 4*.
- **Step 4:** If *sem* is defined for border b, invert C and NC. Proceed to *step 3* with n being c.

The graph analysis (see *Algorithm 2* below) iterates through the starting cells (line 1) triggering the graph traversal (lines 3 and 5). The graph traversal (see *Algorithm 3* below) recursively explores the graph while collecting the three kinds of cells (inside, outside, and conflicting). Line 1 checks if the current cell has not already been processed. Then, line 2 detects conflicting cells, i.e. cells which have already been assigned to the other container. If so, lines 3 and 4 assign the cell to the conflict container, and stop the recursion. Otherwise (line 5), the cell is assigned to the current container (line 6), and the cell's neighbours are explored (line 7). Finally, line 10 checks if a border with the propagated semantic is traversed, which results in a swap of the filled containers for the recursive call (line 11), or in a standard recursion otherwise (line 13).

Algorithm 2 GraphAnalysis

PARAMETERS: ***crt_sem*** the propagated semantic label; ***start_cells*** the starting cells; ***start_inside*** defining if starting cells are inside or outside.

REQUIRE: ***inside_cells*** the container of inside cells; ***outside_cells*** the container of outside cells.

```
1: FOR ALL cell ∈ start_cells DO
2:     IF start_inside THEN
3:         TRAVERSE (cell, inside_cells, outside_cells)
4:     ELSE
5:         TRAVERSE (cell, outside_cells, inside_cells)
6:     END IF
7: END FOR
```

Resolution of Conflicts

After each graph traversal, we must deal with the cells that are potentially in conflict. Indeed, these cells must be assigned to either the *Inside* or the *Outside* container so that the system can continue with the next step. Cells are in conflict when the shapes of two input features with the same semantic label share a segment. Two alternative methods are proposed: 1)

Algorithm 3 Traverse

PARAMETERS: ***crt_cell*** the current cell; ***crt_cont*** the current cell container; ***oth_cont*** the other container.

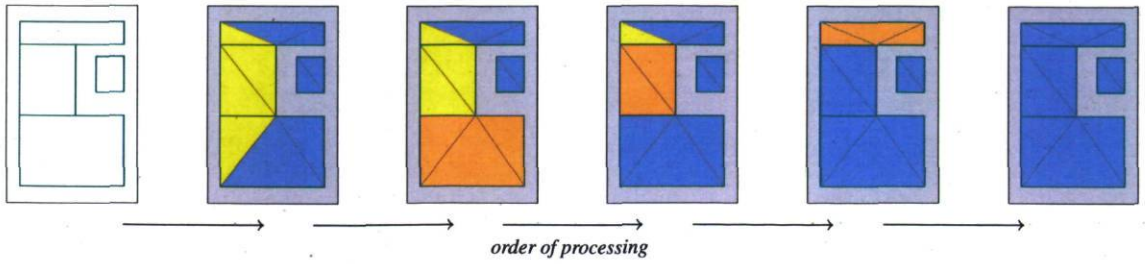
REQUIRE: ***crt_sem*** the propagated semantic label; ***conflict_cells*** the container of conflicting cells.

```
1: IF crt_cell ∉ crt_cont ∧ crt_cell ∉ conflict_cells THEN
2:     IF crt_cell ∈ oth_cont THEN
3:         oth_cont ← oth_cont \ crt_cell
4:         conflict_cells ← conflict_cells ⋃ {crt_cell}
5:     ELSE
6:         crt_cont ← crt_cont ⋃ {crt_cell}
7:         FOR i = 0 to NEIGHBOURSNUMBER (crt_cell) DO
8:             nxt_border ← BORDER (crt_cell, i)
9:             nxt_cell ← NEIGHBOUR (crt_cell, i)
10:            IF crt_sem ∈ SEMANTICSOF (nxt_border) THEN
11:                TRAVERSE (nxt_cell, oth_cont, crt_cont)
12:            ELSE
13:                TRAVERSE (nxt_cell, crt_cont, oth_cont)
14:            END IF
15:        END FOR
16:    END IF
17: END IF
```

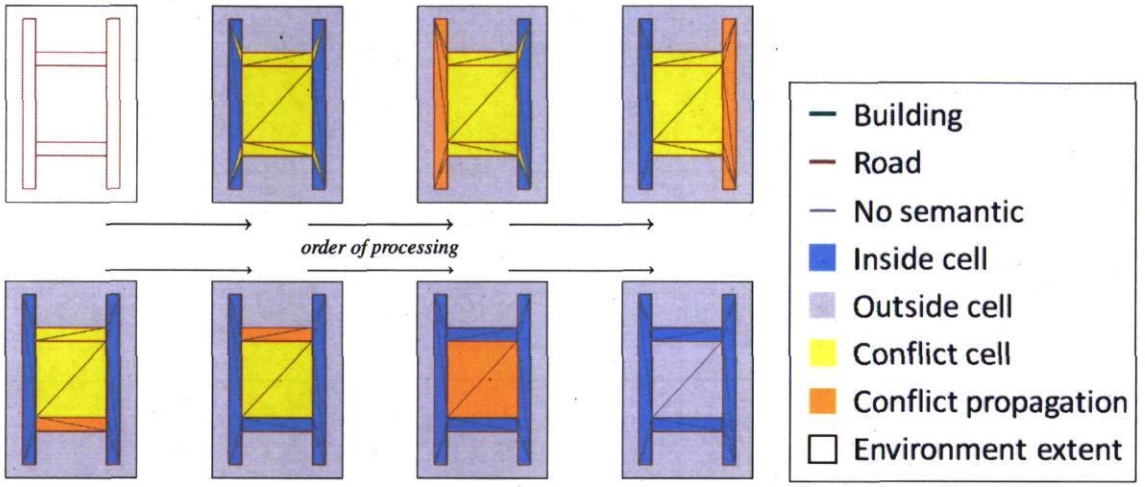
a fast assignment where the conflicting cells are arbitrarily transferred to one of the target containers, and 2) a deductive assignment where an algorithm selects the best option based on geometric considerations.

The arbitrary assignment is used when the internal details of a shape are not relevant to the target application. For example, Figure 7.10(a) presents two buildings. The building on the left-hand side has internal walls producing conflicting cells. These conflicts can be resolved using the arbitrary assignment since they are not relevant when considering a building as an obstacle. The deductive assignment is used when the internal details of a shape are relevant. For example, Figure 7.10(b) presents four connected roads producing conflicting cells. They are resolved thanks to the deductive method in order to determine which parts are effectively a road, and which parts are not. Both methods are carried out in two steps: 1) a local conflicting graph extraction which is the same for both methods, and 2) a decision step which is specific to each method.

The *local conflicting graph extraction* collects all the cells surrounding a conflicting cell, but only if they are reachable through a border which is not marked by the propagated label.



(a) Buildings with conflicts resolved by *arbitrary inside* assignment.



(b) Roads with conflicts resolved by *deductive* assignment.

Figure 7.10: Two examples of conflicting cells resolution, where the left most figure is the original environment, the second figure is the graph analysis, and the last figure is obtained after the resolution of all conflicts.

Each orange zone in Figure 7.10 shows an extracted local conflicting graph. Every time a cell is discovered, it is transfered from the global container *Conflict* to a local container. Then, the algorithm recursively explores the neighbours which are reachable through a border which is not marked by the propagated label, transferring them to the local container. At the end, the algorithm obtains a set of local conflict containers, corresponding to the amount of local graphs considered in conflict.

The *decision* part of the arbitrary assignment only consists of transferring the local conflicting cells to one of the *Inside* or the *Outside* containers. The decision part of the deduction algorithm is based on geometric considerations. If the local conflicting zone is mainly surrounded by *Outside* cells, then the conflict is resolved as *Inside*, and vice-versa. In order to check the surrounding of a conflicting zone, three perimeter lengths are computed: the inside perimeter P_{in} which is the total length of the zone's boundaries connected to *Inside* cells; the outside perimeter P_{out} for *Outside* cells; and the unknown perimeter P_{ukn} when connected to *Conflict* cells. Then, three cases can occur:

1. If $P_{in} > P_{out} + P_{ukn}$ the conflicting cells are assigned to *Outside*;
2. If $P_{out} > P_{in} + P_{ukn}$ the conflicting cells are assigned to *Inside*;
3. Otherwise, neither the inside or outside perimeters are considered significant and the decision will be re-evaluated during a subsequent step.

The third case requires a second decision step for all undecided conflicts, where the unknown perimeter is ignored in order to force a decision. The need for two decision steps is illustrated in Figure 7.10(b): if the last conflict resolution had been made first then the decision would have been impossible, since the zone is totally surrounded by unknown cells; the displayed resolution order avoids any decision failure, only requiring one step.

The *local conflicting graph extraction* (see *Algorithm 4* below) collects all the cells surrounding a conflicting cell, but only if they are reachable through a border which is not marked by the propagated label. Each orange zone in Figure 7.10 shows an extracted local conflicting graph. Every time a cell is discovered, it is removed from the global list of conflicting cells (line 2), and added to the local list (line 3). Then, the algorithm recursively explores the neighbours (line 4) which are reachable through a border which is not marked by the propagated label (line 6).

Algorithm 4 ExtractLocalConflict

PARAMETERS: ***crt_cell*** the current conflicting cell.
REQUIRE: ***crt_sem*** the processed semantic; ***conflict_cells*** the container of conflicting cells; ***local_conflict_cells*** the local graph of conflicting cells.

```
1:  IF crt_cell ∉ local_conflict_cells THEN
2:      conflict_cells ← conflict_cells \ crt_cell
3:      local_conflict_cells ← local_conflict_cells ⋃ {crt_cell}
4:      FOR i = 0 to NEIGHBOURSNUMBER (crt_cell) DO
5:          nxt_border ← BORDER (crt_cell, i)
6:          IF crt_sem ∉ SEMANTICSOF (nxt_border) THEN
7:              nxt_cell ← NEIGHBOUR (crt_cell, i)
8:              EXTRACTLOCALCONFLICT (nxt_cell)
9:          END IF
10:     END FOR
11: END IF
```

The decision part of the arbitrary assignment only consists of transferring the local conflicting cells to one of the *inside* or the *outside* containers. In contrast to the arbitrary assignment, the decision part of the deductive algorithm is based on geometric considerations (see *Algorithm 5* below). If the local conflicting zone is mainly surrounded by outside cells, then

the conflict is inside, and vice versa. This decision process is carried out in two steps (the *imposed* parameter being set to *false*, and then to *true*). First, the conflict cells can remain in an unknown state, no decision being taken if the algorithm is not conclusive. Second, a decision is imposed even if inconclusive.

Three perimeter lengths are computed (lines $1-18$): *inside perimeter* which corresponds to the zone boundaries connected to inside cells; *outside perimeter* for outside cells; and *unknown perimeter* when connected to cells which are still in conflict. If a decision is imposed, the unknown perimeter is ignored (lines $19-21$). Finally, if the inside perimeter is predominant (lines $22-24$), the conflicts are set to outside cells, or to inside cells in the other case (lines $25-27$). The two successive conditions (lines 22 and 25) allow the first step of the algorithm to maintain the conflicting state, waiting for other resolutions. Finally, the second step is done for all undecided local conflicts once all the first decision steps have been carried out. The need for two decision steps is illustrated in Figure 7.10(b). If the last conflict resolution in this figure had been made first, the decision would have been impossible, the zone being totally surrounded by unknown cells; the displayed resolution order avoids any decision failure, and only requires one step.

Semantic Assignment

The last step of the semantic propagation consists of assigning the final semantic labels to the cells. The process is quite simple: each propagated label is assigned to all the corresponding *Inside* cells. One can notice that some cells may have multiple semantics when they are present in more than one *Inside* container. Additionally, it is possible to keep track of the *Outside* cells by assigning them a *negative* semantic, as for example in order to know what is *not a road* in the environment. Finally, an optional process can be performed to remove the semantics from the borders of some detected conflicting cells. Indeed, such borders may distort some spatial reasoning algorithms. For example, when considering road borders as obstacles to plan a path, a simulated vehicle would not be able to go through some passageways (i.e. left hand side of Figure 7.11(b)). After resolution, the semantic of the problematic borders is removed, making them crossable (i.e. right hand side of Figure 7.11(b)). These problematic borders are the ones that are marked with a propagated label and which connect two *Inside* cells. One can note that only the cells previously detected as conflicting need to be tested.

Algorithm 5 DeduceConflict

PARAMETERS: ***local_conflict_cells*** the local graph of conflicting cells; ***imposed*** indicates if a choice must be forced or if the local conflict can remain undecided.

REQUIRE: ***crt_sem*** the processed semantic; ***inside_cells*** the container of inside cells; ***outside_cells*** the container of outside cells.

```
1: inside_perimeter ← 0
2: outside_perimeter ← 0
3: unknown_perimeter ← 0
4: FOR ALL cell ∈ local_conflict_cells DO
5:     FOR i = 0 to NEIGHBOURSNUMBER (cell) DO
6:         nxt_border ← BORDER (crt_cell, i)
7:         IF crt_sem ∈ SEMANTICSOF (nxt_border) THEN
8:             nxt_cell ← NEIGHBOUR (crt_cell, i)
9:             IF nxt_cell ∈ inside_cells THEN
10:                inside_perimeter ← inside_perimeter + LENGTH(nxt_border)
11:            ELSE IF nxt_cell ∈ outside_cells THEN
12:                outside_perimeter ← outside_perimeter + LENGTH(nxt_border)
13:            ELSE
14:                unknown_perimeter ← unknown_perimeter + LENGTH(nxt_border)
15:            END IF
16:        END IF
17:    END FOR
18: END FOR
19: IF imposed THEN
20:    unknown_perimeter ← 0
21: END IF
22: IF inside_perimeter > outside_perimeter + unknown_perimeter THEN
23:    outside_cells ← outside_cells ⋃ local_conflict_cells
24:    local_conflict_cells ← ∅
25: ELSE IF outside_perimeter ≥ inside_perimeter + unknown_perimeter THEN
26:    inside_cells ← inside_cells ⋃ local_conflict_cells
27:    local_conflict_cells ← ∅
28: END IF
```

7.4.3 Semantic abstraction

Depending on their characteristics and goals, agents can interact differently with a given environment. For example, the spatial semantics associated with each cell can be used by

(a) Buildings. (b) Roads.

Figure 7.11: Two examples of the result of semantic propagation on borders. In both figures: left is the original environment, centre is obtained after conflict resolution, right is obtained after assigning semantics and filtering conflict borders.

agents to navigate in favoured areas; a pedestrian might avoid navigation in areas such as roads while preferring to walk on sidewalks, crossings, etc. The aim of the semantic abstraction is to provide a semantic representation of the topological graph for each agent archetype (e.g. a human and a car). We call such a representation an *abstract view*. While navigating, agents can refine this *abstract view* by querying local or global refinements on cell types or on graph structures. The semantic abstraction provides multi-level views of the IVGE in which agents are navigating. A specific view can be generated, using the semantic abstraction, for each agent archetype, depending on its goals and needs. In this way, an agent can retrieve abstract views of the environment containing relevant information for its decision making. For example, an agent simulating a virtual human navigating in a classic VGE would make no distinction between a road and a sidewalk since both of them are locations in the virtual environment. However, thanks to our semantic abstraction, this agent will be able to adapt its behaviour with respect to the environment's characteristics in order to "*speed up when crossing the road*" or to "*cross the street using crosswalk areas*"

In order to build such a semantic abstraction, we first define a semantic type hierarchy as well as a set of semantic expressions (rules) linking various semantic concepts as detailed in Section 7.4.1. The semantic type hierarchy describes the supertype/subtype links between semantic concepts. Figure 7.9(b) shows an example of a semantic type hierarchy for the concept *location*. This concept type lattice includes various semantic concepts such as "*road*" which is a subtype of "*Terrestrial Vehicle Navigable*" and a supertype of "*street*". Figure 7.12(a) shows a virtual environment for a part of Quebec City and Figure 7.13(a) presents a the corresponding informed graph involving semantics summarised in Figure 7.12(b) such as *sidewalk*, *crosswalk*, *one-way street*, *two-way street*, *pedestrian street*, *bikeway*, etc. Taking advantage of our concept type lattice and of the graph structure of the IVGE, we are able to semantically abstract the informed topological graph and to build a new semantic representation of the environment. This new semantic graph overlays the topological graph, which enables the creation of a hierarchical graph structure of the IVGE useful for situated reasoning such as path planning.

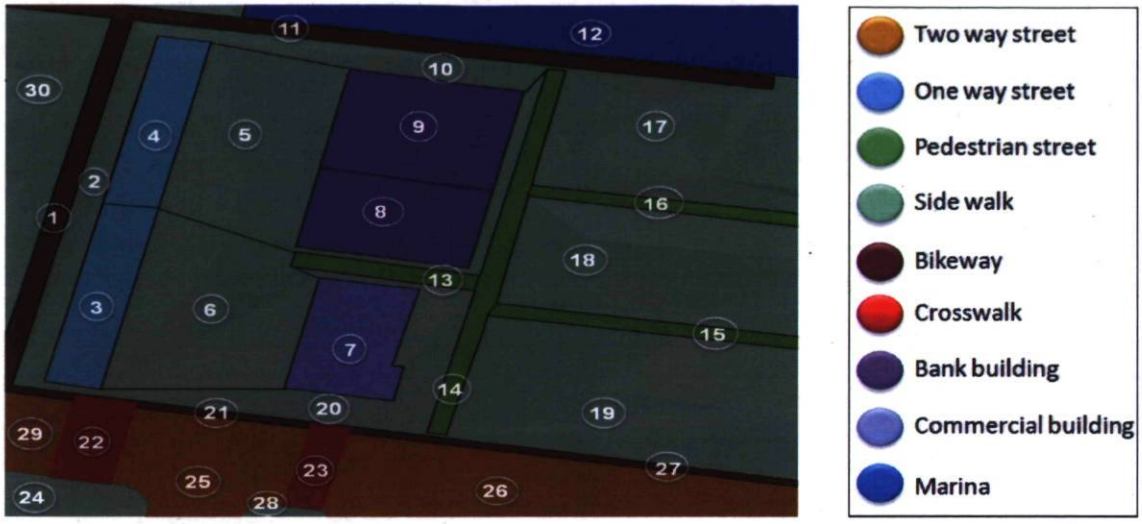
7.5 Case Study: Semantic Abstraction of Quebec City

In this section, we present the results of the IVGE enhancement presented above. First, we illustrate the semantic abstraction of an urban environment representing a part of Quebec City, Quebec, Canada. Second, we propose a holonic-based approach which uses our automated IVGE generation approach (Section 6.2) and combines both topologic abstraction (Section 7.3) and semantic abstraction (Section 7.4.3) into a single formal model. Third, Section 7.7 presents IVGE-Viewer, a software application for the visualization, spatial analysis, and manipulation of IVGE.

In order to explain the semantic abstraction process, let us consider the virtual environment depicted in Figure 7.12 which represents a part of *Quebec City*. Figure 7.13(a) illustrates the informed topological graph associated with this geographic area. The first level of semantic abstraction consists of grouping cells sharing identical elementary semantics as illustrated in Figure 7.13(b). This first abstraction level only simplifies the environment structure by reducing the number of cells. The semantic concept type diagram is then used to guide the computation of the second level of semantic abstraction. The second level of semantic abstraction takes into account the characteristics of a specific category of situated actions occurring in a specific location and involving a specific type of agents. For example, in Figure 7.14(a), adjacent cells of type *bikeway*, *crosswalk*, *one-way street*, and *two-way street* are grouped as navigable areas for bike riders. We assume in this case study that bike riders are not allowed to ride on *sidewalks* nor on *pedestrian streets*. In addition, in Figure 7.14(b), adjacent cells of type *crosswalk*, *one-way street*, and *two-way street* are grouped as navigable areas dedicated to cars. Here, we assume that *bikeways* are prohibited areas for cars.

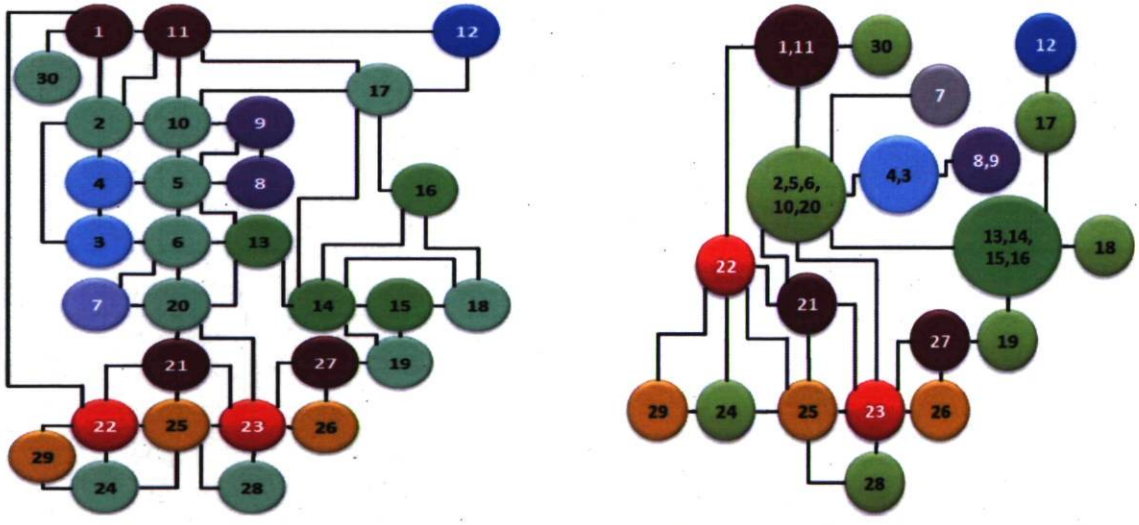
Once the geometric, topologic, and semantic abstraction levels are computed, an agent can plan a path inside a specific view of the IVGE which takes into account both the environment's and the agent's characteristics. Indeed, Figure 7.13(b) and Figure 7.14(a) illustrate the first and second levels of the semantic abstraction process while taking advantage of the semantic type lattices we proposed in Figure 7.9. The semantic abstraction allows for the creation of the hierarchical graph structure of the IVGE which is adapted to hierarchical decision processes. Such a hierarchical graph structure allows for the formulations of precise queries and local refinements and enables the description of complex reasoning processes based on the environment topology. It also supports high-level spatial reasoning processes such as hierarchical path planning that we introduce in the following section.

The initial exact spatial decomposition generates 120,833 triangles (cells) in 4.1 seconds. The geometric abstraction produces approximately 73,000 groups of cells in 2.8 seconds. Besides, the *custom* and *automated* discretisation processes are performed in 1.8 and 1.2



(a) (b)

Figure 7.12: Left: A virtual environment for a part of Quebec City situated near the marina.

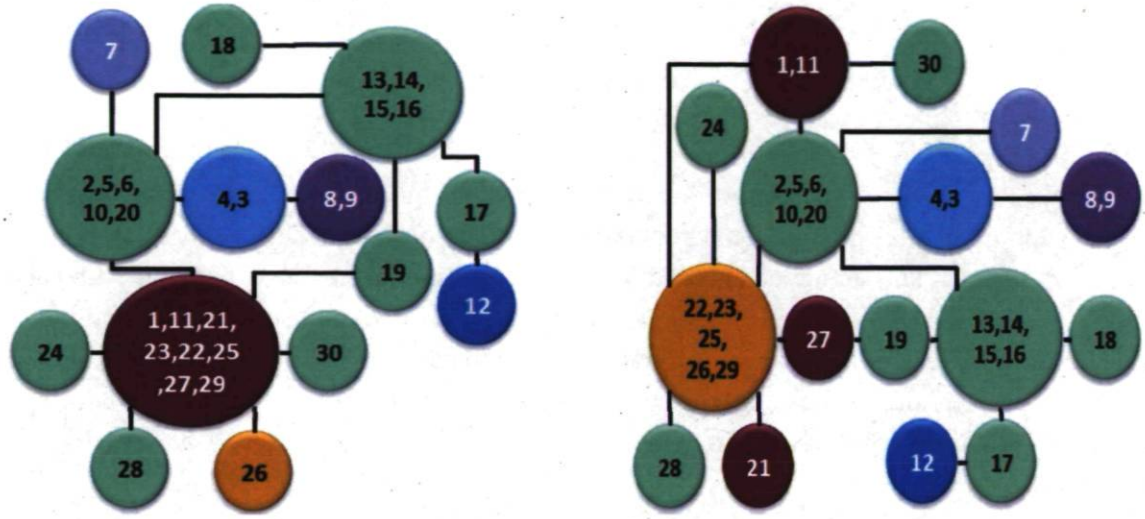


(a) Informed topological graph. (b) The first level of the semantic abstraction.

Figure 7.13: 7.13(b) is the first level of the semantic abstraction (adjacent cells sharing identical semantic information are grouped together) of the informed topological graph 7.13(a).

seconds respectively using 8 semantic slope labels.

Regarding the topological abstraction, we tested our model using two abstraction levels. The first level fully respects the convexity (i.e. convex ratio = 1) and produces around 87,000 convex groups of cells, which corresponds to a reduction of almost 28%. The second level



(a) The second level of the semantic abstraction based on the bike navigation semantic lattice defined in Figure 7.8. Note how *bikeway* nodes are grouped with adjacent *one-way street* and *two-way street* nodes.

(b) The second level of the semantic abstraction based on the car navigation semantic lattice defined in Figure 7.8. Note how *crosswalk* nodes are grouped with adjacent *one-way street* and *two-way street* nodes.

Figure 7.14: Second level of the semantic abstraction taking advantage of the conceptual type lattice and of the graph structure of the IVGE.

performs groupings of adjacent convex groups if and only if the convex ratio is better than 0.9. The second level yields approximately 28,000 cells which represent 22% of the total number of convex cells produced by the first-level.

Regarding the semantic abstraction, we used the elementary semantic concepts illustrated in Figure 6.3 together with the semantic concept type hierarchy detailed in Figure 7.8. The semantic abstraction reduced the number of abstract nodes belonging to the second level of the topological graph from 28000 to approximately 17,000. The comparison between the number of cells yield by the semantic abstraction (17,000) and the initial number of cells produced by the exact space decomposition technique (120,833) indicates a significant simplification of the number of nodes representing the IVGE. Obviously, the performance of spatial reasoning algorithms such as path planning and navigation can benefit from such a simplified description of the IVGE.

7.6 Case Study: A Holonic Approach to Model Large-Scale Geographic Environments

In this section, we propose a novel approach to model an IVGE which uses the holonic approach as a computational geographic methodology and the holarchy as an organizational principle. This approach is based on the models introduced above, namely: 1) the automated generation of IVGE (Section 6.2); the topologic abstraction (Section 7.3); and the semantic abstraction (Section 7.4.3). In order to take into account geometric, topologic, and semantic characteristics of the geographic environment, we propose the use of the holonic approach to build the environment holarchy.

7.6.1 Holonic Approach

The term "*holon*" was originally coined by Arthur Koestler [Koe67], basing it on the Greek word "*holos*" for "*whole*" and the suffix "*-on*" that denotes "part". According to Koestler, a holon is a fractal structure that is stable, coherent, and consisting of several holons acting as sub-structures. In this definition, a holon itself can be considered as part of a higher level architecture of holons. This architecture can be understood by taking the example of a human being that consists of organs which in turn consist of cells that can be further decomposed and so on. Furthermore, the human being is part of a family and a society. None of these components can be understood completely without their sub-components or without the super component of which they are parts. Several holons, each having its own identity, can exist together as components of the same system. In this system, known as a holarchy [JRMJ05], holons are autonomous (or independent) with respect to their subordinate parts and simultaneously dependent on parts of higher hierarchic levels. Thus, a holarchy denotes a hierarchical organization of holons having a recursive structure. This structure can guarantee performance stability, predictability, flexibility, adaptability, and global optimization of hierarchical control [HGL+02]. It has been used, for example, in manufacturing in order to combine top-down hierarchical organisational structure with decentralised control, which takes the bottom-up perspective [MLW+06].

7.6.2 Environment Holarchy

In order to organize this graph-based spatial structure, we use a holarchy of abstracted graphs. This holarchy is fundamentally based on the following roles: *Head* and *Part*. The *head* repre-

sents the members of its holonic organization. Its responsibilities are limited to: 1) integrating new members into its holon group, and 2) releasing members which are not relevant to the holarchy. The part responsibilities are basically: 1) remaining a member of a holarchy, and 2) seeking a holarchy to join. The environment holarchy is obtained by applying a two-phase process: 1) *grouping adjacent cells* to form *groups* of cells; and 2) merging of semantically coherent groups in order to form *zones*. The convex-cell grouping phase (phase 1) is characterised by a strict geometric constraint considering the convexity of the generated groups. During this phase, the *head* makes the decision to integrate or realise a member based on the following model:

$$Func_{Head}(h_i) = \left| \begin{array}{ll} \textbf{If } \nabla(Semantic(h_i),\ S) & \textbf{Then} \\ \max(C(Head \cup h_i), C(Head)) & \\ \textbf{Else} & \\ C(Head) & \end{array} \right.$$

$Func_{Head}(h_i)$ refers to the decision-making model of the *head* regarding the integration of the candidate holon h_i. $\nabla(S_i, S_j)$ is a *compatibility* function between semantics S_i and S_j whose evaluation is a *boolean*. C is the convexity factor and is computed as follows: $C(h_i) = surface(h_i)/surface(CH(h_i))$, and $0 < C(h_i) \leq 1$ where *surface (i)* refers to the surface area of i. $CH(h_i)$ is the convex hull of the geometric form corresponding to the holon h_i, $Semantic(h_i)$ is the semantic of the holon h_i, and S is the list of the $Head$ semantics. The $Head$ checks if the semantic information of h_i is *compatible* with its semantic information set S. If the test is conclusive, the $Head$ computes the convexity factor of its geometric form with and without the candidate $holon_i$. The integration of the holon h_i must maximise the convexity factor of the $Head$ (i.e. $C(Head \cup h_i) > C(Head)$. The same logic is applied to released members. The release of the holon h_j must optimise the convexity factor of the $Head$ (i.e. $C(Head \setminus h_j) > C(Head)$). The merging of semantically coherent groups (phase 2) is characterised by a less strict geometric constraint regarding the convexity of the generated zones and a higher priority on their semantics compatibility functions ∇. Moreover, depending on the application purposes, designers may apply this second-phase grouping strategy several times while progressively increasing the priority of the semantic grouping rules and decreasing the geometric constraint. An example of semantic grouping rules may correspond to the *crossable* relationship between groups. However, designers may freely adopt application-dependent semantic merging rules. For example, a group of cells qualified as *road* can be grouped with a group of cells qualified as *highway* in order to create a zone which enables situated agents representing cars to navigate in the IVGE. In the same way, a group of cells qualified as *sidewalk* can be grouped with a group of cells qualified as *crosswalk* in order to allow situated agents representing pedestrians that can cross roads and thus navigate in the IVGE.

7.7 IVGE-Viewer: Architecture and Implementation

The enhancement approaches presented above have been developed and integrated within a software application that extends the IVGE-Builder architecture and which allows MAGS users *to visualize*, *to manipulate*, and *to filter* IVGE (Figure 7.15). This software application is called *IVGE-Viewer*.

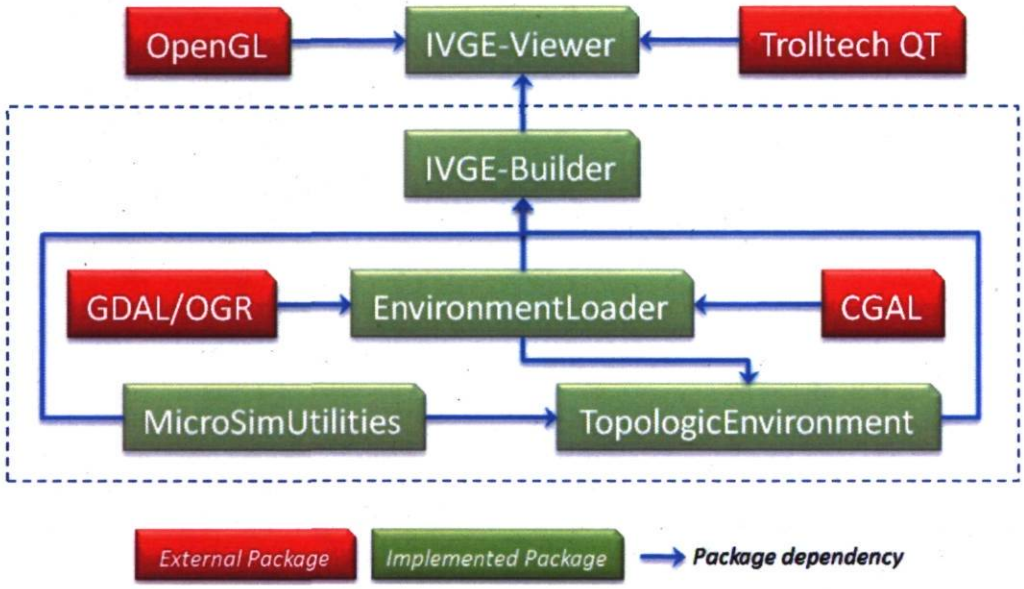


Figure 7.15: Software Architecture of IVGE-Viewer which extends *IVGE-Builder* and uses the *OpenGL* and *Trolltech QT* external packages.

7.7.1 Visualisation of IVGE

The *IVGE-Viewer* tool provides two visualization modes for IVGE. First, a 3*D* view (as shown in Figure 7.17(b)) allows the user to freely navigate in the virtual environment. An optional mode for this view constraints the camera at a given height above the ground, following the terrain variations when navigating. An upper view with orthogonal projection to represent the GIS data as a standard map is also provided. In this view the user can scroll and zoom the map (1 in Figure 7.16), and can precisely view any portion of the environment at any scale. Additionally, one can select a position in the environment in order to retrieve the corresponding data (2 in Figure 7.16), such as the underlying triangle geometry, the corresponding height and the associated semantics, including semantic slope.

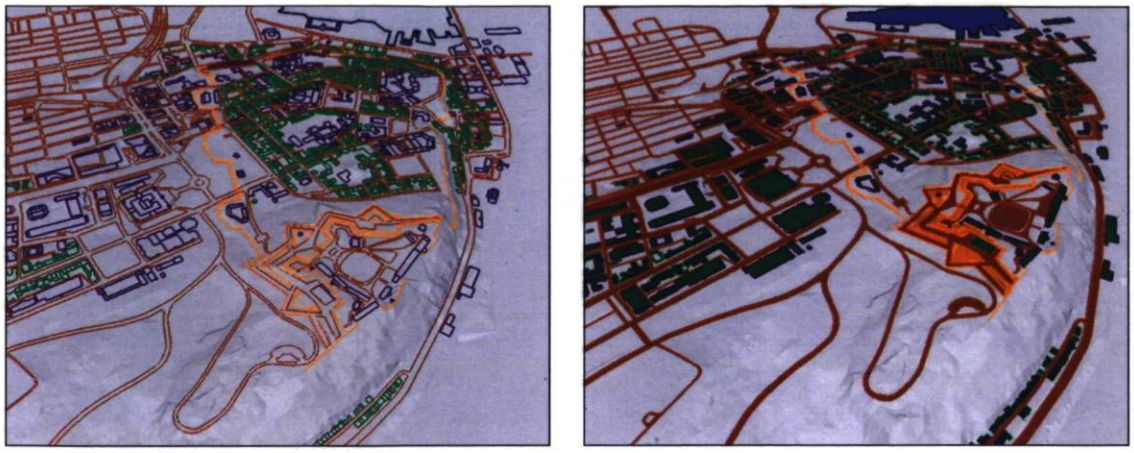
Figure 7.16: 2D map visualisation of the GIS data. (1) unified map, (2) selected position's information including geometric and semantic data.

7.7.2 Manipulation of IVGE

The IVGE-Viewer tool allows a user to load, parse, modify, and save an IVGE description in a compressed binary format. The IVGE-Viewer's architecture relies on two *C++* open-source libraries: *OpenGL* [Ang08] which renders the IVGE spatial decomposition; and *Trolltech QT* [QT09] which provides high-level user interface functionalities. As an example, the size of the IVGE description file representing Quebec City ($30km^2$) using 8 semantic layers and 1 elevation layer is only 1620KB. It also allows automatic popagation of semantic information from borders to cells. It provides a rich set of user-interface screens which assists MAGS users to manipulate IVGE.

7.7.3 Environment Instance

Another important feature provided by IVGE-Viewer is the concept of *Environment Instance*. An Environment Instance (EI) is a subset of the IVGE graph-based structure that only contains cells specified as accessible for a certain type of agents in a MAGS. The MAGS defines which semantic information should be considered as obstacles regarding a given type



(a) Borders semantics. (b) Borders and cells semantics.

Figure 7.17: Propagation of semantics from the borders of the unified map (*a*) to cells (*b*) after semantic propagation.

of agents. Then a traversal algorithm analyses the IVGE structure and extracts cells whose semantic information is not qualified as an obstacle. An EI may be thought of as a filtered view of the IVGE that is initially generated by IVGE-Builder. An environment instance corresponds to the mental view that an entity evolving within an IVGE may have about its surroundings.

7.8 Conclusion

In this chapter, we extended the IVGE model that we developed in the previous chapter and showed how we abstracted its description as illustrated in Figure 7.18. First, we described a ***geometric abstraction*** process which enriches the IVGE description with terrain semantics. Moreover, the geometric abstraction process helps to detect and filter elevation anomalies and qualifies the terrain shape, specifically slope. In order to deal with large-scale virtual geographic environments, we described a ***topologic abstraction*** which builds an hierarchical topologic graph. This hierarchical structure reduces the size of the topological graph representing the IVGE and optimizes the performance of path computation.

Regarding the representation, propagation, and exploitation of the semantic information, we use the CG formalism, which constitutes another original aspect of this work. First, we demonstrated how easily we can represent spatial semantics, how we take advantage of the concept type lattice to specify rules at the abstract concept level, and how they remain valid for more specialized concepts. Second, we outlined how the propagation of semantic

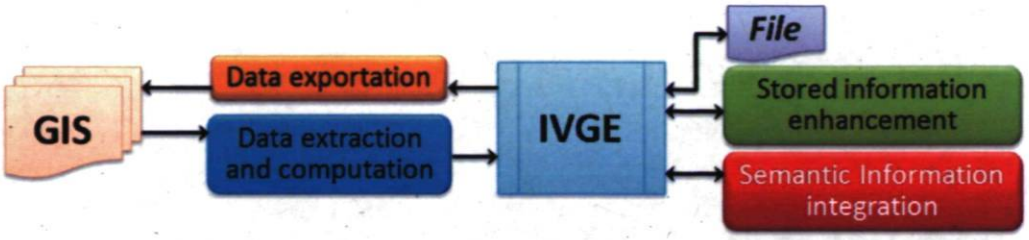


Figure 7.18: Global architecture for IVGE management: in blue, GIS data extraction and associated processes; in orange, IVGE data exportation to GIS format; in green and red, optional stages for IVGE information enhancement; in red, the enhancement of the semantic information representation.

information from the borders to the cells improves the description of the IVGE and provides a coherent qualification of space (such as roads and buildings). Third, we showed how the ***semantic abstraction*** process enhances the hierarchical topological graph using the concept type lattice in order to build different views of the IVGE.

The abstraction model we proposed provides an hierarchically-structured graph-based description of IVGE which is suitable for the simulation of agents' spatial behaviours using spatial reasoning algorithms such as path planning and navigation. In the next chapter, we show how we leverage our enhanced IVGE model to support path planning algorithms which take into account both the environment and the agent type's characteristics.

Chapter 8

Motion Planning in Informed Virtual Geographic Environments

8.1 Introduction

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 find a path, using a search algorithm such as *A** [HNR72] or *Dijkstra* [Dij59], increases with the size of the search space (see Section 4). As a consequence, path planning in large-scale geographic environments can result in serious performance bottlenecks. However, representing the virtual environment using the hierarchical approach allows a reduction in the size of the search space as well as the problem complexity in path planning [HB08]. Two recent hierarchical triangulation-based path planning approaches are described in [DB06], 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. Like our method, 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* is notably different from our work. They aim to maximise triangle size whereas we aim to topologically abstract the IVGE by merging triangles into convex polygons. We also handle semantically enriched environment descriptions including qualification of space and terrain characteristics while both *TA** and *TRA** assume a homogeneous flat environment.

There is also a need for autonomous situated agents which are able to plan paths, and 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 characteristics or features. *Dynamic obstacles* correspond to other moving agents which are navigating in the VGE.

In this chapter, we address the issue of path planning for agents with different capabilities evolving in environments with several static and dynamic characteristics (Figure 8.1). We present an algorithm that implements hierarchical path planning and takes advantage of the hierarchical graph structure of the IVGE and of the rich semantic information provided by CGs in order to compute paths in semantically-enriched IVGE.

Section 8.2 presents how we leverage the hierarchical graph structure of the IVGE model in order to support situated reasoning algorithms such as hierarchical path planning. First, we propose a novel algorithm for hierarchical path planning in large-scale, complex, and informed virtual geographic environments. Second, we analyse the complexity of our algorithm and outline its efficiency to support motion planning for situated agents evolving within the IVGE. Third, it proposes a path optimization approach which enhances the path computed by the hierarchical path planning algorithm. In Section 8.3, we first present a graph-based structure called *Neighborhood Graph* (NG) to deal with collision detection and avoidance between moving agents in 3D virtual environments. We then highlight some results obtained by applying our approach to an urban geographic environment in order to address the issue of path planning as well as obstacle avoidance navigation with respect to both the environment's and the agents' characteristics.

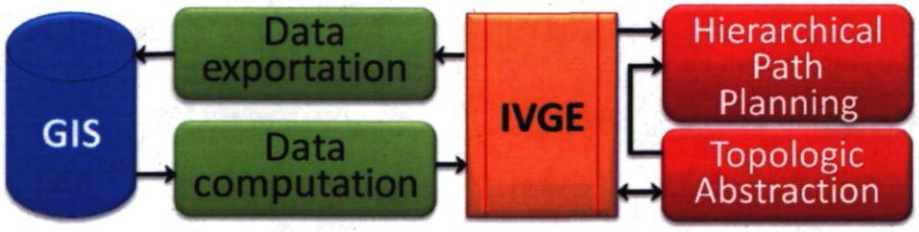


Figure 8.1: Global architecture for Hierarchical Path Planning in IVGE; Green: IVGE generation using GIS Data; Red: the topologic graph abstraction and hierarchical path planning processes.

8.2 Hierarchical Path Planning

In this section, we present our *Hierarchical Path Planning* algorithm (HPP for short) which leverages the Hierarchical Topological Graph (HTG) resulting from the topologic abstraction process (Section 7.3). We also propose a path enhancement method in order to optimise the computed paths for more realistic agent movements.

8.2.1 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 hierarchical topological graph (HTG) structure is multi-levels, 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 an HTG 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 *start*, a final destination *target*, and G, the objective of our algorithm is to plan a path from *start* to *target* 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[1] need to be considered:
  - Case 1: Both positions *start* and *target* are located 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: Positions *start* and *target* are located in different abstract nodes k and j at level i. Proceed to *step 2*.

[1]The case where one node is a *leaf* and the other is an *abstract node* may not occur because the topologic abstraction process generates an HTG in which every abstract node contains at least a single low level node.

- **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 *start* 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 closet 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 closet common boundary with the abstract node t.
  - If the final destination position *target* belongs to t, then identify to which group gt of t it belongs and proceed to *step 2*, in order to compute the path from the closet common boundary with the abstract node s. Else proceed to *step 2* in order to compute the path from the closet common boundary with the abstract node s to the center of the abstract node t.
- **Step 4:** Once in a leaf, apply a path planning algorithm (we use either the Djkstra or the A* algorithm) 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 at Step 2 is actually a coarse-grained path whose direction is only indicative. Since the path is refined in a *depth-first* way, agents can perform a local and accurate navigation inside an abstract node without requiring a complete and fine-grained path computation towards the final destination. The sub-paths of lower levels (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. Classical path planning approaches use the entire set of cells representing the environment and compute the complete path between the starting and 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 HPP 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 HPP can be 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.

8.2.2 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 target position t located in the virtual environment. The computation cost of the shortest path between s and t at level 0 (represented by the graph G_0) is denoted by $C_0(N)$ and is given by the following equation [BY98]:

$$C_0(N) = O(N * ln(N)) \tag{8.1}$$

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:

$$h(G_{i-1}(V_{i-1}, E_{i-1})) = G_i(V_i, E_i) \quad with \;\; 0 \leq i \leq k-1 \tag{8.2}$$

Let l_i be the *abstraction rate* between two successive levels $i-1$ and i (with $0 \leq i \leq k-1$). l_i characterizes the topologic abstraction process presented in Section 7.3. Since the abstraction process aims to reduce 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 8.3.

$$l_i = \frac{|V_{i-1}|}{|V_i|} \quad with \;\; l_i > 1 + \varepsilon \; and \; \varepsilon > 0 \tag{8.3}$$

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 8.2 and 8.3 as follows:

$$N = M * l_{k-1} * ... * l_0 \tag{8.4}$$

$$N \geq M * (1+\varepsilon)^k \quad with \;\; k > 0 \; and \;\; \varepsilon > 0 \tag{8.5}$$

$$N = M * \prod_{i=0}^{k-1} (l_i) \quad with \;\; k > 0 \; and \;\; M > 0 \tag{8.6}$$

Let l_{Avg} be the average value of l_i (with $0 \leq i \leq k-1$). Using l_{Avg}, equation 8.6 becomes:

$$N \approx M * l_{avg}^k \quad with \;\; k > 0 \; and \;\; M > 0 \tag{8.7}$$

Let us replace the term N in equation 8.1 by its value in equation 8.7:

$$C_0(M) \approx MO(M * l_{avg}^k * ln(M * l_{avg}^k)) \tag{8.8}$$

Equation 8.8 can be developed as follows:

$$C_0(M) \approx MO(M * ln(M) * l_{avg}^k + M * l_{avg}^k * ln(l_{avg}^k)) \tag{8.9}$$

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:

$$Nb_k \approx MO(M * ln(M)) \quad with \;\; k > 0 \; and \;\; M > 0 \tag{8.10}$$

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 t 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:

$$C_k(M) \approx MNb_k * \sum_{j=0}^{k-1} l_{avg}^j \tag{8.11}$$

$$C_k(M) \approx MNb_k * \frac{l_{avg}^k - 1}{l_{avg} - 1} \tag{8.12}$$

The term Nb_k in equation 8.12 is replaced by its value expressed in the equation 8.10 as follows:

$$C_k(M) \approx MO(M * ln(M) + M * \frac{l_{avg}^k - 1}{l_{avg} - 1}) \tag{8.13}$$

Let us compare the computation costs of standard path planning approaches (equation 8.9) and our hierarchical path planning approach (equation 8.13). First, it is obvious that the first term $M * ln(M)$ in equation 8.9 is inferior to the first term $M * ln(M) * l_{avg}^k$ in equation 8.13

since the abstraction rate $l^k_{avg} > 1$. Second, in a similar way, the second term $M * l^k_{avg} * ln(l^k_{avg})$ in equation 8.9 is inferior to the second term $M * (l^k_{avg} - 1/l_{avg} - 1)$ in equation 8.13. In conclusion, the hierarchical path planning algorithm along with the hierarchical topologic graph that we propose is at least $ln(l^k_{avg})$ orders of magnitude faster than standard path planning approaches.

8.2.3 Path Optimisation

The topological abstraction only assembles 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 optimisation* (Figure 8.2).

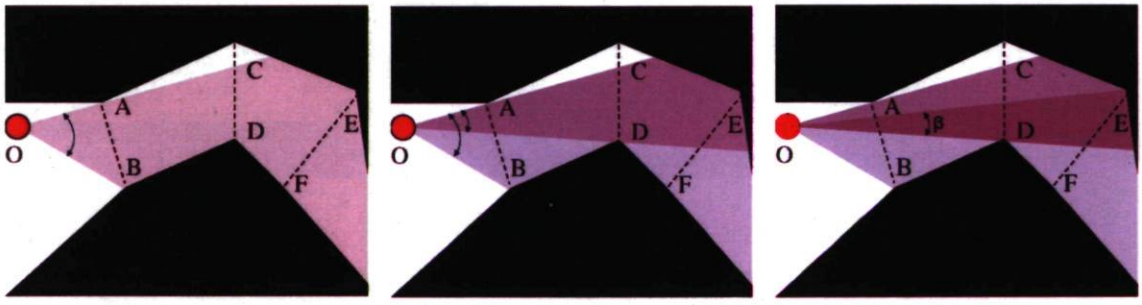
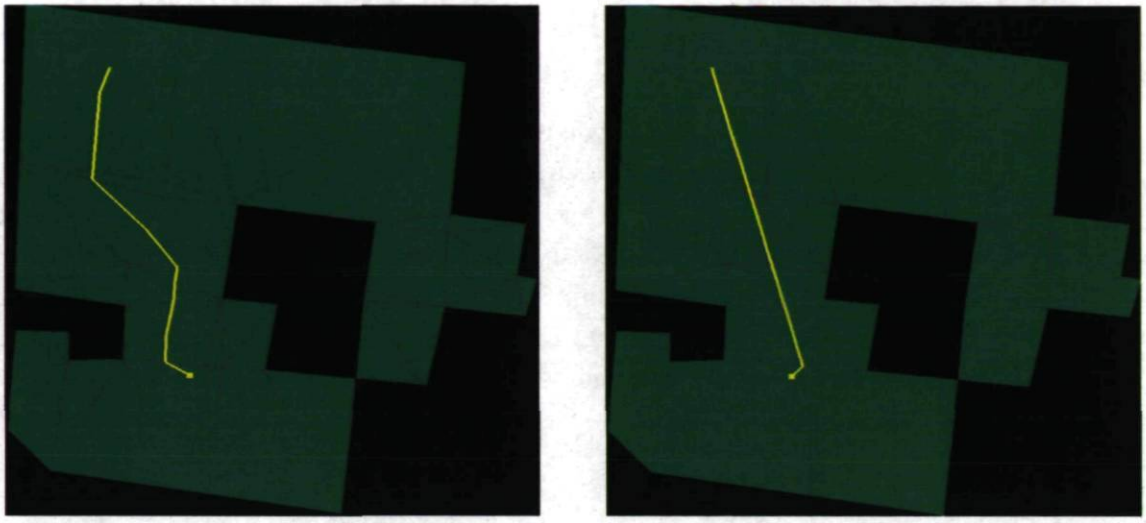


Figure 8.2: The optimization process of the computed path in IVGE using sequentially intersecting visibility cones. This process stops if the cone is empty or if the angle of the cone is lower than a certain user-defined threshold β.

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. The optimization algorithm consists of sequentially intersecting visibility cones defined using the path sub-optimal parts. This computation stops if the cone is empty or if the angle of the cone is lower than a certain user-defined threshold β. If one of those two constraints is violated, the last valid cone is selected by the algorithm. Since cells are convex, this cone contains an obstacle free straight line path (the perpendicular line from the base to the apex) that the agent can follow (Figure 8.3).

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



(a) (b)

Figure 8.3: (a) The original computed path ; (b) The computed path after optimisation.

the original computed path length with the optimised path length. Figure 8.4 depicts the comparison of the non-optimised computed path length and the optimised path length. It shows how the optimisation process reduces the length of the computed path by an average of 16%. The average time for the computation of an optimised path is 217 milliseconds. The computation of an optimised path costs 12% more time than a standard path planning algorithm.

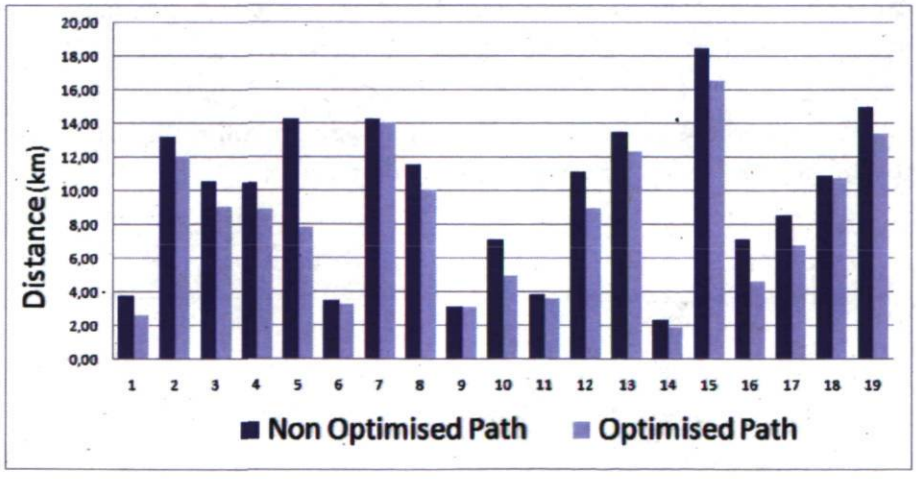


Figure 8.4: Optimised versus non-optimised paths lengths.

8.3 Navigation and Neighborhood Graph

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 collision 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 agents' navigation. Next, we detail the algorithm that we developed to build an NG while taking into account obstacles such as walls and fences, as well as terrain elevation.

8.3.1 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 any obstructions resulting from the terrain's shape (*static information*). Figure 8.5 shows an example of construction of a NG considering obstacles and obstructions within the IVGE. In the following sub-section, we present an algorithm to build NGs and we analyze its complexity.

8.3.2 Algorithm

In this section, we present a step by step description of our algorithm to build NGs. 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:

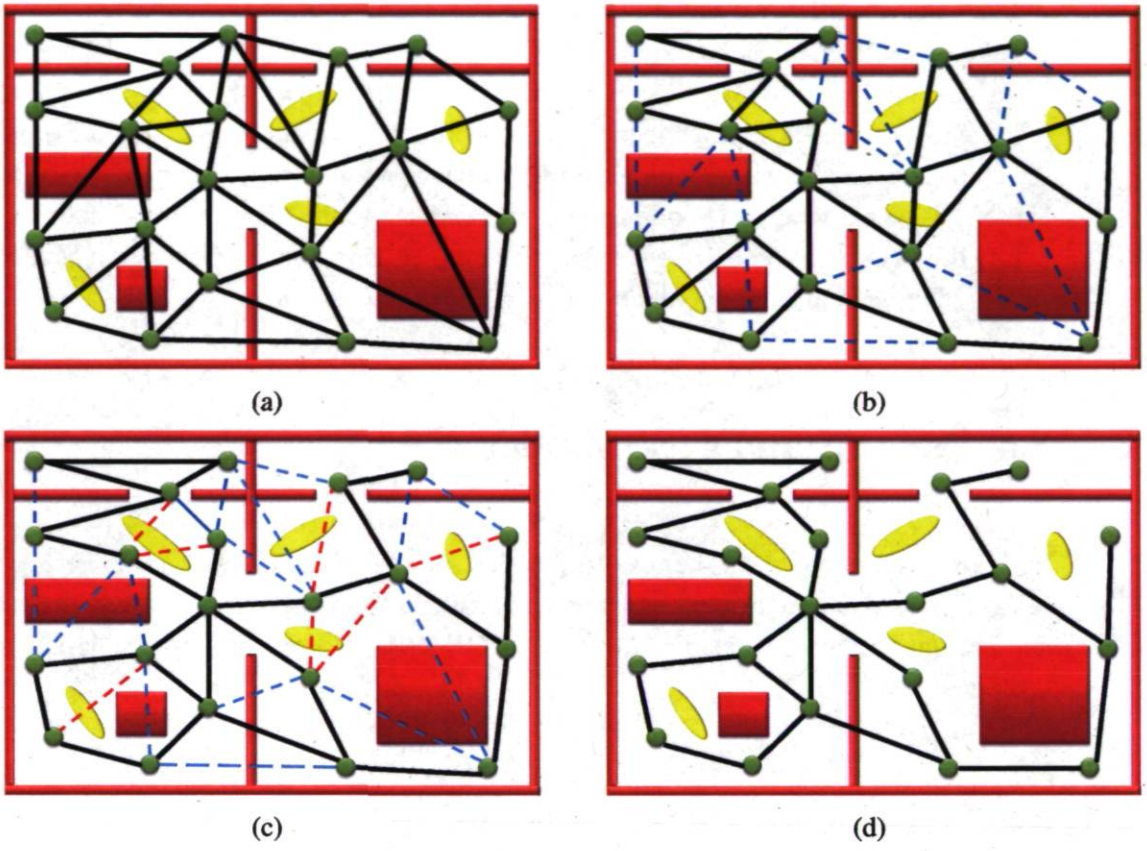


Figure 8.5: Generation of the neighborhood graph (NG): (a) initial 3D-DT using the moving agents positions (green circles); (b) filter of 3D-DT considering obstacles (red shapes) in the IVGE; (c) filter of 3D-DT considering terrain obstructions (yellow shapes) in the IVGE.

- **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), determine the triangle t_i that contains P_i using the accurate localization technique that we introduced in 6.3.2. Once triangle t_i is found, a graph search over the informed topologic graph is performed regardless of the semantic information, starting from t_i until the triangle containing P_j is found. The edges as well as the triangles traversed by the graph search are stored.
- **Step 3**: If $E_{i,j}$ crosses an edge that is associated with an obstacle semantic label or if $E_{i,j}$ is obstructed as a result of the terrain, it is removed.

Figure 8.5(d) illustrates how the edge is deleted from the 3D-DT if $E_{i,j}$ is not free from environmental obstacles (blue dashed lines identified through the step 1) or from freedom from terrain obstructions (red dashed lines identified through the step 2).

The NG algorithm generates a linear number of neighborhood relations (edges of the triangulation) with an upper bound of $3n-3$ relations. Those relations ensure the prediction of collisions, because one of the Delaunay triangulation properties is to link each point (moving agent) to its nearest neighbour. Moreover, this triangulation generates the topological structure of agents' position with a computational cost independent of the distance between entities. The neighborhood relation length is correlated to the local density of moving agents within the IVGE. As a consequence, collision prediction in a sparse population of agents and near prediction in a dense population of agents are both supported.

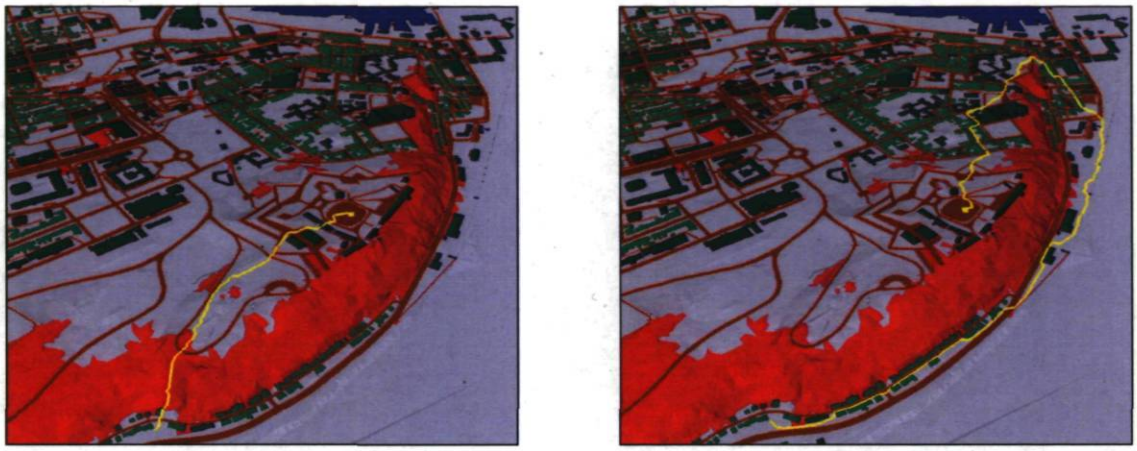
8.4 Results and Case Studies

In this section, we present the results obtained with our IVGE model, our abstraction processes, and our hierarchical path planning algorithm. We analyze the achieved performance, and present various case studies in which IVGE, HPP and spatially-reasoning agents have been used.

8.4.1 Case Study: HPP in Quebec City

The HTG resulting from the topologic abstraction process is particularly suitable to support HPP in an IVGE. Two types of HPP have been implemented: 1) a path linking two positions located in the IVGE using the A* algorithm (Figure 8.6(a) and Figure 8.6(b)); and 2) a search path linking a position to a qualified area within the IVGE using the *Dijkstra* algorithm (Figure 8.7(a) and Figure 8.7(b)). Figure 8.6(a) shows a planned path which avoids obstacles such as *buildings* and *walls*, but does not take into account the terrain characteristics. Therefore, this path crosses areas coloured in red which represent steep slopes. However in Figure 8.6(b), the algorithm has generated a path which respects both the terrain and the obstacles in the IVGE. Indeed, the steep slopes (coloured in red) are avoided since they are now considered as obstacles. This path is longer, but it fully respects the constraints of the environment and the slope of the terrain.

In multi-agent geo-simulations, agents may not aim to reach a particular position but rather a particular area in the IVGE. This type of path planning answers the question: *how to find a path to reach a specific area while respecting the environment's constraints?* The semantic information (building, house, marina, wall of the old city, etc.) integrated in the IVGE description helps answer such a question. Using the HTG along with the *Dijkstra* algorithm, the system computes the shortest path to reach a specific area located in the IVGE. In our



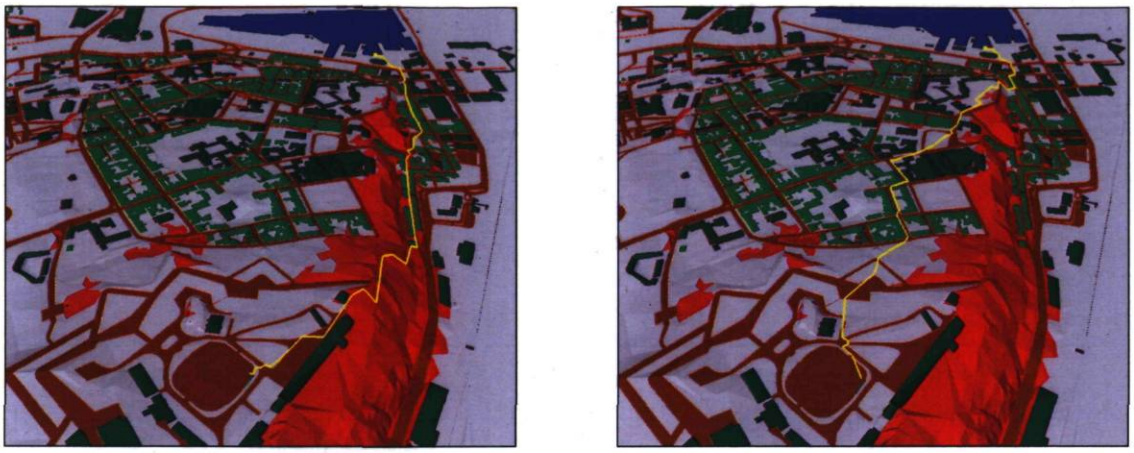
(a) (b)

Figure 8.6: HPP in the IVGE (the computed path is coloured in yellow). (a) path computed with no regard for the terrain; (b) path computed with regard for terrain shape.

case, the stopping condition of the algorithm is related to the semantics of the visited nodes. If the semantics associated with the visited node correspond to the target area's characteristics, then the algorithm stops and the path is generated. To illustrate path planning towards a target area qualified by one or several semantic conditions, Figure 8.6(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 8.7(b) respects both the terrain and the obstacles in the IVGE. We propose the following example: a tourist who moves using a wheelchair is located inside the old city of Quebec. This tourist wants to visit an attraction spot called the *marina*. Here, the *marina* is not identified by coordinates (x, y, z), but rather by semantic information. A path that only avoids the obstacles of the environment but crosses steep slopes areas (coloured in red) is obviously not acceptable for this tourist. Figure 8.7 shows two computed paths to reach the marina (the marina is coloured in blue at the top of the figure). Figure 8.7(a) illustrates a planned path with no regard for obstacles or terrain and Figure 8.7(b) considers both.

8.5 Discussion

In order to reduce the search space, we proposed a HTG (see Section 7.3) 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 of 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



(a) (b)

Figure 8.7: HPP in the IVGE (the computed path is coloured in yellow). (a) path computed to get to a place (marina) in the IVGE (place described by semantics) with no regard for the terrain shape; (b) path computed with regard for the terrain shape.

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, regardless of 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 its top to its bottom in order to compute such a set of cells. The assumption is that the hierarchical traversal cost is at least offset by reducing the search space - and therefore the computation cost - at lower levels of the hierarchy. However, it is possible that the benefit may not occur with a sufficiently high number of abstraction levels and sufficiently complex graphs.

Second, the *information richness* decreases with each grouping level, which could lead to unnecessary additional abstraction levels that may not improve the decision making function 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 ability to represent geographic environments which are distantly separated 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, let us say *312 Marie-Louise*, to a given address in New York City, let us say *1213 4th 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 4th Avenue*. 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 separated geographic environments and hence allows us to accurately compute optimal paths with respect to these environments' characteristics.

In contrast to Lamarche and Donikian's NG [LD04] which only takes into account static obstacles such as walls and fences (Figure 8.5(b)), our NG model also includes obstructions resulting from the terrain's elevation (Figure 8.5(c)). Lamarche and Donikian's NG is based on a 2D-DT implementing the algorithm proposed in [BY98] whose computation cost is of $O(n\log n)$ [LD04]. However, Attali *et al.* proposed an optimised algorithm to build a 3D-DT which also runs in $O(n\log n)$ [ABL03]. 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 the 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 densely populated environments, the $k-direct$ neighborhood contains a set of agents which are visible and close to the agent E, and in environments with sparse populations of agents, 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.

8.6 Conclusion

In this chapter, we presented an hierarchical path planning algorithm for the support of agents' motion planning in large-scale and complex virtual geographic environments. This algorithm leverages the hierarchical graph structure which characterizes our IVGE model and provides computed paths which take into account the agents' and environment's characteristics. In addition, we detailed the algorithm and discussed its complexity with regard to standard planners. We also proposed a process for the optimisation of the computed paths. In order to support agents' navigation in IVGE, we proposed the neighborhood graph approach. This approach uses the Delaunay triangulation technique and provides agents with a visibility that is free from environmental obstacles and terrain obstructions.

Chapter 9

Informed Virtual Geographic Environments and Knowledge Management

Introduction

In this chapter, we propose a novel approach which exploits our IVGE description in order to effectively manage knowledge about the environment in order to support agents' spatial behaviours. Our approach relies on previous work on humans spatial behaviours and the way people apprehend the spatial characteristics of their surroundings in order to navigate and to interact with the physical world. It is also inspired by Gibson's work on affordances [Gib79] and knowledge provided by the environment to guide agent-environment interactions. The main contribution of our approach is to provide situated agents with: (1) knowledge about the environment represented using Conceptual Graphs (CG) [Sow99]; (2) tools and mechanisms that allow them to acquire knowledge about the environment; and (3) the capability to reason about this knowledge and to autonomously make decisions and to act with respect to both their own and the virtual environment's characteristics.

Section 9.1 provides a short survey of works on agents' spatial behaviours, introduces the *affordance* concept, presents the notion of *knowledge about the environment*, and outlines its importance for spatial agents in order to let them make decisions that take into account the characteristics of the virtual geographic environment in which they evolve. Section 9.2 introduces the concept of *Environment Knowledge* (EK) and details the method that we propose to define it using Conceptual Graphs (CGs) [Sow99]. Section 9.2.2 presents the environ-

ment knowledge base and Section 9.2.3 details the decision making process which uses this knowledge base and involves an inference engine in order to generate facts. Section 9.3.1 provides a description of the proposed agent model and Section 9.3.2 presents patterns of spatial behaviours. Section 9.4 illustrates, through two case studies, how environment knowledge management supports situated agents's spatial behaviours.

9.1 Spatial Behaviours and Knowledge Management

Research on spatial behaviours investigates the processes that take place when spatial agents representing people or other dynamic actors orient themselves and navigate through complex and large-scale virtual geographic environments [AM05, Rau01]. In order to build agents that exhibit plausible spatial behaviours with respect to their capabilities and to the virtual environment characteristics in which they evolve, we need to analyse humans' spatial behaviours in the physical world [Zac06]. We also need to determine how spatial agents can make decisions using knowledge provided by the virtual environment.

In this section, we present several works related to spatial behaviours and outline the importance of knowledge about the environment for the support of agents' spatial behaviours.

9.1.1 Spatial Behaviours

Several theories in the field of *human spatial behaviours* have been proposed in order to explain how people navigate in the physical world, what people need to find their ways, and how people's visual abilities influence their decisions [FBR01]. Actually, these theories point out the use of various spatial and cognitive abilities to apprehend the physical world in which people evolve and with which they interact [GRW08]. Weisman identified four classes of environmental variables that influence spatial behaviours in physical worlds: *visual access*; *architectural differentiation*; *signs to provide identification or directional information*; and *plan configuration* [Wei81]. Seidel's study at the Dallas/Fort Worth Airport showed that the spatial structure of the physical environment has a strong influence on people's spatial behaviours [Sei82]. Arthur and Passini introduced the term *environmental communication*, arguing that the built environment and its parts should function as a communication device [AP98]. For example, the authors mention two main aspects regarding the understanding of buildings: a *spatial aspect* that refers to the total dimensions of the building and a *sequential aspect* that considers a building in terms of its destination routes. Information about the geographic environment along with the spatial and cognitive capabilities are fundamen-

tal inputs to the spatial decision-making process [GRW08]. These abilities are a necessary prerequisite to use environmental information or representations of spatial knowledge about the environment. This knowledge include information collected using perception capabilities, memorised information resulting from past experiences, and information provided by the environment it self [JOH07]. The latter knowledge is known as "'*affordance*"'.

9.1.2 Affordance

The theory of affordances [Gib79] is based on ecological psychology which advocates that knowing is a direct process: the perceptual system extracts invariants embodying the ecologically significant properties of the perceiver's world. Gibson's theory is based on the tenet that animal and environment form an inseparable pair. Affordances have to be described relative to the person. For example, a chair's affordance "'*to sit*"' results from a bundle of attributes, such as "'*flat and hard surface*"' and "'*height*"', many of which are relative to the size of an individual. Later work with affordances built on this so-called agent-environment mutuality [War95].

Many researchers believe that Gibson's theory is insufficient to explain perception because it neglects processes of cognition. His account only deals with individual phenomenon, but ignores categories of phenomena [Lak90]. Norman investigated affordances of everyday things such as doors, telephones and radios, and argued that they provide strong clues to their operation [Nor02]. He recast affordances as the results from the mental interpretation of things, based on people's past knowledge and experiences, which are applied to the perception of these things. In an analogous way, our approach we suggest that exploiting the enriched description of virtual environments can provide agents with knowledge about their environment to support their spatial reasoning mechanisms and to help them make decisions that take into account the characteristics of their surroundings.

9.1.3 Knowledge about the Environment

Knowledge is an important asset for agents because it allows them to reason about it and to autonomously make informed decisions [WJ95]. By its very nature, knowledge is *disparate* and *heterogeneous* and can be represented in various ways (qualitatively and quantitatively), and can be either structured or unstructured, as we discussed in Section 2.6. Knowledge usually includes information about the agent's characteristics, as well as about the description of the geographic environment in which it is situated. On the one hand, the agent's characteristics encompasses its beliefs, desires, intentions, preferences, constraints, goals [Bra99]. On

the other hand, the description of the geographic environment relies on its geometric, topologic and semantic characteristics [WSH05]. Thus, spatial agents require knowledge about their environment in order to reason about it, to infer facts, and to draw conclusions which will guide them to make decisions and to act. A number of challenges arise when creating knowledge about the environment for spatial agents' decision-making, among which we mention: 1) to represent knowledge using a standard formalism; 2) to provide agents with tools and mechanisms to allow them acquire knowledge about the environment; and 3) to infer and to draw conclusions and facts using premises that characterise the geographic environment.

Although several research works have addressed the issue of modelling and representing agents's characteristics using formal and standard formalisms [RG91, WJ95, RG95, Bra99, TPH02, Min03], very few works have attempted to adopt a standard formalism in order to represent virtual environments' characteristics [WOO07]. The main reason why virtual environments have received less interest from practitioners is that geographic environments may be *complex*, *large-scale*, and *densely* populated with a variety of *geographic features*. As a consequence, formally representing knowledge about geographic environments is usually complex and time consuming [WSH05]. Another issue which needs to be addressed is the way to allow spatial agents to acquire this knowledge in order to autonomously make decisions. There is a need for a *knowledge management* approach: (1) to represent knowledge about geographic environments using a standard formalism; (2) to allow spatial agents to acquire knowledge about the environment; (3) to allow agents to reason using knowledge about geographic environments.

Knowledge management relies on a contextualised, organised, structured information characterising the virtual environments [GDB+98]. However, most of the current virtual environments are mainly built for computer and behavioural animation purposes (see Chapter 2). In contrast to our IVGE model, the description of these virtual environments only includes geometric data and very seldom topologic characteristics and semantic information (see Section 2.5). Moreover, the description of these virtual environments does not rely on standard geographic data provided by GIS (see Section 2.1), neither on exact space decomposition techniques. In addition, these descriptions are not structured to support complex and large-scale geographic environments (see Section 2.4). The semantic information that qualifies and characterises these virtual environments is not represented using a standard formalism which allows for its organisation and contextualisation [LaV06]. Current virtual geographic environments do not allow agents to acquire, to organise and to use knowledge to reason about the environment.

In contrast, our IVGE model has been built with respect to the above mentioned knowledge management requirements. In the following section, we detail our approach of knowledge management in IVGE to support agents' spatial behaviours.

9.2 From Semantic Information to Environment Knowledge

In Chapter 6, we proposed a novel model along with a complete methodology for the automated generation of informed VGEs. Then, we presented our abstraction approach which enriches and structures the description of the IVGE, using geometric, topologic and semantic characteristics of the geographic environment. In order to represent semantic information which characterises our informed virtual environment model, we also proposed to use Conceptual Graphs (CGs) [Sow99] (see Section 7.4.1). Our aim now is to evolve the semantic information to the level of *knowledge* and hence to build a knowledge-oriented virtual geographic environment in which spatial agents autonomously make informed decisions.

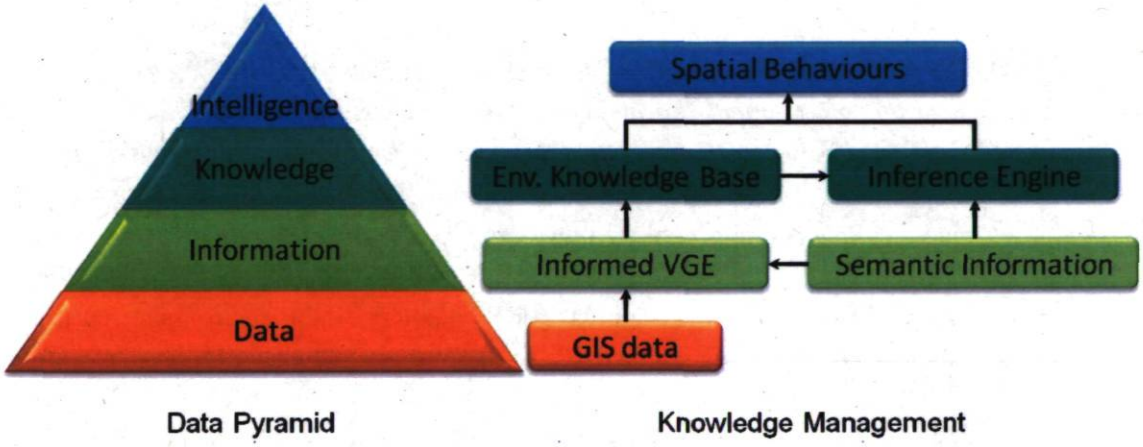


Figure 9.1: The proposed knowledge management approach; on the left hand side, the pyramid data [LG89]; on the right hand side, the knowledge management approach relying on our IVGE model and involving a knowledge base coupled with an inference engine to support agents' spatial behaviours.

The process of making an informed decision has been modeled as a pyramid built on data [LG89] as shown on the left hand side of Figure 9.1. *Data* corresponds to the transactional, incremental physical records [LG89]. In our IVGE model, this data corresponds to the geometric and geographic data provided by GIS. In and of itself this data is not sufficient to support spatial agents' decision-making. This data must be organized into information in order to be useful. *Information* is data that has been contextualized, categorized, often calculated (from initial data), corrected, and usually condensed [Saj08]. In our IVGE model, information corresponds to the description of the IVGE resulting from the exact spatial decomposition of the geographic environment and enhanced with semantic information. Information often contains patterns within it and is sometimes useful for simple spatial behaviours such as motion planning as we showed in Section 8.2. However, the context of these spatial behaviours can only be formed using some *knowledge*. *Knowledge* provides the next step of data organisation. For information to become knowledge, the context of the infor-

mation needs to include predictive capabilities. Using predictive capabilities of knowledge, spatial agents can autonomously make informed decisions. The more complex and voluminous the underlying data sets are, the more effort is required to progressively organise it so that it becomes knowledge useful to the agents' decision-making. However, since our IVGE description is structured as a hierarchical topologic graph resulting from the geometric, topologic, and semantic abstraction processes, and since the semantic information is expressed using conceptual graphs, we are able to build knowledge about the environment to support agents' spatial behaviours.

9.2.1 Environment Knowledge

We define the notion of ***Environment Knowledge*** (EK) as "'*a specification of a conceptualization of the environment characteristics: the objects, agents, and other entities that are assumed to exist in the informed virtual geographic environment and the relationships that hold among them*"'. Hence, EK is a description (like a formal specification of a program) of the spatial *concepts* (geographic features) and *relationships* (topologic, semantic) that may exist in a geographic environment. This description is provided by users in order to enrich the qualification of the geographic features which characterise the environment. It is expressed using a standard formalism which is close to natural language and computer tractable.

Let us emphasize that enhancing a multi-agent geo-simulation with EK, allows spatial agents to reason about the characteristics of the virtual geographic environment. Practically, EK is composed of *spatial concepts* (i.e., ask queries and make assertions) and spatial relationships (i.e., describe actions and behaviours). Our aim is to improve the perception-decision-action loop (see Section 3.1) on which rely most agent models. Considering Newell's pyramid [New90] which comprises the reactive, cognitive, rational and social levels of agent behaviours, we mainly focus on the knowledge acquisition process in order to support the decision-making capabilities of spatial agents. Figure 9.2 illustrates two elements: (1) the knowledge acquisition process, and (2) the action archetype process, that we introduced in order to extend Newell's initial pyramid. The knowledge acquisition process is detailed in Section 9.2.3. The agent and action archetypes are described in Section 9.3.1 and Section 9.3.2.

Figure 9.3 illustrates the way we extend our IVGE model in order to evolve the semantic information that enriches the description of the IVGE to the level of environment knowledge. The management of the environment knowledge is composed of two main parts: (1) an *Environment Knowledge Base* (EKB) which relies on spatial semantics represented using the CG formalism; and (2) an *Inference Engine* (IE) which allows to manipulate and to acquire environment knowledge in order to provide spatial agents with the capability of reasoning

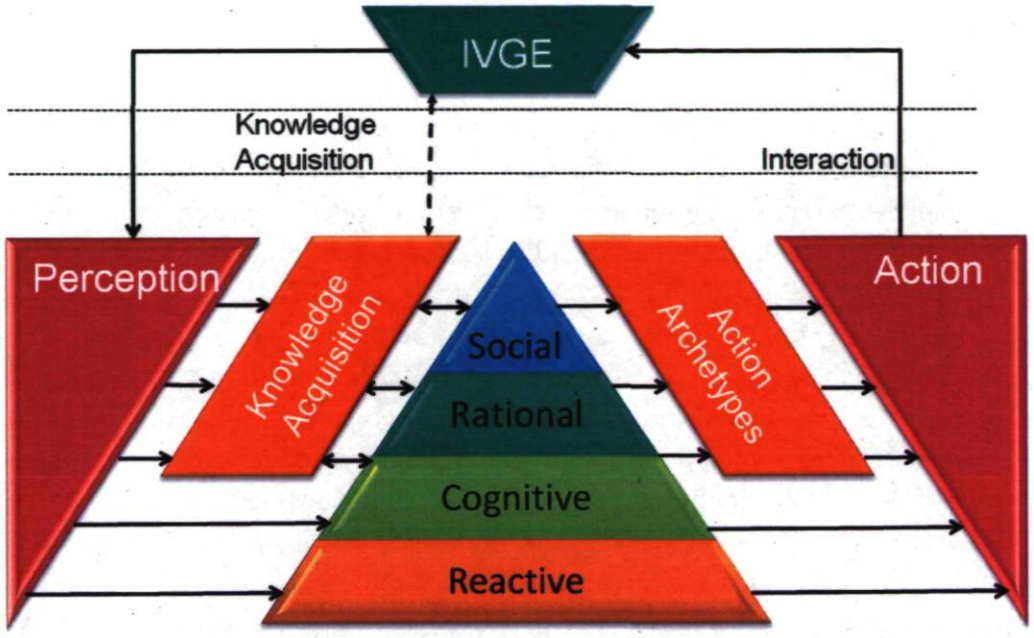


Figure 9.2: The enhanced perception-decision-action loop.

about it.

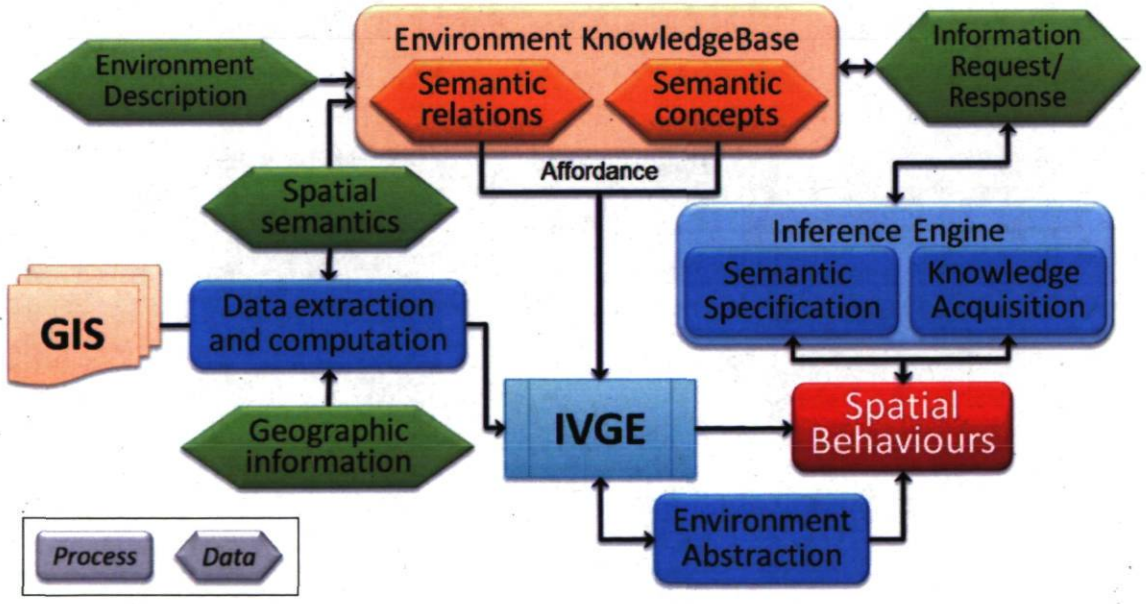


Figure 9.3: Environment Knowledge Management using the IVGE Model; in blue the different processes including the knowledge acquisition process which is supported by the inference engine, in green the data structures; in orange the environment knowledge base which uses CGs to represent the spatial semantic based on spatial *relations* and *concepts*.

9.2.2 Environment Knowledge Base

In Section 7.4.1, we showed how we use CGs to represent semantic information. However, CGs are typically used to represent knowledge [Sow99, SW86]. Actually, CGs enable us to formally represent spatial semantics characterizing our IVGE model and allow us to build a structured *Environment Knowledge Base* (EKB) based on a finite bipartite graph [Sow84]. The EKB allows MAGS users to represent, using a standard formalism, the information characterizing the virtual environment as well as the objects and agents it contains. Moreover, the EKB enables us to explicitly specify *affordances* [Gib79] in order to support the agents spatial interactions with the informed virtual geographic environment in which they evolve. Figure 9.3 details our approach to extend the IVGE model in order to support the management process of environment knowledge. The environment knowledge base, which is part of this process relies on the notion of *spatial semantics*. *Spatial Semantics* (SS) consists of a structured, conceptualised, and organised representation of geographic features, agents, and objects that an informed virtual geographic environment may contain. Spatial semantics relies on two types of nodes: *semantic concepts* and *semantic relations*. Semantic concepts represent entities such as agents, objects and zones as well as attributes, states and events. Semantic relations represent the relationships that hold among semantic concepts.

The environment knowledge can be constructed by assembling percepts. In the process of assembly, semantic relations specify the role that each percept plays and semantic concepts represent the percepts themselves. Semantic concepts involve two types of functions; *referent* and *type*. The function *referent* maps semantic concepts to generic markers denoted by names starting with an asterisk * or individual markers usually denoted by numbers. For example, if the referent is just an asterisk, as in $[HOUSE : *]$, the concept is called a generic semantic concept, which may be read as *a house* or *some house*. The function *type* maps concepts to a set of type labels. A semantic concept s_c with type $(s_c)= t$ and reference $(s_c)= f$ is displayed as $[t{:}f]$. The function type can also be applied to relations. For example, if the referent is a number [HOUSE:#80972], the field to the left of the colon contains the type label HOUSE, the field to the right of the colon contains the referent #80972 which designates a particular house.

To sum up, the EKB contains knowledge about the informed virtual geographic environment that an agent may use. This knowledge is provided basically by users to enrich the qualification of the geographic features which characterise the IVGE. Finally, this knowledge is structured using semantic concepts and relations expressed using conceptual graphs.

9.2.3 Inference Engine

Now that we have defined the environment knowledge base as a structure which contains explicit descriptions of geographic features using CGs, let us describe the Inference Engine (IE) which is part of our knowledge management approach. Figure 9.3 shows how the IE module is integrated in our knowledge management architecture. The IE is a computer program that derives answers from our environment knowledge base (Figure 9.4). Therefore, the IE must be able to logically manipulate symbolic CGs using formulas in the first-order predicate calculus. In order to acquire knowledge about the virtual environment, agents use the IE and formulate queries using a semantic specification that is compatible with CGs. Agents interpret the answers provided by the IE and act on the environment. They can also enrich the EKB by adding new facts that result from their observation of the virtual environment (Figure 9.4).

In this sub-section, we first present how CGs allow us to map knowledge about the environment into first-order logic formulas. Then, we provide a short survey of existing tools that support the manipulation of CGs. We also discuss the capabilities of these tools to provide a programming language with CGs, related operations, and inference engine. Finally, we present the Amine platform [Kab06], a platform to manipulate CGs using an inference engine embedded in PROLOG+CG language.

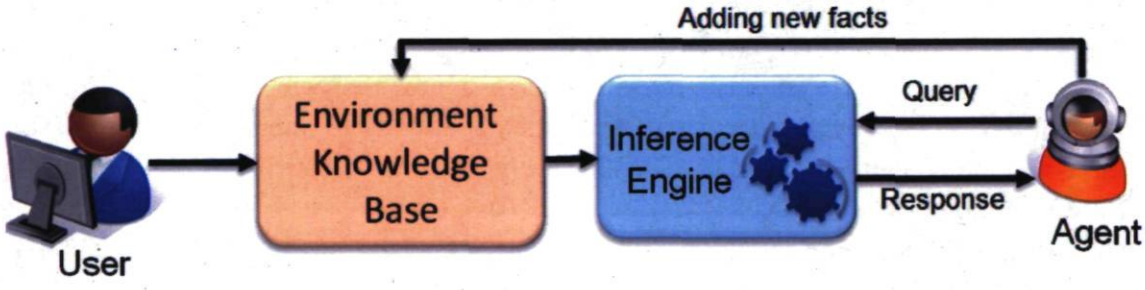


Figure 9.4: The inference engine uses the EKB for the purpose to answer queries formulated by agents.

Conceptual graphs offer the opportunity to map knowledge about the environment into formulas in the first-order predicate calculus. If u is a description of a geographic feature represented using conceptual graphs, then fu is a formula determined by the following construction as proposed [Sow84]:

- If u contains k generic semantic concepts, assign a distinct variable symbol $x1$, $x2$, $x3$, ..., xk to each one.
- For each semantic concept c of u, let *identifier* (c) be the variable assigned to c if c is

generic or $referent(c)$ if c is individual.

- Represent each concept c as a predicate whose name is the same as $type(c)$ and whose argument is $identifier(c)$.
- Represent each conceptual relation r of u as a predicate whose name is the same as $type(r)$. For each i from 1 to n, let the ith argument of the predicate be the identifier of the concept linked to the ith arc of r.
- Then fu has a quantifier prefix $x_1\ x_2\ x_3\ ....\ x_k$ and a body consisting of the conjunction of all the predicates for the concepts and conceptual relations of u.

In order to illustrate this construction method, consider the following example. If graph u is:

[CAR:#98077]->(STAT)->[SIT]->(LOC)->[ROAD]

then fu is:

xy(CAR(#98077) $\wedge$ STAT(#98077,x) $\wedge$ SIT(x) $\wedge$ LOC(x,y) $\wedge$ ROAD(y))

Using such formulas in the first-order predicate calculus, it is possible to build tools that allow spatial agents to manipulate knowledge about virtual environments represented using CGs. Moreover, it is possible to build tools that allow for logic and symbolic manipulations of environment knowledge and provide the opportunity to infer and to predict facts or assumptions about virtual environments. Several tools can be used to manipulate CGs (Amine, CGWorld, CoGITaNT, CPE, Notio, WebKB). CGWorld was a Web based workbench for joint distributed development of a large knowledge base of CG, which resides on a central server [DT02]. CGWorld is not supported anymore. WebKB is a KB annotation tool and a large-scale KB server [ME01]. CoGITaNT is an IDE for CG applications [GS98]. Notio is not a tool but a Java API specification for CG and CG operations [SL99]. It is not supported anymore. CPE has been developed as a single standalone application[1]. Currently, CPE is being upgraded to a set of component modules (to render CPE an IDE for CG applications). Its author announces that CGIF and basic CG operations (projection and maximal join) are coming soon[2]. The new upgraded version of CPE is underway and it is not yet available. Kabbaj developed *Amine*, a platform to manipulate CGs [KB05, Kab06]. Amine provides a pattern-matching and rule-based functions embedded in PROLOG+CG language which is basically an object-based and CG-based extension of the PROLOG language.

[1] http://port.semanticweb.org/CPE

[2] http://conceptualgraphs.org/

CG tools can be classified under at least 8 categories of tools: CG editors, executable CG tools, algebraic tools (tools that provides CG operations), KB/ontology tools, ontology server tools, CG-based programming languages, IDE tools for CG applications and, agents/MAS tools. The category "'*IDE for CG applications*"' means a set of APIs and of GUIs that allow a user to construct and manipulate CGs and to develop various CG applications. Only Amine and CoGITaNT belong to this category. The category "'*CG-based programming language*"' concerns any CG tool that provides a programming language with CG, related operations, and inference engine. Only Amine belongs to this category, with its programming language: *Prolog+CG*. Therefore, we propose to use the Amine platform and Prolog+CG in order to logically manipulate symbolic CGs and to provide spatial agents with an inference engine that allows them to query the environment knowledge, to acquire environment knowledge and reason about it.

Figure 9.5 presents the global architecture of the Amine platform. Using this platform, users can build an environment knowledge base (EKB) using CGs and query the Amine's inference engine (IE) to derive new knowledge from the content of the EKB using queries. The Amine platform provides a graphic user interface to support the manipulation of the EKB. Agents are able to send queries, during the simulation process, in order to acquire the knowledge they need to make a decision, using the Prolog+CG language which is provided by the Amine platform. These queries are processed by the IE which interrogates the EKB and sends back the response to agents.

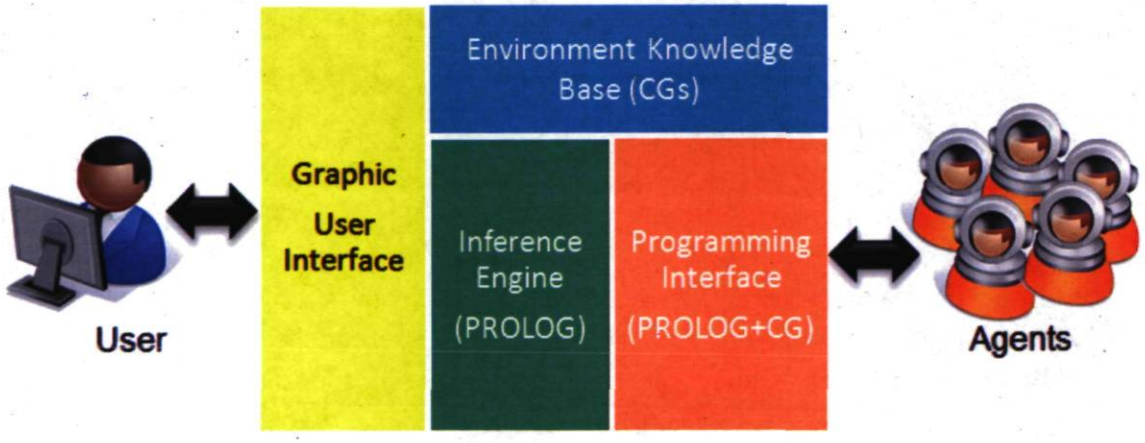


Figure 9.5: Architecture of the Amine platform.

In order to illustrate such a querying process, let us consider the following simple environment knowledge, composed of a set of two facts which provide an idea of the use of conceptual structures as a Prolog+CG data structure:
cg([Man:Mehdi]<-agnt-[Study]-loc->[University]).
cg([Man:Mehdi]<-agnt-[Play]-obj->[Soccer]).

And the following request: "'*Which actions are done by Man Mehdi ?*"'
?- cg([Man:Mehdi]<-agnt-[x]).

The answer provided by the Amine platform using its Prolog+CG inference engine is:
x = Study;
x = Play;

Now that we introduced the main parts of our environment knowledge management approach, namely EKB and IE, we detail in the following section the notions of agent and action archetypes and the way we use them to build spatial behaviours.

9.3 From Environment Knowledge to Spatial Behaviours

When dealing with MAGS involving a large number of spatial agents of various kinds, the specification of their attributes and associated spatial behaviours might be complex and time and effort consuming. In order to characterise our spatial agents, we propose to specify: (1) the agent archetype, its super-types and sub-types according to the semantic type hierarchy that we defined in Section 9.2.2; (2) the agent category (such as actor, object, and spatial area); and (3) the agent spatial behavioural capabilities, including moving within the IVGE content, perception of the IVGE and of other spatial agents. In the following subsections we discuss these elements.

9.3.1 Agent Archetypes

In our environment knowledge management approach, the description of agents as well as objects and geographic features (spatial areas and zones) is enriched with semantic information. This means that these spatial agents belong to a semantic type hierarchy. Using the semantic type hierarchy allows us to take advantage of *inheritance* mechanisms. Hence, when modelling a MAGS involving a large number of agents, we only need to specify the attributes that are associated with the highest-level types of agents that we call agent archetypes rather than repeatedly specifying them for each lower-level agents. Let us define the Prolog+CG rule used to build a semantic type-hierarchy as follows: *Supertype* > *Subtype1*, *Subtype2*, ..., *SubtypeN*.

Below is an example of a portion of semantic type-lattice expressed in Prolog+CG whose graphical representation is provided in Figure 9.6. Note how each line conforms to the rule

given above:

Entity > Physical, Abstract.
Physical > Object, Process, Property.
Object > Animate, Inanimate.
Animate > Human, Animal, Plant.
Property > Juvenile, Adult, Gender, Yellow, White.

We now explain this example. The example starts at the top of the lattice with Entity. This super-type is then declared to have two immediate sub-types: *Physical*, and *Abstract*. The *Abstract* node is not associated with any subtype, and so remains a leaf node. The *Physical* node is given three immediate subtypes: *Object*, *Process*, *Property*, each of them being associated with subtypes. These subtypes may also have subtypes, and so on down the lattice.

Another important characteristic of agent archetypes is the "'*multi-inheritance*"' property which allows an agent type to belong to two (or several) different agent archetypes and hence to inherit from their characteristics. Let us consider the following example.

Adult > Woman, Man.
Young > Girl, Boy.
Female > Woman, Girl.
Male > Man, Boy.

Let us notice that *Woman* occurs at several places. This is allowed, as long as there is no circularity (i.e., as long as a type is not specified to be a subtype of itself) whether immediately or indirectly.

There is a fundamental difference between an *archetype* on the one hand, and *instances* of that type on the other hand. For example, while *SchoolBus* is an archetype, *SchoolBus1* and *SchoolBus2* are instances of that archetype. Instances are members of the group of entities which is named by the archetype. The archetype is the name of the group.

In Prolog+CG, we have two ways of saying that a type has an instance: (1) we can simply declare it as an individual in the referent of some CG; (2) we can declare it at the top of the program in a catalog of individuals. A catalog of individuals for a given type is written as follows: *Archeype = Instance1, Instance2, ..., InstanceN.*

Here, a number of instances (on the right-hand side) are declared to be instances of a

specific archetype (on the left-hand side). This rule is then repeated for each archetype that has instances. For example:

Entity > *Vehicle*.
Vehicle > *Bus*, *Boat*.
Bus > *SchoolBus*, *CityBus*.
SchoolBus = *SchoolBus1*, *SchoolBus2*.
CityBus = *CityBus1*, *CityBus2*.
Boat = *OasisOfTheSeas*, *Titanic*.

This example starts with a semantic type-hierarchy introducing the *Entity*, *Vehicle*, *SchoolBus* and *CityBus* agent archetypes (specified with >). This declares our archetypes. The *SchoolBus* archetype has two instances: *SchoolBus1* and *SchoolBus2*. The *CityBus* archetype has two instances: *CityBus1* and *CityBus2*. Also, the *Boat* archetype also has two instances: *OasisOfTheSeas* and *Titanic*.

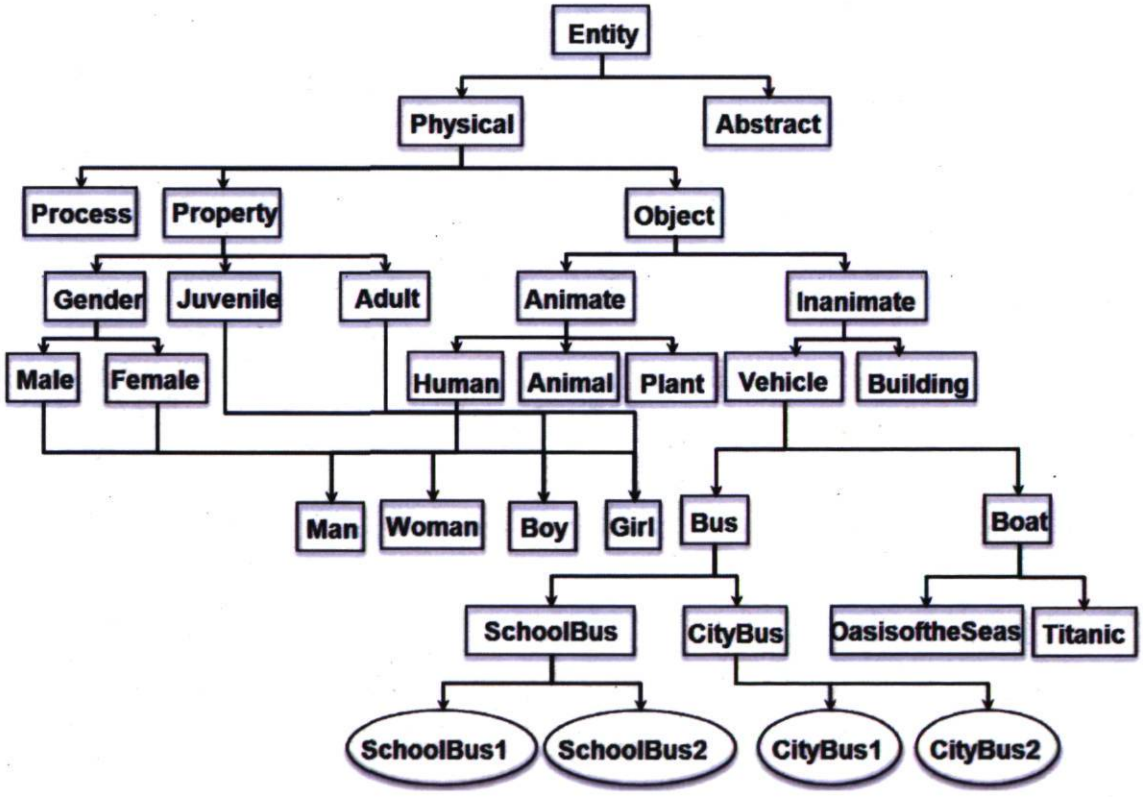


Figure 9.6: A graph of *Semantic Type Lattice* with instances attached to agents archetypes (circle shapes).

9.3.2 Action Archetypes

Since our research addresses the simulation of spatial behaviours, it has been influenced by some basic tenets of *activity theory* [BBC+96]. In particular, our approach to manage environment knowledge rests on the commitments in activity theory that: (1) activities are directed toward objects, zones, or actors [KNM99]; (2) activities are hierarchically structured; and (3) activities capture some context-dependence of the meaning of information [BBC+96];

Theoretically, the common philosophy between our approach and activity theory is a view of the environment from the perspective of an agent interacting with it. Practically, we borrowed from activity theory two main ideas: (1) the semantics of activities and objects are inseparable [KNM99]; and (2) activities, objects as well as agents are hierarchically structured [BBC+96].

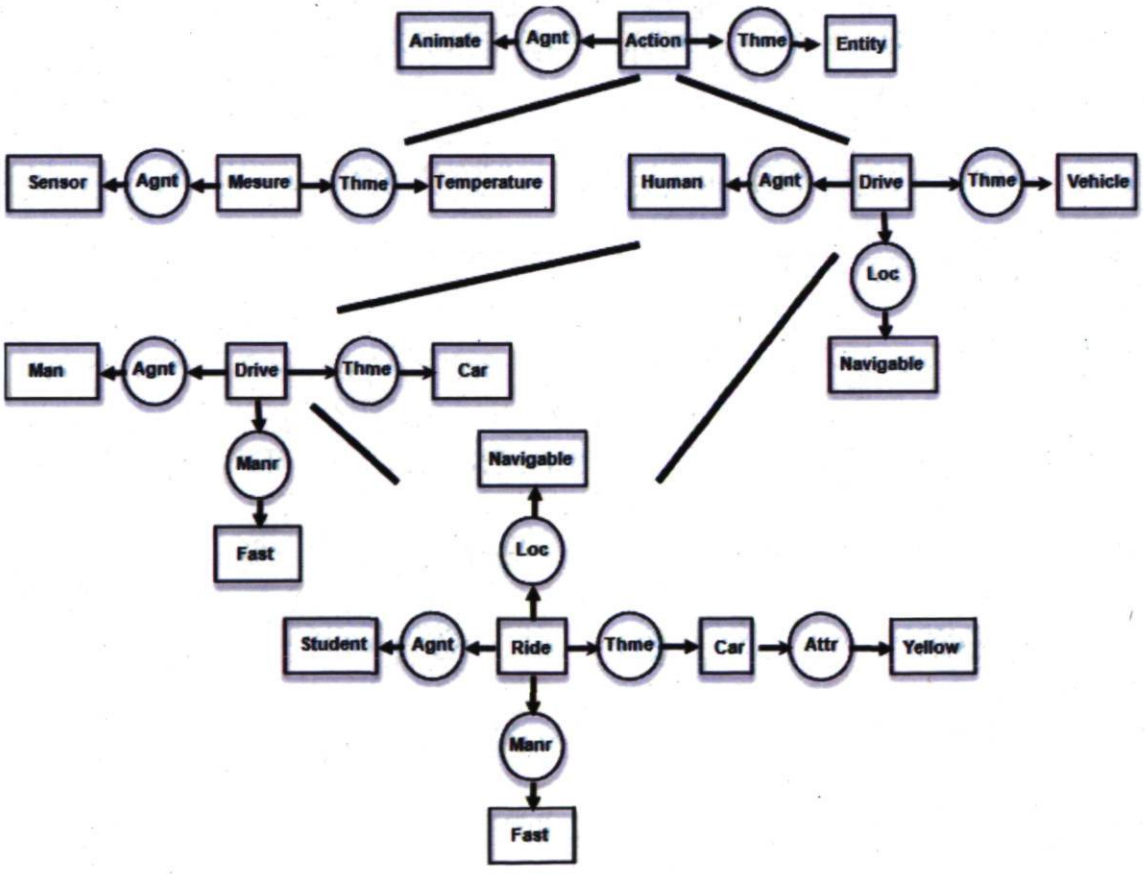


Figure 9.7: A lattice of action archetypes describing spatial situations which involve various agent archetypes. Bold lines point out the specification of spatial situations.

We define an action archetype as a pattern of activities which are associated with agent archetypes (Figure 9.7). Hence, an action archetype describes a situation which involves one

or several agent archetypes. We define an action archetype as a lattice of actions.

Let us consider the *Vehicle* agent archetype that we introduced in Section 9.3.1. Let us define the action archetype *Drive* that we associate with the *Vehicle* agent archetype as follows: "'*an agent *m, which is a man, drives fast an object *c, which is a car*"'. This action archetype is expressed using the following CG:

[DRIVE:*d]
-(agnt)->[MAN:*m]
-(obj)->[CAR: *c]
-(manr)->[FAST]

And since the above description is equal or more specific than the following action archetype:

[DRIVE:*d]
-(agnt) [HUMAN:*h]
-(obj)->[VEHICLE: *v]
-(loc)->[NAVIGABLE]

Then, it can be inferred, by deduction, that:

[DRIVE:*d]
-(agnt)->[MAN:*m]
-(obj)->[CAR: *c]
-(manr)->[FAST]
-(loc)->[NAVIGABLE]

And since *Student* is a sub-type of the archetype *Man*, the following description is also valid:

[DRIVE:*d]
-(agnt)->[STUDENT:*s]
-(obj)->[CAR: *c]
-(attr)->[YELLOW]
-(manr)->[FAST]
-(loc)->[NAVIGABLE]

Hence, using the action archetype lattice and considering the fact that student is a subtype of man which itself is a subtype of human, we are able to deduce the following assump-

tion: "'*an agent of type student drives fast an object of type car whose color is yellow*"'. Using agent and action archetype hierarchies we are able to describe situations and infer by deduction a new ones.

9.3.3 Agentification of Geographic Features

The MAGS paradigm usually involves different agents representing real world actors of various kinds. These actors may represent humans, cars, animals, etc. They may also represent geographic features and spatial phenomena occurring in the physical world. Examples of agents representing a geographic feature include spatial areas such as parks, blocks, land parcels, city, etc. Moreover, examples of agents representing spatial phenomena include fog, smog, fire, etc.

Agentification is the process of identifying and modelling spatial agents associated with geographic features of the informed virtual geographic environment. Our agentification process deals with two categories of geographic features: (1) geographic features associated with the exact space decomposition that we call *CellAgent*; (2) geographic features freely modelled by MAGS users that we call *FloatingAgent*.

A *CellAgent* may be thought of as a footprint of a geographic feature representing a surface within the geographic environment. Since the *CellAgent* is linked to the space decomposition, the agentification process associates the spatial agent $CellAgent_A$ with a set of n cells as follows:
$CellAgent_A = cell_1, cell_2, ..., cell_n$

The agentification process is also responsible for the computation of various geometric and topologic characteristics associated with the created agent such as boundaries, surface, convex hull, center, etc. The agentification process may occur at any time during the MAGS simulation process. MAGS users are able to specify when to trigger the agentification process while creating the MAGS scenario.

Although the *FloatingAgent* is not associated with cells resulting from the exact space decomposition process, it is still represented within the IVGE using convex geometric volumes defined by MAGS users. Agents representing these volumes may be thought of as floating volumes overlaying the spatial decomposition of the geographic environment. It is hence easy to extrapolate the center of any of these volumes in order to determine its precise location within the IVGE. Although FloatingAgent may be mobile or stationary, CellAgent may only be stationary since this latter is closely tied to the spatial decomposition of the ge-

ographic environment. When dealing with mobile CellAgent or FloatingAgent, it is easy for agents evolving in the IVGE to perceive these agentified geographic features and to react to them. Indeed, the CellAgent and FloatingAgent may or may not be specified as obstacles to other agents evolving in the IVGE.

9.4 Results and Case Studies

In this section, we present two case studies that illustrate how the IVGE model and the proposed knowledge management approach are used in practice.

9.4.1 Case Study 1: Human agents taking buses

This case study aims to illustrate how spatial agents representing humans leverage the environment knowledge management approach that we propose. In order to acquire knowledge about the environment and to reason about it, spatial agents apprehend the virtual environment and make decisions according to their types and capabilities and taking into account its characteristics. In this example, a few human agents representing students and workers interact with the IVGE and our EKB in order to plan their path using a bus to get to their final destinations (university and office). This case study also involves a few agents of type *CellAgent* representating bus stations.

Let us consider three agent archetypes: *Bus*, *Student*, and *Worker* and several action archetypes including *STOP*, *GO*, *GETIN*, *WALK* and *ROLL*. The Bus archetype represents the different kinds of buses including city buses, school buses, etc. The Student archetype includes schoolchildness, pupils, students, etc. The Worker archetype represents working persons. This case study involves an informed virtual geographic environment representing a part of Quebec City (Figure 9.8). An environment knowledge base (EKB) is created using the Amine platform. In this EKB, we first specified the different semantic information that qualify our virtual urban environment. The GIS data used to build the informed virtual geographic environment of Quebec City have been introduced in Figure 6.3. Second, we specified the above introduced agent archetypes namely, *BUS*, *STUDENT*, and *WORKER*. Third, we agentified the geographic feature (a polygon) which represents the bus station as a *CellAgent* type. Two IVGE instances are specified: (1) *HUMANNAV* representing a view of the IVGE including the different geographic zones on which an agent of type human can move;(2) *VEHICLENAV* representing a view of the IVGE including the different geographic zones on which an agent of type vehicle can move. Besides, we specify the following facts:

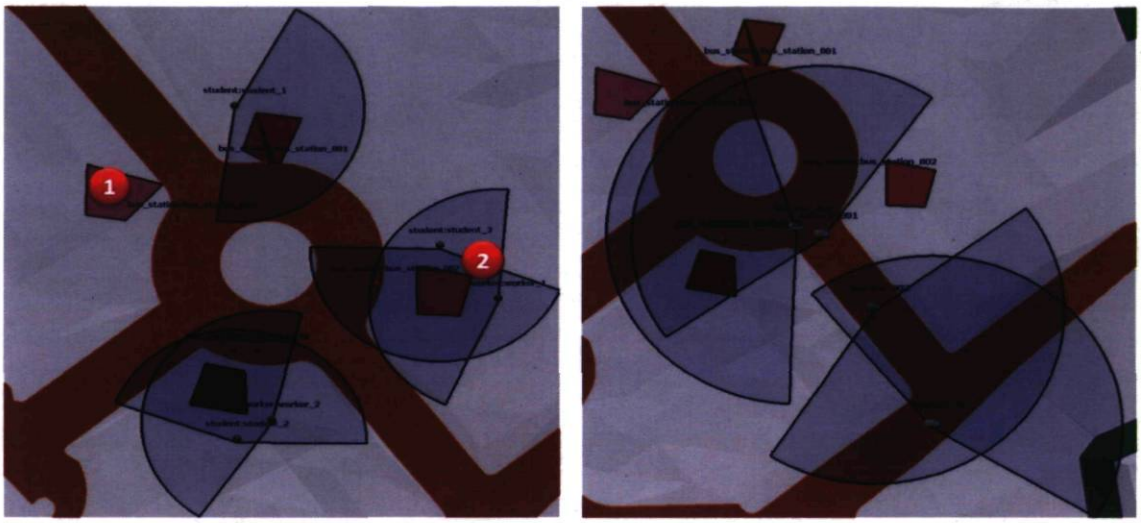
Figure 9.8: The IVGE where the simulation takes place: (1) 4 departure bus stations each dedicated to a specific bus line; (2) final bus station near by the city hall building; (3) bus station near by the Teluq building; (4) bus station near by a Laval University's building; (5) bus station near by a governmental building.

students and workers use buses to respectively reach universities and work places; humans walk on human navigable zones; vehicles roll on vehicle navigable zones; buses stop at stations.

cg([STUDENT]<-agnt-[GETIN]-obj->[BUS]).
cg([HUMAN]<-agnt-[WALK]-loc->[HUMANNAV]).
cg([STUDENT]<-agnt-[GO]-loc->[UNIVERSITY]).
cg([VEHICLE]<-agnt-[ROLL]-loc->[VEHICLENAV]).
cg([WORKER]<-agnt-[GETIN]-obj->[BUS]).
cg([WORKER]<-agnt-[GO]-loc->[WORKPLACE]).
cg([BUS]<-agnt-[STOP]-loc->[STATION]).

In addition, two instances of buses, two instances of stations, and two instances of destinations are defined: Bus1, Bus2, Station1, Station2, *w*, and *u* . Bus1 which stops at station1 goes to the workplace *w*. Bus2 which stops at station2 goes to the university *u*.

cg([BUS: Bus1]<-agnt-[GO]-loc->[WORKPLACE:w]).
cg([BUS: Bus2]<-agnt-[GO]-loc->[UNIVERSITY: u]).
cg([BUS: Bus1]<-agnt-[Stop]-loc->[STATION: Station1]).
cg([BUS: Bus2]<-agnt-[Stop]-loc->[STATION: Station2]).



(a) (b)

Figure 9.9: Stations, student and worker passengers, and buses: (a) (1) an agentified group of cells representing a bus station of type *CellAgent*; (2) 3 students and 2 workers agents (green icon); (b) 4 agents of type *Bus* approaching the stations. Agents either students, workers, or buses are associated with their respective perception fields (an angle of 160°) which are highlighted in blue. Notice how the bus perception field is larger than human agents ones.

Now that the agent archetypes are specified, and the facts which characterise their instances are defined, we carry out the simulation in which two agents of type *student* and three agents of type *worker* interact with the IVGE in which they evolve in order to localise the appropriate station from which they can catch the right bus to reach their final destinations. For simplification purposes, agents of type bus follow a pre-defined computed paths (Figure 9.9(b)). Agents of type student and worker start by identifying their own locations within the IVGE. Next, they interrogate the EKB in order to know which bus they should take in order to reach their final destinations (Figure 9.9(a)). The student agent asks the following query: "'which bus goes to the university?"'

?- cg([?]<-agnt-[GO]-loc->[UNIVERSITY]).

The answer provided by the Amine platform is:

x = Bus2;

Then, the student agent asks the following query:"'where does Bus2 stop at?"'

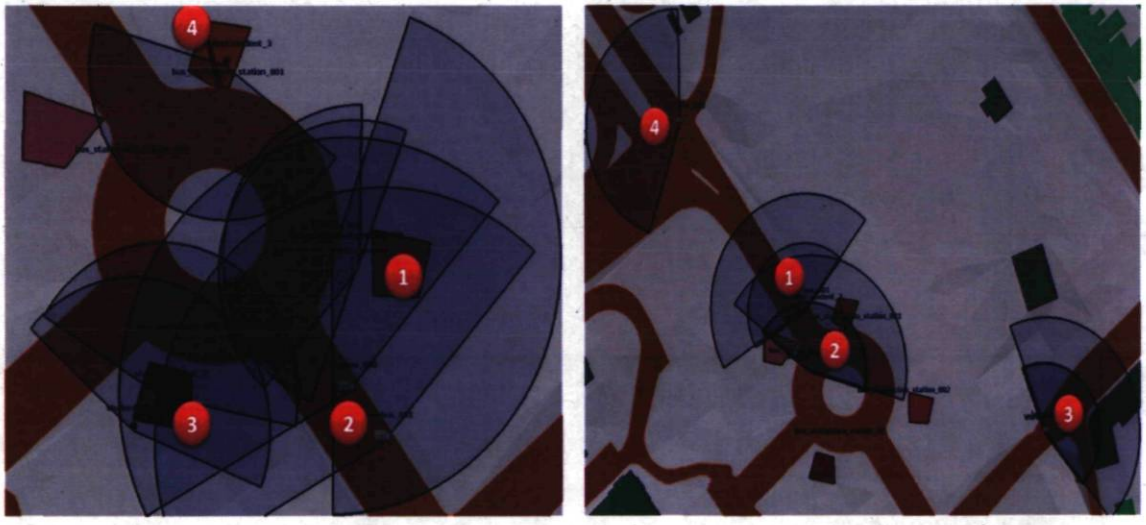
?- cg([BUS: Bus2]<-agnt-[STOP]-loc->[?]).

The answer provided by the Amine platform is:

x = Station2;

Once the answer is provided by the Amine platform, agents plan paths using this semantic description as detailed in Section 8.2. Agents move towards the appropriate bus station, then wait for the bus (Figure 9.10). Since our agents are endowed with perception capabilities, they are able to detect when a bus arrives at the station. The agent bus is also endowed with the same spatial capabilities and waits at the station until all the agents get in it.

Figure 9.10 depicts the simulation of human agents representing the students and the workers taking the bus in order to get to their final destinations.



(a) (b)

Figure 9.10: Simulation of human agents (students and workers) getting on the bus and moving towards their final destinations.

9.4.2 Case Study 2: A sensor web monitoring a weather storm

This case study simulates a sensor web deployed in an IVGE representing the experimental forest of Montmorency for weather monitoring purposes. It shows how agents adapt their spatial behaviours with respect to the knowledge they acquire from the IVGE using our environment knowledge management system along with their perception capabilities. The objective of the simulated sensor web is to identify and to monitor dynamic phenomena such

as storms within the IVGE. In the following example, a few agent archetypes representing different kinds of sensors are involved. In addition, a few agents of type *FloatingAgent* are also used in order to represent weather conditions and dynamic phenomena.

In order to model the weather monitoring simulation, let us first consider the two following agent archetypes: *ZONE* and *SENSOR*. The *ZONE* agent archetype is of type *FloatingAgent* and represents a geographic area. It is characterised by its surface area and the position of its center. In contrast, the *SENSOR* agent archetype is associated with individual sensors deployed in the informed virtual geographic environment and is characterised by its position. Let us now define the sub-types of the *SENSOR* archetype: *TEMPSENSOR* for temperature measurement, *PRESSENSOR* for atmospheric pressure measurement, *WINDSENSOR* for wind speed and orientation measurement, and *HUMISENSOR* for humidity measurement (Figure 9.11). Using the Amine platform, these archetypes, sub-types and instances are specified as follows:

SENSOR > TEMPSENSOR, PRESSENSOR, WINDSENSOR, HUMISENSOR.
TEMPSENSOR = TEMPSENSOR1, ...,TEMPSENSOR13.
PRESSENSOR = PRESSENSOR1, ...,PRESSENSOR7.
WINDSENSOR = WINDSENSOR1, ...,WINDSENSOR12.
HUMISENSOR = HUMISENSOR1, ...,HUMISENSOR7.

Let us consider the two following agent sub-types: *WEATHERZONE* and *STORMZONE*. Agents of type *WEATHERZONE* are stationary and simplifically represent meteorologic conditions within the geographic area that they cover. Five instances of *WEATHERZONE* are created in order to approximately cover the monitored area (Figure 9.12). In addition to their geometric characteristics, these agents are associated with attributes which characterise the meteorological conditions such as temperature = 18° C, pressure = 1010 hPa, humidity = 30%, and wind = $3km/h$-NW. A single instance of type *STORMZONE* is created to represent the storm phenomenon. Obviously, this agent is mobile and is also associated with attributes characterizing its meteorological characteristics. Using the Amine platform, these archetypes, sub-types and instances are specified as follows:

ZONE > WEATHERZONE, STORMZONE.
STORMZONE = STORMZONE-1.
WEATHERZONE = WEATHERZONE-1,..., WEATHERZONE-5.

When a sensor detects a difference above a certain threshold, it adds a fact in the EKB using the CGs formalism. Consider the following example involving the *sensortemp1* adding

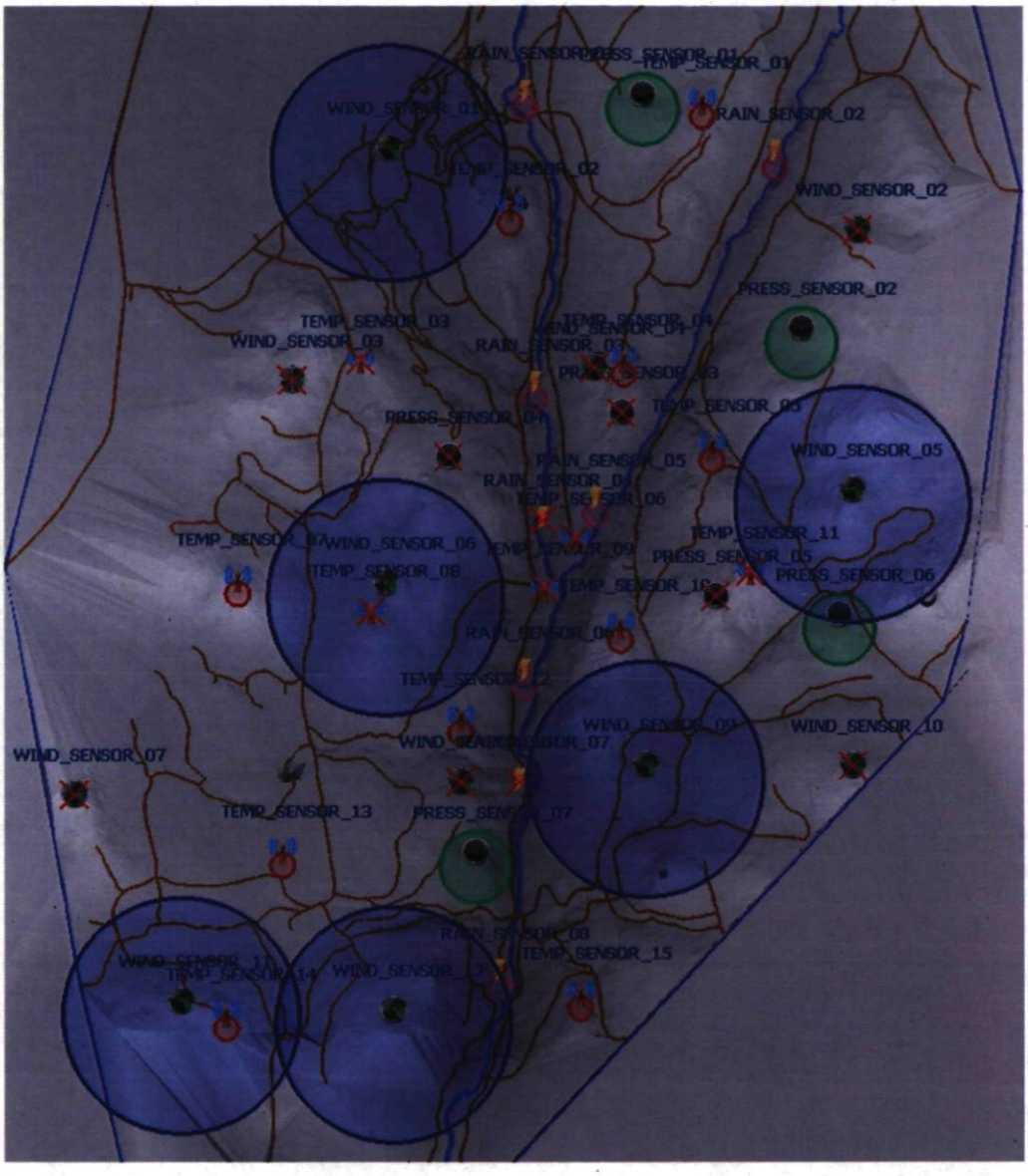


Figure 9.11: Various sensors of different kinds: 13 sensors of type *TEMPSENSOR* (red icon); 12 sensors of type *WINDSENSOR* (blue icon); 7 sensors of type *PRESSENSOR* (green icon); and 7 sensors of type *HUMISENSOR* (purple icon). Idle sensors are marked by a red square icon.

a fact which describes an observed difference of temperature measurement of value 18 at cell 367 at $15h:34:22$.

[MEASUREMENT:*measure1]
-(agnt)->[SENSOR:*sensortemp1]
-(obj)->[TEMPERATURE: 18]
-(time)->[TIMESTAMP: 15h:34:22]

Figure 9.12: 5 agents of type *FloatingAgent* covering approximatively the monitored geographic area of Montmorency experimental forest.

-(loc)->[CELL: 367]

The simulation uses a scenario which describes the simulation model and particularly the number and types of agents as well as their deployment position and the time of their creation. When the simulation starts, sensors are randomly deployed in the IVGE. Each sensor sends a query to the IE in order to determine its deployment position within the IVGE. For example, the sensor agent which is responsible for sensing temperatures asks the Amine platform the following query: "'What is my deployment position?"'.

*?- cg([SENSOR:*sensortemp1]<-agnt-[DEPLOY]-loc->[?]).*

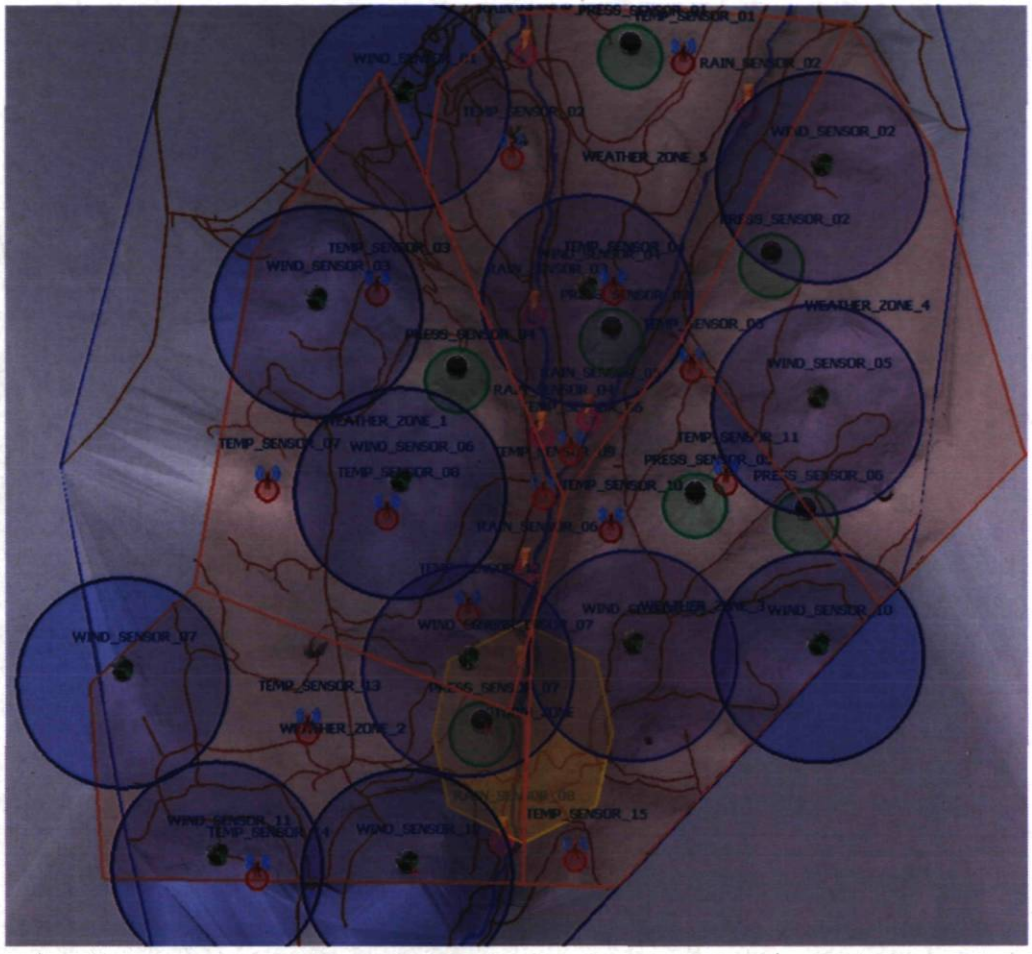


Figure 9.13: The simulation of a sensor web for weather monitoring: the simulated storm appears at the bottom of the figure with a yellow shape.

The answer provided by the Amine platform is:

x = (256109.666667, 5239674, 743.333333333);

Using this response, the sensor agent computes a path in order to reach its deployment position. The computed path takes into account the sensor's capabilities as well as the characteristics of the geographic environment (using an IVGE instance which provides a view of the environment that allows the sensors to move while avoiding obstacles such as dense vegetation and rivers). When sensors reach their final destinations, some of them stay active while other switch to idle in order to preserve the overall energy of the sensor web. Active and idle sensors are specified by users in the simulation scenario. Active sensors make measurements at different frequencies in order to monitor weather conditions. Each kind of sensors is associated with a specific sensing frequency. For example, the sensor agent of type TEMPSENSOR asks the following query: "'What is my initial frequency of measurement?"'.

*?- cg([SENSOR:*sensortemp1]<-agnt-[MEASURE]-manr->[?]).*

The answer provided by the Amine platform is:

x = (2);

The users must specify the initial measurement frequency for each kind of sensors. In this example, sensors of type TEMPSENSOR perform a measurement every 2 seconds. The inference engine using the EKB is responsible for providing each agent instance with the associated measurement frequency by taking into account the agent category (i.e. *TEMPSENSOR*, *PRESSENSOR* , *WINDSENSOR* , and *HUMISENSOR*).

According to the scenario, the simulated storm (represented by an agent of type *FloatingAgent*) appears after a time t after the beginning of the simulation. If an active sensor perceives the storm agent, it directly accesses its properties and extracts the information that it monitors according to its type (i.e. temperature, pressure, wind speed and direction, or humidity). This simulates the capability of the sensor to get the corresponding data in the field. If a difference above a certain threshold Δ is observed, the sensor proceeds as follows: (1) it accelerates its measurement frequency, (2) it notifies the EKB by adding a new fact that keeps track of the event with its timestamp; and (3) it sends a message to wake up idle sensors of the same kind. In order to determine which sensors of the same type are candidate to be waked up, the sensor interrogate the EKB using the IE in order to compute an approximation of the storm direction using the geographic locations of the performed measurements. In order to approximate the direction of the storm, the sensor agent of type TEMPSENSOR asks the following query: "'What are the cells visited by the storm?"'.

?- cg([SENSOR]<-agnt-[MEASUREMENT]-loc->[?]).

The answer provided by the Amine platform is:

x = (2324, 445);

Two cells with IDs 2324 and 445 have been reported by the Amine platform. Using the geometric positions of the centers associated with these cells, the sensor agent can approximate the direction of the storm agent. In order to determine which sensors should be waked up, the sensor agent of type TEMPSENSOR asks the following query: "'Where are the sensors of type temperature located at?"'.

?- cg([TEMPSENSOR]<-agnt-[DEPLOY]-loc->[?]).

The answer provided by the Amine platform is:

x = (226551, 5239651.33333, 646.55666667),
(257133, 52394.33333, 676.666666667),
(257205, 5240119, 716.666666667),
(257837.333333, 5238324.33333, 783.333333333),
(256024, 5238074, 713.333333333),
(258055, 5242722, 793.333333333);

Six sensors of type TEMPSENSOR are located at the position provided by the EKB. Depending on the direction of the storm and on the perimeter of the coverage area that is specified by the user, the sensor agent may weak up all of these sensors, some of them, or no one of them.

As the simulation time progresses and the storm agent evolves in the IVGE, most of initially idle sensors become active to sense the observed phenomenon. When the storm is out of the sensor's perception field, a new difference between the previous and the current measurements is detected by the sensor agent. The sensor agent notifies the EKB by adding a new fact that keeps track of the new event with its timestamp; Idle and active sensors switch states in order to preserve their energy.

In this case study, we illustrated how we create spatial agents using agent and action archetypes. We also provided examples of agentified geographic features (weatherzones and storm). This example puts forward an interesting aspect of the interactions between spatial agents and the EKB. In contrast to the spatial agents in the first case study (Section 9.4.1) which only access the EKB for reading and querying purposes, spatial agents in this second case study access the EKB to add facts describing the evolution of the situation during the simulation process.

9.5 Discussion

The first case study presented the simulation of virtual human in an IVGE. It involved various agent archetypes and highlighted the affordance they provide. This affordance is described using CGs forming what we called an EKB. This EKB is queried using an inference engine to support spatial agents with respect to the virtual environments's and their archetypes' characteristics. This example shows how the *BUS* archetype provides the affordance of transportation to spatial agents of type *STUDENT* and *WORKER*. It also outlines how an agentified geographic feature such as a *STATION* provides affordance of taking the *BUS* for human

agents. The *STATION* contains information about which buses stop there and when. In addition, this agent archetype provides also the affordance of protecting pedestrians from the rain.

More recently, Haddad and Moulin proposed a qualitative model to represent and reason about dynamic phenomena in a geographic space [Had09, HM07]. This model consists in transforming the results of multi-agent geo-simulations into a qualitative representation expressed in terms of spatio-temporal situations. Despite the fact that this model uses the conceptual graphs formalism in order to represent spatio-temporal situations, it focuses on the issue of the analysis of the simulation results rather than on providing spatial agents with knowledge to reason about and to autonomously make decisions with respect to the virtual environment characteristics. Moreover, Haddad and Moulin use a virtual geographic environment which does not take into account the terrain elevation. The description of this virtual geographic environment is based on an approximate spatial decomposition using raster format GIS data. It is exclusively geometric and lacks for topologic and semantic information. It is important to notice the complementarity between the model proposed by Haddad and Moulin and our approach to support spatial agents' decision-making in informed virtual geographic environments. Indeed, our approach provides agents with knowledge about the IVGE in order to let them reason about it and autonomously make decisions that take into account the characteristics of this virtual geographic environment during the simulation process. Using Haddad and Moulin's qualitative model to represent dynamic phenomena would provide MAGS users with the capacity to analyse the simulation results using their causal reasoning approach in order to identify causal relationships between spatio-temporal situations.

9.6 Conclusion

In this chapter, we presented a knowledge management approach which aims to provide spatial agents with knowledge about the environment in order to support their autonomous decision making process. Our approach is influenced by some basic tenets of *activity theory* [BBC+96] as well as by the notion of affordance [Gib79]. It is based on our IVGE model to represent complex and large-scale geographic environments. It uses the *Conceptual Graphs* formalism to represent knowledge about the environment(*Environment Knowledge*) structured as an Environment Knowledge Base (EKB). This approach also includes an inference engine which uses the Prolog+GC language to interrogate, infer and make deductions based on facts, cases, situations, and rules stored within the EKB.

Our environment knowledge management approach is original in various aspects. First, a multi-agent geo-simulation model which integrates an informed virtual geographic environ-

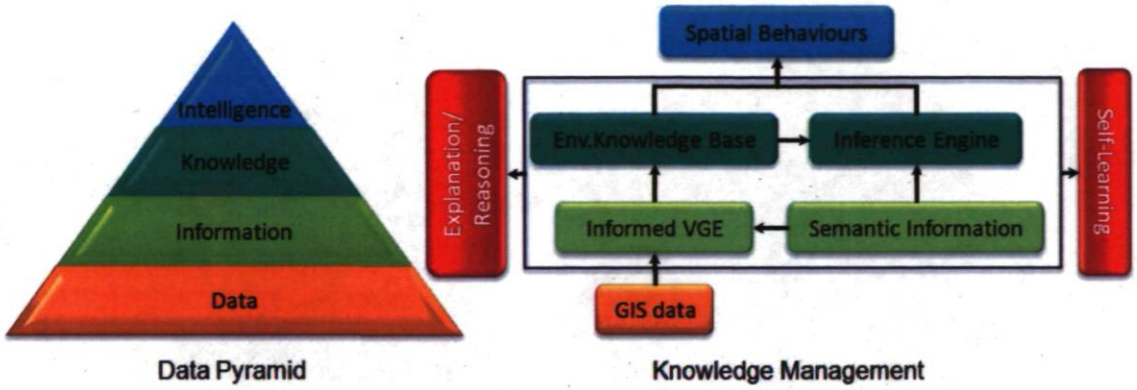


Figure 9.14: Towards a knowledge-oriented multi-agent geo-simulation framework.

ment populated with spatial agents capable of acquiring and reasoning about environment knowledge did not exist. Second, a formal representation of knowledge about the environment using CGs which leverages a semantically-enriched description of the virtual geographic environment has not yet been proposed. Third, providing agents with the capability to reason about a contextualised description of their virtual environment during the simulation is also an innovation that characterises our approach.

As we emphasised in Section 9.5, only a few research work have been found addressing the issue of spatio-temporal situations using standard formalisms. Among which, Haddad and Moulin's model [Had09] constitutes a potential improvement to our approach to support the qualitative analysis of the results of muti-agent geo-simulations. Coupling our environment knowledge management approach with Haddad and Moulin's qualitative model to analyse simulation results would provide an interesting knowledge-oriented multi-agent geo-simulation framework which would support: (1) the simulation of spatial behaviors using knowledge-based spatial agents; (2) the qualitative analysis and explanation of the simulation results; (3) the use of expertise by the agents' decision-making process; and (4) the agents self-learning from cases (i.e. case base reasoning) (Figure 9.14). This is an interesting avenue for future works.

Chapter 10

Conclusion and Perspective

In this chapter, we present the conclusion and the future perspectives of our work on the simulation of agents' spatial behaviours evolving and interacting with an informed virtual geographic environment. First, we provide a synthesis of the models and approaches that we presented throughout this thesis. Next, we highlight the contributions of our works. Then, we discuss and outline the limits and shortcomings of this thesis and present some future works.

10.1 Synthesis

Throughout this thesis we proposed a novel approach to simulate agents' spatial behaviours in virtual geographic environment. First, we proposed a model and a method for the automated generation of virtual geographic environments using standard geographic data provided by geographic information systems. This model allows for the production of an *Informed Virtual Geographic Environment* (IVGE) whose description includes geometric and topologic data, as well as semantic information.

Second, we presented three abstraction techniques in order to support complex and large-scale geographic environments: 1) geometric; (2) topologic; and (3) semantic abstraction. The geometric abstraction process relies on an automated terrain shape qualification process and offers two discretisation approaches: *automated* and *customised* elevation discretisations. In addition, we proposed a *topologic abstraction* process in order to build a *Hierarchical Topologic Graph* (HTG) which allows for modelling a multi-level IVGE representing large-scale and complex geographic environments. Moreover, we enhanced the representation with semantic information using a standard formalism based on *Conceptual Graphs* (CG) and

proposed a *semantic abstraction* process in order to build different semantical views of the IVGE.

Third, we proposed a *Hierarchical Path Planning* (HPP) algorithm and a *Navigation Graph* (NG) model which uses the enriched and hierarchical topologic graph-based description of the IVGE and allows to efficiently support motion planning (path planning and navigation) of spatial agents in informed virtual geographic environments. This HPP algorithm allows agents to plan paths with respect to various constraints, while taking into account both the agents' and the IVGE's characteristics.

Finally, we proposed an approach to manage knowledge about the environment in order to support the agents' spatial behaviours. Our approach extends the enriched description of the IVGE and introduces two components: 1) an *Environment Knowledge Base* (EKB) and; 2) an Inference Engine (IE). The EKB stores the structured and conceptualised descriptions of the geographic features present in the physical world and the affordances they provide to spatial agents (what we call *Environment Knowledge* (EK)). The IE provides agents with the capability to acquire and to reason about environment knowledge. In order to implement this environment knowledge management approach, we used the Amine platform which allows us to describe the environment knowledge base using conceptual graphs and to query this base using the Prolog+CG programming language.

10.2 Contributions

The first contribution of this thesis is a geometrically-precise and semantically-enhanced model of virtual geographic environments based on a graph structure.

The second contribution is a complete method and a set of tools (IVGE-Builder) for the automated generation of informed virtual geographic environments based on reliable geographic data provided by *Geographic Information Systems* (GIS) and using the *Constrained Delaunay Triangulation* (CDT) technique as a spatial decomposition approach.

The third contribution is a geometric, a topologic, and a semantic abstraction processes and a tool (IVGE-Viewer) which are used to build and to visualize multi-level (hierarchical) multiple views (environment instances) of complex and large-scale informed virtual geographic environments.

The fourth contribution is an *Hierarchical Path Planning* (HPP) algorithm coupled with a *Neighborhood Graph* (NG) model for efficient motion planning (navigation and path plan-

ning) capacities of autonomous agents evolving in informed virtual geographic environments.

The fifth and last contribution of this thesis is our environment knowledge management approach to support the autonomous spatio-temporal reasoning and decision-making of spatial agents in informed virtual geographic environments.

From an application perspective, this thesis illustrates the above mentioned theoretical contributions in domains such as wireless communications, virtual humans simulation and sensor webs.

The first illustration is a communication analysis tool which precisely computes path loss and radio signal attenuation due to earth, buildings, and foliage areas' obstructions using the geometrically-precise and semantically-enhanced description of the IVGE which allows the to easily and efficiently compute lines of sight.

In the second illustration, we showed how spatial agents representing humans, buses and bus stations compute paths and leverage the environment knowledge management approach that we propose. Spatial agents use our hierarchical path planning algorithm to compute optimised itineraries in the IVGE using geometric and semantic criteria and benefit from our neighboring graph-based approach for navigation purposes. They also acquire knowledge about the environment from the environment knowledge base and reason about it using the inference engine.

The third illustration is a sensor web management prototype which uses the multi-agent geo-simulation paradigm and our IVGE model in order to carry out geo-simulations of sensor web deployment in informed large-scale virtual geographic environments for the monitoring of dynamic phenomena (i.e. weather conditions).

10.3 Limits and Shortcomings

The first main shortcoming of our approach concerns the exact decomposition of geographic environments. This decomposition is a pre-process which generates a set of convex cells using a reliable GIS data. The resulting convex cells' description is *static* and does evolve during the simulation process. However, when simulating dynamic phenomena, the spatial decomposition needs to be updated at least locally in order to take into account the occurring changes. For example, when simulating a flooding river due to heavy rain, the boundary of the river is expected to change as the simulation evolves. However, our IVGE model in its current version does not support such a dynamic spatial decomposition. A desirable

extension of our IVGE model would be to compute a new spatial decomposition of the geographic environment at run time (i.e. during the simulation process) when such a need arises. However, computing a new spatial decomposition may be a complex process. Indeed, when dealing with geometrically complex or large-scale geographic environments, such a process may suffer from high costs in both time and memory use. However, we might consider the computation of only parts of the IVGE where the dynamic phenomenon occurs.

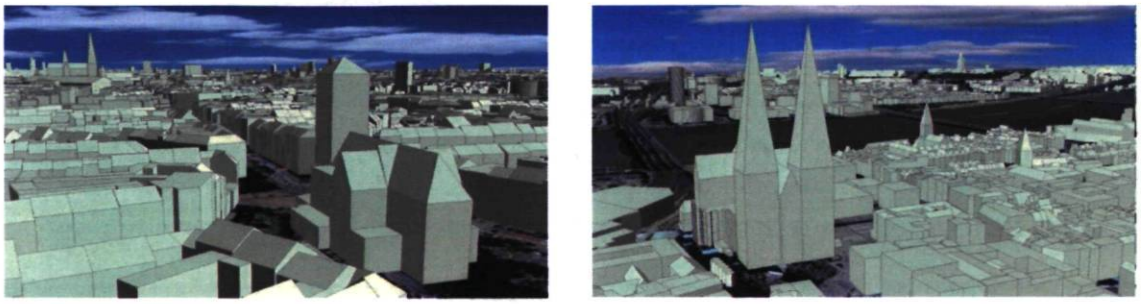
The second limit of our IVGE model concerns the representation of buildings. Currently, buildings are represented using footprint boundaries characterized by their geometric and topologic data and qualified using semantic information. Such a representation is not sufficient if we want to create and visualize realistic and plausible 3D models of urban environments. There is a need for a realistic 3D building reconstruction approach, which produces 3D shapes from existing ground plans (footprints) of buildings extracted from GIS data.

The third limit concerns our environment knowledge management approach. This approach, in its current version, is a proof of concept which demonstrates the capability of our IVGE model to : 1) integrate knowledge about the environment; 2) to allow agents to reason about it using an inference engine. Although the provided scenarios are simplified, they illustrate the advantages of extending our IVGE model by: 1) using a standard knowledge representation formalism (Conceptual Graphs) and; 2) integrating an inference engine such as the Amine platform. When the agent is acting, it uses the environment knowledge base, its observations of the virtual environment, and its goals and abilities to choose what to do and to update its own knowledge. Hence, the environment knowledge base corresponds to the agent's long-term memory, where it keeps the knowledge that is needed to act in the future. This knowledge comes from prior knowledge (provided by MAGS users) and is combined with what is learned from data and past experiences. The beliefs, intentions and desires of the agent correspond to its short-term memory. Although a clear distinction does not always exist between long-term memory and short-term memory, this issue might be addressed as part of the extension of our knowledge management approach. Moreover, there is feedback from the inference engine to the environment knowledge base, because observing and acting in the world provide more data from which the agents can learn. Evolving and allowing the agent model to learn from such data is another challenging task.

10.4 Future Work

We already mentioned several improvements that can be explored in future works. The space decomposition of the IVGE should be dynamic. Its computation should be spatially localized within the IVGE in order to reduce time and memory use costs. Although the work done by

Walendziuk and Kwieckowski [WK06] uses a grid-based space representation, their approach may be of interest to guide the dynamic IVGE exact spatial decomposition. Moreover, our IVGE model may use the approach proposed by Kadaa *et al.* [HBK07, KM09] which constructs $3D$ models by assembling building blocks from a library of parameterised standard shapes. The basis of their work is a $2D$ partitioning algorithm that splits a building's footprint into non-intersecting, mostly quadrangular sections (Figure 10.1). Using Kadaa's approach



(a) (b)

Figure 10.1: 3D city models of East Berlin (a) and Cologne (b).

to model buildings in virtual urban environment would greatly improve the realism of our IVGE.

Regarding our hierarchical path planning (HPP) algorithm, we analysed its complexity and concluded that it performs better than standard path planning algorithms. However, HTG assumes that the hierarchical topologic graph (HTG) already exists. Therefore, we are currently working on two objectives: 1) minimizing the dependency between the HPP and the HTG in order to generalize the use of HPP with other graph-based descriptions of virtual geographic environments; and 2) assessing the efficiency of our HPP algorithm compared with other hierarchical planners such as *Triangulation A** and *Triangulation Reduction A** [DB06].

Another interesting improvement concerns the environment knowledge management approach that we proposed in order to model spatio-temporal situations using agent and action archetypes. We proposed a decision-making process which uses the Amine platform and allows spatial agents to make decisions while taking into account the enriched description of the virtual geographic environment. As we emphasised in Section 9.5, only a few research works have been found addressing the issue of spatio-temporal situations using standard formalisms. Among which, Haddad and Moulin's model [Had09] constitutes a potential improvement to our approach to support the qualitative analysis of the results of multi-agent geo-simulations. Therefore, we are currently examining the possibility to couple of our approach with Haddad and Moulin's qualitative model to represent and reason about dynamic phenomena in a geographic space. More broadly, our future work targets a knowledge-oriented multi-agent geo-simulation framework which will support: 1) the simulation of spatial behaviors using

knowledge-based spatial agents; 2) the qualitative analysis and explanation of the simulation results; 3) the use of expertise by the agents' decision-making process; and 4) the agents' self-learning from cases (Case Base Reasoning (CBR)).

Bibliography

[Aba06] Tolga Abaci. *Object Manipulation and Grasping for Virtual Humans*. PhD thesis, EPFL, Lausanne, 2006.

[ABG05] Rasmus Andersen, Jean Louis Berrou, and Alex Gerodimos. On some limitations of grid-based (ca) pedestrian simulation models. In *Proceedings of the First International Workshop on Crowd Simulation (V-Crowds'05)*. VRlab, EPFL, 2005.

[ABL03] Dominique Attali, Jean-Daniel Boissonnat, and André 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.

[ACF01] Okan Arikan, Stephen Chenney, and D. A. Forsyth. Efficient multi-agent path planning. In *Proceedings of the Eurographic Workshop on Computer Animation and Simulation*, pages 151–162, New York, NY, USA, 2001. Springer-Verlag New York, Inc.

[AK00] Franz Aurenhammer and R. Klein. *Handbook of Computational Geometry*, chapter Voronoi diagrams, pages 201 – 290. Elsevier, 2000.

[AM05] Walid Ali and Bernard 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.

[Ang08] Edward Angel. *Interactive Computer Graphics: A Top-Down Approach Using OpenGL (5th Edition)*. Addison Wesley, 5 edition, April 2008.

[AP98] Paul Arthur and Romedi Passini. *Wayfinding: People, Signs, and Architecture*. McGraw-Hill Companies, 1998.

[ASG+01] Peter Allen, Ioannis Stamos, Atanas Gueorguiev, Ethan Gold, and Paul Blaer. Avenue: Automated site modeling in urban environments. In *Proceedings of*

the Third International Conference on 3D Digital Imaging and Modeling, 2001.

[AT05] Tolga Abaci and Daniel Thalmann. Planning with smart objects. In *Proceedings of the 13th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision: WSCG*, pages 25–28, 2005.

[Aur91] Franz Aurenhammer. Voronoi diagrams: A survey of a fundamental geometric data structure. *ACM Computing Surveys*, 23(3), September 1991.

[AZ04] David Arctur and Michael Zeiler. *Designing Geodatabases: Case Studies in GIS Data Modeling*. ESRI Press, 2004.

[Bad06] Marwan Badawi. *Synoptic Objects Describing Generic Interaction Processes to Autonomous Agents in an Informed Virtual Environment*. PhD thesis, INSA, Rennes, December 2006.

[BBA06] Michael Batty, Joana Barros, and Junior Sinesio Alves. *Complexity and Co-Evolution: Continuity and Change in Socio-Economic Systems*, chapter Cities: continuity, transformation and emergence, pages 1–17. Edward Elgar, Cheltenham, UK, 2006.

[BBB+98] Norman Badler, Rama Bindiganavale, Juliet Bourne, Martha Palmer, Jianping Shi, and William Schuler. A parameterized action representation for virtual human agents. In J Cassell, J. Sullivan, S. Prevost, and E. Churchill, editors, *Embodied Conversational Agents*, pages 256–284. MIT Press, 1998.

[BBC+96] Rachel Bellamy, Susanne Bødker, Ellen Christiansen, Yrjö Engeström, Virginia Escalante, Dorothy Holland, Victor Kaptelinin, Kari Kuutti, Bonnie A. Nardi, Arne Raeithel, James Reeves, Boris Velichkovksy, and Vladimir P. Zinchenko. *Context and Consciousness: Activity Theory and Human-Computer Interaction*. The MIT Press, first edition, November 1996.

[BBT99] Christophe Bordeux, Ronan Boulic, and Daniel Thalmann. An efficient and flexible perception pipeline for autonomous agents. In *Eurographics'99*, volume 18, pages 20–30, 1999.

[BF95] Avrim Blum and Merrick Furst. Fast planning through planning graph analysis. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI 95)*, pages 1636–1642, 1995.

[BG01] Blai Bonet and Héctor Geffner. Planning as heuristic search. *Artificial Intelligence*, 129:5–33, 2001.

[Bil07] William Bille. *Conceptual Modeling of Complex Objects for Virtual Environments: A Formal Approach.* PhD thesis, Vrije University of Brussel, Brussel, Belgium, 2007.

[BLA02] Burchan Bayazit, Jyh-Ming Lien, and Nancy M Amato. Roadmap-based flocking for complex environments. In *PG '02: Proceedings of the 10th Pacific Conference on Computer Graphics and Applications*, page 104, Washington, DC, USA, 2002. IEEE Computer Society.

[BLA03] Burchan Bayazit, Jyh-Ming Lien, and Nancy M Amato. Better group behaviors in complex environments using global roadmaps. In *ICAL 2003: Proceedings of the Eighth International Conference on Artificial Life*, pages 362–370, Cambridge, MA, USA, 2003. MIT Press.

[Blu97] Bruce Mitchell Blumberg. *Old Tricks, New Dogs: Ethology and Interactive Creatures.* PhD thesis, Massachusetts Institute of Technology, 1997.

[BMS04] Adi Botea, Martin Müller, and Jonathan Schaeffer. Near optimal hierarchical path-finding. *Journal of Game Development*, 1:7–28, 2004.

[Bol94] Alessandro Bollini. *From Perception to Action Conference*, chapter Action oriented control of perception, pages 368–371. IEEE Computer Society Press, 1994.

[Bra99] Michael E. Bratman. *Intention, Plans, and Practical Reason.* Cambridge University Press, March 1999.

[Bro91] Rodney Brooks. Intelligence without reason. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 569–595. Morgan Kaufmann, 1991.

[BRR+05] Daniel Brown, Rick Riolo, Derek Robinson, Michael North, and William Rand. Spatial process and data models: Toward integration of agent-based models and GIS. *Journal of Geographical Systems*, 7(1):25–47, 2005.

[BT98] Srikanth Bandi and Daniel Thalmann. Space discretization for efficient human navigation. *Computer Graphics Forum*, 17(3):195–206, September 1998.

[BT04] Isaak Benenson and Paul Torrens. *Geosimulation: Automata-Based Modeling of Urban Phenomena.* John Wiley and Sons Inc., 2004.

[BWB+95] Norman Badler, Bonnie Webber, Welton Becket, Christopher Geib, Michael Moore, Catherine Pelachaud, Barry Reich, and Matthew Stone. Planning and parallel transition networks: Animation's new frontiers. In S Shin and

T Kunii, editors, *The Second International Pacific Conference on Computer Graphics and Applications*, pages 101–117, 1995.

[BWB+96] Norman Badler, Bonnie Webber, Welton Becket, Matthew Stone, Christopher Geib, Catherine Moore, Michael Pelachaud, and Barry Reich. *Interactive Computer Animation*. Prentice Hall, 1996.

[BY98] Jean-Daniel Boissonnat and Mariette Yvinec. *Algorithmic Geometry*. Cambridge University Press, New York, NY, USA, 1998.

[CC01] Stephen Chu and Branko Cesnik. Knowledge representation and retrieval using conceptual graphs and free text document self-organisation techniques. *International Journal of Medical Informatics*, 62(2-3):121 – 133, 2001.

[CCM02] Marc Cavazza, Fred Charles, and Steven Mead. Planning characters behaviour in interactive storytelling. *The Journal of Visualization and Computer Animation*, 13(11):121–131, May 2002.

[CGA] Cgal: Computational geometry algorithms library. http://www.cgal.org, last access August 2008.

[Cha87] Bernard Chazelle. *Advances in Robotics 1: Algorithms and Geometric Aspects of Robotics*, chapter Approximation and decomposition of shapes, pages 145–185. Lawrence Erlbaum Associates, Hillsdale, NJ, 1987.

[Chr01] Nicholas Chrisman. *Exploring Geographical Information Systems, Second Edition*. John Wiley and Sons Inc., 2001.

[Cig05] Jan Ciger. *Collaboration with agents in VR environments*. PhD thesis, EPFL, Lausanne, 2005.

[CKB99] Sonu Chopra-Khullar and Norman I. Badler. Where to look? automating attending behaviors of virtual human characters. In *AGENTS '99: Proceedings of the Third Annual Conference on Autonomous Agents*, pages 16–23, New York, NY, USA, 1999. ACM.

[CKP95] James Cremer, Joseph Kearney, and Yiannis Papelis. Hcsm: A framework for behavior and scenario control in virtual environments. *ACM Transactions on Modeling and Computer Simulation*, 5:242–267, 1995.

[CL03] Min Gyu Choi and Jehee Lee. Planning biped locomotion using motion capture data and probabilistic roadmaps. *ACM Transactions on Graphics*, 22:182–203, 2003.

[CMY+06] Pablo de Heras Ciechomski, Jonathan Maïm, Barbara Yersin, Jean-Paul Laumond, Daniel Thalmann, and Julien Pettré. Real-time navigating crowds: scalable simulation and rendering. *Computer Animation and Virtual Worlds*, 17(3-4):445–455, July 2006.

[CN01] Howie Choset and Keiji Nagatani. Topological simultaneous localization and mapping (SLAM): toward exact localization without explicit localization. *IEEE Transactions on Robotics and Automation*, 17(2):125–137, 2001.

[Cod89] Bill Coderre. *Modeling Behavior in Petworld*, chapter Artificial Life: The Proceedings of an Interdisciplinary workshop on the Synthesis and Simulation of Living Systems, pages 407–420. Westview Press, 1989.

[CSSM03] Gary Comparetto, Jonathan Schwartz, Nil Schult, and Jim Marshall. A communications analysis tool set that accounts for the attenuation due to foliage, buildings, and ground effects. In *MILCOM 2003: Proceedings of the Military Communications Conference*, volume 1, pages 1 – 5, Boston, MA, USA, October 2003.

[CW05] Sheng-Gwo Chen and Jyh-Yang Wu. A geometric interpretation of weighted normal vectors and its improvements. In *Proceedings of the International Conference on Computer Graphics, Imaging and Vision: New Trends*, pages 422–425, Beijing, China, 26-29 July 2005.

[DB06] Douglas Demyen and Michael 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.

[DBCVKO08] Mark De Berg, Otfried Cheong, Marc Van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 3rd edition, 2008.

[DF98] Vincent Decugis and Jacques Ferber. Action selection in an autonomous agent with a hierarchical distributed reactive planning architecture. In *AGENTS '98: Proceedings of the second international conference on Autonomous agents*, pages 354–361, New York, NY, USA, 1998. ACM.

[DHK+07] Paul Davidsson, Johan Holmgren, Hans Kyhlbäck, Dawit Mengistu, and Marie Persson. Applications of mult-agent based simulation. *Multi-Agent-Based Simulation VII, Lecture Notes in Computer Science*, 4442:15–27, 2007.

[Dij59] Edsger Dijkstra. A note on two problems in connexion with graph. *Numerische Mathematik*, 1:269–271, 1959.

[DLB96] Brett Douville, Libby Levison, and Norman Badler. Task-level object grasping for simulated agents. *Presence*, 58(4):416–430, 1996.

[Don97] Stephane Donikian. VUEMS: A virtual urban environment modeling system. In *Proceedings of the Conference on Computer Graphics International (CGI'97)*, pages 84–92, Washington, DC, USA, June 23-27 1997.

[Don01] Stéphane Donikian. HPTS: a behaviour modelling language for autonomous agents. In *AGENTS '01: Proceedings of the fifth international conference on Autonomous agents*, pages 401–408, New York, NY, USA, 2001. ACM.

[DSM06] Luiz Gonzaga Da Silveira and Soraia Raupp Musse. Real-time generation of populated virtual cities. In *VRST '06: Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, pages 155–164, New York, NY, USA, 2006. ACM.

[DT02] Pavlin Dobrev and Kristina Toutanova. CGWorld - architecture and features. In *ICCS '02: Proceedings of the 10th International Conference on Conceptual Structures*, pages 261–270, London, UK, 2002. Springer-Verlag.

[EHN94] Kutluhan Erol, James Hendler, and Dana Nau. UMCP: A sound and complete procedure for hierarchical task-network planning. In *AIPS'94: Proceedings of the Second International Conference on Artificial Intelligence Planning Systems*, pages 249–254, 1994.

[EHN95] Kutluhan Erol, James Hendler, and Dana Nau. Complexity results for HTN planning. *Annals of Mathematics and Artificial Intelligence*, 18(1):69–93, Mars 1995.

[Far01a] Nathalie Farenc. *An Informed Environment for Inhabited City Simulation.* PhD thesis, École Polytechnique Fédérale de Lausanne, 2001.

[Far01b] Gerald Farin. *Curves and Surfaces for CAGD.* Kaufmann, Morgan, 2001.

[FBR01] Andrew Frank, Steffen Bittner, and Martin Raubal. Spatial and cognitive simulation with multi-agent systems. In D Montello, editor, *Spatial Information Theory - Foundations of Geographic Information Science, Proceedings of COSIT 2001, Morro Bay, CA, USA, September 2001*, volume 2205 of *Lecture Notes in Computer Science*, pages 124–139, Berlin, Heidelberg, New York, 2001. Springer.

[FBT99] Nathalie Farenc, Ronan Boulic, and Daniel 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.

[FG96] Stan Franklin and Art Graesser. Is it an agent, or just a program?: A taxonomy for autonomous agents. In *Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages*, pages 21–35. Springer-Verlag, 1996.

[FMJ06] Philippe Fuchs, Guillaume Moreau, and Tisseau Jacques. *Le traité de la réalité virtuelle*, chapter Introduction à la réalité virtuelle, pages 3–32. Collection sciences mathématiques et informatiques, 2006.

[FN71] Richard E. Fikes and Nils J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3-4):189–208, 1971.

[For97] Steven Fortune. Voronoi diagrams and Delaunay triangulations. *Handbook of Discrete and Computational Geometry*, pages 377–388, 1997.

[FP93] Tsung-Pao Fang and Les A. Piegl. Algorithm for constrained delaunay triangulation. *Visual Computing*, 10(5):255–265, 1993.

[FR02] Stewart Fortheringham and Peter Rogerson, editors. *Spatial Anaysis and GIS's*. Taylor & Francis, LTD, 2002.

[FRMS+98] Nathalie Farenc, Soraia Raupp Musse, Eric Schweiss, Michael Kallmann, O Aune, Ronan Boulic, and Daniel Thalmann. One Step Towards Virtual Human Management for Urban Environment Simulation. In *Proceedings of the Workshop on Intelligent User Interfaces*, 1998.

[FSK+99] Nathalie Farenc, Elsa Schweiss, Marcelo Kallmann, Olivier Aune, Ronan Boulic, and Daniel Thalmann. A paradigm for controlling virtual humans in urban environment simulations. *Applied Artificial Intelligence*, 14:69–91, 1999.

[Fun98] John David Funge. *Making Them Behave: Cognitive Models for Computer Animation*. PhD thesis, University of Toronto, Toronto, Ont., Canada, 1998.

[GA05] Mario Gutérrez Alonso. *Semantic Virtual Environments*. PhD thesis, École Polytechnique Fédérale de Lausanne,Swisse, 2005.

[GBR+95] John Granieri, Welton Becket, Barry Reich, Jonathan Crabtree, and Norman Badler. Behavioral control for real-time simulated human agents. In *SI3D '95: Proceedings of the 1995 Symposium on Interactive 3D Graphics*, pages 173–180, New York, NY, USA, 1995. ACM.

[GD00] Thomas Gwenola and Stéphane Donikian. Modelling virtual cities dedicated to behavioural animation. *Computer Graphics Forum*, 19:71–80, September 2000.

[GDB+98] Simon Goerger, Rudolph Darken, Mark Boyd, Todd Gagnon, Stewart Liles, Joseph Sullivan, and John Lawson. Spatial knowledge acquisition from maps and virtual environments in complex architectural spaces. In *Proceedings of the 16th Applied Behavioral Sciences Symposium*, pages 6–10, Colorado, USA, 22-23 April 1998. U.S. Air Force Academy.

[Gib79] James Jerome Gibson. *The Ecological Approach to Visual Perception*. London: Lawrence Erlbaum Associates, 1979.

[GM03] Phil Graniero and H Miller. Real-time, wireless field data acquisition for spatial data infrastructures. In *GeoTec Event 2003*, Vancouver BC, Canada, March 2003.

[Goo06] Michael F. Goodchild. GIS and disasters: Planning for catastrophe. *Computers, Environment and Urban Systems*, 30(3):227 – 229, 2006.

[Gra04] Phil Graniero. A spatial analysis of gps availability and radio data transmission reliability for real-time, wireless gis updating in urban environments. Technical report, MEMF Lab, University of Windsor, Dept. of Earth Sciences, 2004.

[GRM09] Alejandra Garcia Rojas Martinez. *Semantics for Virtual Humans*. PhD thesis, École Polytechnique Fédérale de Lausanne, 2009.

[GRW08] Dan Guo, Bo Ren, and Cheng Wang. *Advances in Computation and Intelligence*, volume 5370/2008, chapter Integrated Agent-Based Modeling with GIS for Large Scale Emergency Simulation, pages 618– 625. 2008.

[GS98] David Genest and Eric Salvat. A platform allowing typed nested graphs: How CoGITo became CoGITaNT (research note). In Marie-Laure Mugnier and Michel Chein, editors, *Lecture Notes in Computer Science*, volume 1453, pages 154–164. Springer, 1998.

[Had09] Hedi Haddad. *Une approche pour supporter l'analyse qualitative des suites d'actions dans un environnement géographique virtuel et dynamique:*

l'analyse "What-If" comme exemple. PhD thesis, Department of Computer Science and Software Engineering, Laval University, 2009.

[HB05] Wesley Huang and Kristopher Beevers. Topological map merging. *International Journal of Robotics Research*, 24(8):601–613, 2005.

[HB08] Daniel Harabor and Adi 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.

[HBK07] Norbert Haala, Susanne Becker, and Martin Kada. Cell decomposition for building model generation at different scales. In *Urban Remote Sensing Joint Event*, pages 1–6, 2007.

[HBL05] Kris Tim Hauser, Tim Bretl, and Jean-claude Latombe. Non-gaited humanoid locomotion planning. In *Humanoids*, pages 7–12, Tsukuba, Japan, December 2005.

[HC07] Kazumasa Hanaoka and Graham P. Clarke. Spatial microsimulation modelling for retail market analysis at the small-area level. *Computers, Environment and Urban Systems*, 31(2):162–187, 2007.

[HD04] Svetlana Hensman and John Dunnion. Using linguistic resources to construct conceptual graph representation of texts. In *Text, Speech and Dialogue*, pages 81–88. Springer Berlin / Heidelberg, 2004.

[HGL+02] Biqing Huang, Hongmei Gou, Wenhuang Liu, Yu Li, and Min Xie. A framework for virtual enterprise control with the holonic manufacturing paradigm. *Computers in Industry*, 49(3):299–310, 2002.

[HKLC98] Haddad Haddad, Maher Khatib, Simon Lacroix, and Raja 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.

[HM07] Hedi Haddad and Bernard Moulin. Using cognitive archetypes and conceptual graphs to model dynamic phenomena in spatial environments. In Uta Priss, Simon Polovina, and Richard Hill, editors, *ICCS'07: Proceedings of the 15th International Conference on Conceptual Structures: Knowledge Architectures for Smart Applications*, pages 69–82, Sheffield, UK, July 22-27 2007. Springer.

[HNR72] Peter Hart, Nils Nilsson, and Bertram Raphael. Correction to a formal basis for the heuristic determination of minimum cost paths. *SIGART Newsletter*, 37:28–29, 1972.

[HNR07] Peter Hart, Nils Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, February 2007.

[HWS02] Steven Harper, James Westervelt, and Anne-Marie. Shapiro. *Integrating geographic information systems and agent-based modeling techniques for simulating social and ecological processes*, chapter Management application of an agent-based model: Control of cowbirds at the landscape scale, pages 105–124. Oxford: Oxford University Press, 2002.

[JOH07] Petra Jansen-Osmann and Martin Heil. The process of spatial knowledge acquisition in a square and a circular virtual environment. *Advances in Cognitive Psychology*, 3(3):389–397, 2007.

[JRMJ05] Jacquie Jarvis, Ralph Rönnquist, Duncan McFarlane, and Lakhmi Jain. A team-based holonic approach to robotic assembly cell control. *Journal of Network and Computer Applications*, 29(2-3):160–176, 2005.

[Kab06] Adil Kabbaj. Development of intelligent systems and multi-agents systems with Amine platform. In Henrik Schärfe, Pascal Hitzler, and Peter Øhrstrøm, editors, *ICCS'06: Proceedings of the 14th International Conference on Conceptual Structures*, volume 4068 of *Lecture Notes in Computer Science*, pages 286–299. Springer, 2006.

[Kal01] Marcelo Kallmann. *Object Interaction in Real-Time Virtual Environments*. PhD thesis, École Polytechnique Fédérale de Lausanne, 2001.

[KB91] Benjamin Kuipers and Y.T. Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Journal of Robotics and Autonomous Systems*, 8:47–63, 1991.

[KB05] Adil Kabbaj and Soudi Amir Bouzouba, Karim. Amine platform: an artificial intelligence environment for the development of intelligent systems. In *First Information and Communication Technologies International Symposium ICTIS'05*, Tetuan, Morocco,, June 2005.

[Kel62] Joseph Keller. Geometric theory of diffraction. *Journal of the Optical Society of America*, 52:162–130, 1962.

[Kha86] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research*, 5(1):90–98, 1986.

[KKK04] Athanasios Karalopoulos, Margarita Kokla, and Marinos Kavouras. Geographic knowledge representation using conceptual graphs. In *Proceedings of the 7th Agile conference on Geographic Information Science*, Heraklion, Greece, April 29- 1st May 2004.

[KKK05] Athanasios Karalopoulos, Margarita Kokla, and Marinos Kavouras. *GeoSpatial Semantics*, volume 3799/2005 of *Lecture Notes in Computer Science*, chapter Comparing Representations of Geographic Knowledge Expressed as Conceptual Graphs, pages 1–14. November 2005.

[KKL98] Lydia E. Kavraki, Mihail N. Kolountzakis, and Jean-Claude Latombe. Analysis of probabilistic roadmaps for path planning. In *IEEE Transactions on Robotics and Automation*, volume 14, February 1998.

[KLF04] Sven Koenig, Maxim Likhachev, and David Furcy. Lifelong planning A*. *Artificial Intelligence Journal*, 155(1):93–146, 2004.

[KM09] Martin Kadaa and Laurence McKinleyb. 3D building reconstruction from lidar based on a cell decomposition approach. In U Stilla and N. Rottensteiner, F.and Paparoditis, editors, *CMRT09-ISPRS Workshop, Object Extraction for 3D City Models, Road Databases and Traffic Monitoring Concepts, Algorithms and Evaluation*, pages 47–52, 2009.

[KNM99] Victor Kaptelinin, Bonnie Nardi, and Catriona Macaulay. The activity checklist: a tool for representing the "'space"' of context. *Interactions Magazine*, 6(4):27–39, 1999.

[Koe67] Arthur Koestler. *The Ghost in the Machine*. London, 1967.

[Kor85] Richard E. Korf. Depth-first iterative-deepening: An optimal admissible tree search. *Artificial Intelligence*, 27:97–109, 1985.

[KP74] R.G Kouyouijian and P.H. .Pathak. A uniform geometric theory of diffraction for an edge in perfectly conducting surface. In *IEEE*, volume 62, pages 1448–1461, 1974.

[KS02] Ansgar Kirchner and Andreas Schadschneider. Simulation of evacuation processes using a bionics-inspired cellular automaton model for pedestrian dynamics. *Physica A*, 312:260–276, 2002.

[KSLO96] Lydia Kavraki, Petr Svestka, Jean-Claude Latombe, and Mark Overmars. Probabilistic roadmaps for path planning in high dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

[KT98] Marcelo Kallmann and Daniel Thalmann. Modeling objects for interaction tasks. In *The 9th Eurographics Workshop on Animation and Simulation (EGCAS)*, pages 73–86, Lisbon, Portugal, 1998.

[Lai01] John E. Laird. It knows what you're going to do: adding anticipation to a quakebot. In *AGENTS '01: Proceedings of the Fifth International Conference on Autonomous Agents*, pages 385–392, New York, NY, USA, 2001. ACM.

[Lak90] George Lakoff. *Women, Fire, and Dangerous Things*. University Of Chicago Press, April 1990.

[Lam03] Fabrice Lamarche. *Humanoïdes virtuels, réaction et cognition : une architecture pour leur autonomie*. PhD thesis, Université Rénnes 1, 2003.

[Lat91] Jean-Claude Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Norwell, MA, USA, 1991.

[LaV06] S.M. LaValle. *Planning Algorithms*. Cambridge University Press., Cambridge, 2006.

[LB09] Hui Lin and Michael Batty, editors. *Virtual Geographic Environments*. Science Press, China, 2009.

[LD01] Fabrice Lamarche and Stéphane Donikian. The orchestration of behaviours using resources and priority levels. In *Proceedings of the Eurographic Workshop on Computer Animation and Simulation*, pages 171–182, New York, NY, USA, 2001. Springer-Verlag.

[LD02] Fabrice Lamarche and Stephane Donikian. Automatic orchestration of behaviours through the management of resources and priority levels. In *AAMAS'02: Proceedings of the First International Conference on Autonomous Agents and Multiagent Systems*, pages 15–19. ACM, 2002.

[LD04] Fabrice Lamarche and Stephane Donikian. Crowds of virtual humans: a new approach for real time navigation in complex and structured environments. *Computer Graphics Forum, Eurographics'04*, 2004.

[Lea92] Geoff Leach. Improving worst-case optimal delaunay triangulation algorithms. In *Proceedings of the 4th Canadian Conference on Computational Geometry*, page 15, 1992.

[LG89] Douglas Lenat and Ramanathan Guha. *Building Large Knowledge-Based Systems; Representation and Inference in the Cyc Project*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[LGMR02] Paul Longley, Michael Goodchild, David Maguire, and David Rhind. *Geographic Information Systems and Science*. ESRI Press, 2002.

[Lin82] Andrzej Lingas. *Automata, Languages and Programming*, chapter The power of non-rectilinear holes, pages 369–383. 1982.

[LMM03] Celine Loscos, David Marchal, and Alexandre Meyer. Intuitive crowd behaviour in dense urban environments using local laws. In *TPCG '03: Proceedings of the Theory and Practice of Computer Graphics 2003*, page 122, Washington, DC, USA, 2003. IEEE Computer Society.

[LNR87] John Laird, Allen Newell, and Paul Rosenbloom. SOAR: an architecture for general intelligence. *Artificial Intelligence*, 33(1):1–64, 1987.

[LRDG90] Jed Lengyel, Mark Reichert, Bruce. R. Donald, and Donald 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.

[LRL+97] Hector Levesque, Raymond Reiter, Yves Lespérance, Fangzhen Lin, and Richard Scherl. Golog: A logic programming language for dynamic domains. *Journal of Logic Programming*, 31, 1997.

[LSV95] John Lewis, Peter Skarek, and Lawrence Varga. A rule-based consultant for accelerator beam scheduling used in the CERNPS complex. In *ICALEPCS'95: International Conference on Accelerator and Large Experimental Physics Control Systems*, Chicago, Illinois USA, October 30-November 3 1995.

[LYLN07] Gang Li, Shiwen Yang, Yanhui Liu, and Zaiping Nie. A variable step length hybrid approach for electromagnetic ray tracing in ionosphere. *Electromagnetics*, 27(6):331–340, 2007.

[Mae90] Patti Maes. Situated agents can have goals. In Patti Maes, editor, *Designing Autonomous Agents*, pages 49–70. MIT Press, 1990.

[Mal97] Hanspeter Mallot. Behavior-oriented approaches to cognition: Theoretical perspectives. In *Theory in Biosciences*, volume 116, pages 196–220, 1997.

[Mat92] Maja Mataric. Integration of representation into goal-driven behavior-based robots. *IEEE Transactions on Robotics and Automation*, 8(3):304–312, 1992.

[Mat05] Romain Matschek. A geometrical optics and uniform theory of diffraction based ray tracing optimisation by a genetic algorithm. *Comptes rendus de physique*, 6(6):595–603, July 2005.

[MBCT98] Soraia Raupp Musse, Christian Babski, Tolga Capin, and Daniel Thalmann. Crowd modelling in collaborative virtual environments. In *VRST '98: Proceedings of the ACM symposium on Virtual reality software and technology*, pages 115–123, New York, NY, USA, 1998. ACM.

[ME01] Philippe Martin and Peter W. Eklundm. WebKB-2: Cooperatively-built knowledge bases on the www. In *WWW Posters*, 2001.

[MH69] John McCarthy and Patrick J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence*, 4:463–502, 1969.

[Min74] Marvin Minsky. *A Framework for Representing Knowledge*. MIT-AI Laboratory, 1974.

[Min03] Guy Mineau. Representing and enforcing interaction protocols in multi-agent systems: an approach based on conceptual graphs. In *Proceedings of the EEE/WIC International Conference on Intelligent Agent Technology*, pages 261 – 267, 13-16 Oct 2003.

[MLW+06] Peter McGuire, Geoffrey Liggins, Peter Wojcik, Abderrezak Benaskeur, and Robert Brennan. The application of holonic control to tactical sensor management. In *Proceedings of the IEEE Workshop on Distributed intelligent Systems: Collective intelligence and Its Applications*, pages 225–230, Washington, DC, 2006. IEEE Computer Society.

[Mol98] Martien Molenaar. *An introduction to the Theory of Spatial Object Modelling for GIS*, chapter A formalism for the vector data model, pages 73–102. Tailor & Francis, 1998.

[MPB03] Jean-Eudes Marvie, Julien Perret, and Kadi Bouatouch. Remote interactive walkthrough of city models. In *Proceedings of the 11th Pacific Conference on Computer Graphics and Applications (PG'03)*, pages 389–393, October 2003.

[MT97] Raupp Musse and Daniel Thalmann. A Model of Human Crowd Behavior. In *CASA'97, Springer Verlag, Wien*, pages 39–51, 1997.

[New90] Allen Newell. *Unified theories of cognition*. Harvard University Press, Cambridge, Massachusetts, 1990.

[Nil82] Nils Nilsson. *Principles of Artificial Intelligence*. Springer-Verlag, Berlin ; Heidelberg ; New York, third edition, 1982.

[NN04] Robert Najlis and Michael North. Repast for GIS. In *Proceedings of Agent 2004: Social Dynamics: Interaction, Reflexivity and Emergence*, University of Chicago and Argonne National Laboratory, IL, USA, 2004.

[Nor02] Donald Norman. *The Design of Everyday Things*. Basic Books, September 2002.

[NT97] Hansrudi Noser and Daniel Thalmann. Sensor based synthetic actors in a tennis game simulation. In *Proceedings of the International Conference on Computer Graphics*, (Cat. No.97TB100104), 1997. Comput. Graphics Lab., Swiss Federal Inst. of Technol., Lausanne, Switzerland.

[OOB00] Atsuyuki Okabe, Michiko Okabe, and Barry Boots. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. John Wiley and Sons Inc., 2000.

[Ope08] Open Source Geospatial Fundation (OSGEO). GDAL-OGR: Geospatial Data Abstraction Library / Simple Features Library Software, August 2008. http://www.gdal.org.

[Par04] Dawn Parker. *GIS, Spatial Analysis and Modelling*, chapter Integration of Geographic Information Systems and Agent-based Models of Land Use: Challenges and Prospects, pages 2–20. Redlands, CA: ESRI Press, 2004.

[PCTC07] L Pun-Cheng, M Tang, and I Cheung. Exact cell decomposition on base map features for optimal path finding. *International Journal of Geographical Information*, 21(2):175–185, 2007.

[PDB06] Sébastien Paris, Stéphane Donikian, and Nicolas Bonvalet. Environmental abstraction and path planning techniques for realistic crowd simulation. *Computer Animation and Virtual Worlds*, 17:325–335, 2006.

[Pea84] Judea Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley, 1984.

[Pet99] Sylvain Petitjean. Algebraic geometry and computer vision : Polynomial systems, real and complex roots. *Journal of Mathematical Imaging and Vision*, 10(3):191–220, May 1999.

[PGT07] Julien Pettré, Helena Grillon, and Daniel Thalmann. Crowds of Moving Objects: Navigation Planning and Simulation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3062–3067, 2007.

[PLS03] Julien Pettré, Jean-Paul Laumond, and Thierry Siméon. A 2-stages locomotion planner for digital actors. In *SCA '03: Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 258–264, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[QT09] Trolltech QT, november 2009. http://qt.nokia.com/products.

[Rau01] Martin Raubal. Ontology and epistemology for agent-based wayfinding simulation. In S. Winter, editor, *Geographical Domain and Geographical Information Systems - EuroConference on Ontology and Epistemology for Spatial Data Standards*, volume 19 of *GeoInfo Series*, pages 85–87, France, September 2001. La Londe-les-Maures.

[Rey87] Craig Reynolds. Flocks, herds, and schools: A distributed behavioral model. In *Proceedings of ACM SIGGRAPH Conference on Computer Graphics*, pages 25–34, Anaheim, California, July 1987.

[RG91] Anand Rao and Michael Georgeff. Modeling rational agents within a BDI-architecture. In James Allen, Richard Fikes, and Erik Sandewall, editors, *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*, pages 473–484. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA, 1991.

[RG95] Anand Rao and Michael Georgeff. BDI-agents: from theory to practice. In *Proceedings of the First International Conference on Multiagent Systems*, San Francisco, 1995.

[Rho97] Bradley Rhodes. PHISH nets: planning heuristically in situated hybrid networks. In *AGENTS '97: Proceedings of the First International Conference on Autonomous Agents*, pages 480–481, New York, NY, USA, 1997. ACM.

[Saj08] Priti Srinivas Sajja. Multi-agent system for knowledge-based access to distributed databases. *Interdisciplinary Journal of Information, Knowledge, and Management*, 3:1–9, 2008.

[SB05] Nathan Sturtevant and Michael Buro. Partial pathfinding using map abstraction and refinement. In *AAAI'05: Proceedings of the 20th National Conference on Artificial Intelligence*, pages 1392–1397. AAAI Press, 2005.

[SCM03] Jonathan Schwartz, Gary Comparetto, and Jim Marshall. The communications resource planning tool. In *Military Communications Conference, 2003. MILCOM 2003. IEEE*, volume 1, pages 227–230, Boston, MA, USA, October 2003.

[Sei82] Alain Seidel. Way-finding in public spaces: The dallas/fort worth, usa airport. In *Proceedings of the 20th International Congress of Applied Psychology*, Edinburgh, Scotland, 1982.

[SGC04] Mankyu Sung, Michael Gleicher, and Stephen Chenney. Scalable behaviors for crowd simulation. *Computer Graphics Forum*, 23(3):519–528, 2004.

[Sha06] Wei Shao. *Animating Autonomous Pedestrians*. PhD thesis, Department of Computer Science, Courant Institute of Mathematical Sciences, New York University, 2006.

[SK97] Hagit Shatkay and Leslie Kaelbling. Learning topological maps with weak local odometric information. In *IJCAI'97: Proceedings of the International Joint Conference on Artificial Intelligence*, pages 920–927, 1997.

[SKG05] Mankyu Sung, Lucas Kovar, and Michael Gleicher. Fast and accurate goal-directed motion synthesis for crowds. In *SCA'05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 291–300, New York, NY, USA, 2005. ACM.

[SL99] Finnegan Southey and James G. Linders. Notio - a Java API for developing CG tools. In *ICCS'99: Proceedings of the International Conference on Conceptual Structures*, pages 262–271, 1999.

[Sow84] J. F. Sowa. *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1984.

[Sow87] John Sowa. *Encyclopedia of Artificial Intelligence*. Wiley, 1987.

[Sow99] John Sowa. *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Course Technology, August 1999.

[SR94] S.Y. Seidal and T.S. Rappaport. Site-specific propagation prediction for wireless in-building personal communication system design. *IEEE Transactions on Vehicular Technology*, 43(4):879–891, 1994.

[ST05] Wei Shao and Demetri Terzopoulos. Environmental modeling for autonomous virtual pedestrians. *Digital Human Modeling for Design and Engineering Symposium*, 2005.

[Ste94] A. Stentz. Optimal and efficient path planning for partially-known environments. In *Proceedings of the International Conference on Robotics and Automation*, volume 1, pages 3310–3317, 1994.

[SW86] John Sowa and Eileen Way. Implementing a semantic interpreter using conceptual graphs. *IBM J. Res. Dev.*, 30(1):57–69, 1986.

[SW06] Elmar Schömer and Nicola Wolpert. An exact and efficient approach for computing a cell in an arrangement of quadrics. *Computational Geometry*, 33(1-2):65 – 97, 2006. Robust Geometric Applications and their Implementations.

[TB96] Sebastian Thrun and Arno Bücken. Integrating grid-based and topological maps for mobile robot navigation. In *AAAI'96: Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 944–951, Portland, Oregon, August 1996. AAAI Press/MIT Press.

[TB05] Paul Torrens and Isaak Benenson. Geographic automata systems. *Journal of Geographic Information Systems*, 19(4):385–412, 2005.

[TC00] Franco Tecchia and Yiorgos Chrysanthou. Real-time rendering of densely populated urban environments. In *Proceedings of the Eurographics Workshop on Rendering Techniques*, pages 83–88. Springer, 2000.

[TD00] Gwenola Thomas and Stephan Donikian. Virtual humans animation in informed urban environments. *Computer Animation 2000*, pages 112–119, 2000.

[TD03] Romain Thomas and Stéphane Donikian. A model of hierarchical cognitive map and human memory designed for reactive and planned navigation. In *Proceedings of the 4th International Space Syntax Symposium*, volume 1, pages 72–100, Londres, 2003.

[TLC02] Franco Tecchia, Céline Loscos, and Yiorgos Chrysanthou. Visualizing crowds in real-time. *Computer Graphics Forum*, 21(4):753–765, 2002.

[TLCC01] Franco Tecchia, Céline Loscos, Ruth Conroy, and Yiorgos Chrysanthou. Agent behaviour simulator (ABS): A platform for urban behaviour development. In *GTEC'01: Proceedings of the ACM/EG Games Technology Conference*, pages 17–21, 2001.

[TPH02] John Thangarajah, Lin Padgham, and James Harland. Representation and reasoning for goals in BDI agents. volume 24, pages 259–265, Los Alamitos, CA, USA, 2002. IEEE Computer Society Press.

[TT94] Xiaoyuan Tu and Demetri Terzopoulos. Perceptual modeling for behavioral animation of fishes. In *Second Pacific Conference on Computer Graphics*, pages 185–200, 1994.

[Tyr94] Toby Tyrrell. An evaluation of Maes's bottom-up mechanism for behavior selection. *Adaptive Behavior*, 2(4):307–348, 1994.

[Var10] Simona Elena Varlan. Knowledge representation in the context of e-business applications. *BRAND. Broad Research in Accounting, Negotiation, and Distribution*, 1(1):1–4, September 2010.

[VDP93] Michiel Van De Panne. Sensor-actuator networks. In *SIGGRAPH '93: Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, pages 335–342, New York, NY, USA, 1993.

[Wan05] Xinhao Wang. Integrating GIS, simulation models, and visualization in traffic impact analysis. *Computers, Environment and Urban Systems*, 29(4):471–496, July 2005.

[War95] William Warren. *Perception of Space and Motion: Handbook of perception and cognition*, chapter Self-motion: Visual perception and visual control, pages 263–325. Academic Press, San Diego, CA, US, 1995.

[WCC+07] Shiuh Wang, Lian Chou, Hirata Chiu, Yan Tseng, Min Hsu, Wen Cheng, Woo Liu, and Thiao Ho. NCTUns 4.0: An integrated simulation platform for vehicular traffic, communication, and network researches. In *Proceedings of the 1st IEEE International Symposium on Wireless Vehicular Communications*, Baltimore, MD, USA, 2007.

[Wei81] Jerry Weisman. Evaluating architectural legibility: Way-finding in the built environment. *Environment and Behavior*, 13(2):189–204, March 1981.

[Wei82] Mark Weissberger. An initial critical summary of models for predicting the attenuation of radio waves by trees. In *Final Report Electromagnetic Compatibility Analysis Center, Annapolis, MD.*, Annapolis, Md, July 1982.

[WH03] Danny Weyns and Tom Holvoet. Model for situated multi-agent systems with regional synchronization. In *CE'03: Proceedings of the 10th International Conference on Concurrent Engineering, Agent and Multi-Agent Systems*, pages 177–188. Springer-Verlag, 2003.

[WHBAJ06] Matthias Weber, Guido Heumer, Heni Ben Amor, and Bernhard Jung. *Advances in Artificial Reality and Tele-Existence*, chapter An Animation System for Imitation of Object Grasping in Virtual Reality, pages 65–76. Elsiver, 2006.

[Wil98] Svein Yngvar Willassen. A method for implementing mobile station location in GSM. Master's thesis, Norwegian University of Techniques and Sciences, 1998.

[WJ95] Michael Wooldridge and Nick Jennings. Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10(2), 1995.

[WK06] Wojciech Walendziuk and Slawomir Kwieckowski. The use of the dynamic space decomposition algorithm of a computational area in heterogeneous

cluster computations. In *Proceedings of the International Symposium on Parallel Computing in Electrical Engineering*, pages 466–469, 2006.

[WL08] Kai Yip Wong and Céline Loscos. *Motion in Games*, chapter Hierarchical Path Planning for Virtual Crowds, pages 43–50. 2008.

[WOO07] Danny Weyns, Andrea Omicini, and James Odell. Environment, first-order abstraction in multiagent systems. *Journal of Autonomous Agents and Multiagent Systems*, 2006:5–30, 2007.

[WSH05] Danny Weyns, Kurt Schelfthout, and Tom Holvoet. Exploiting a virtual environment in a real-world application. Environments for Multi-Agent Systems II, february 2005.

[Yer09] Barbara Yersin. *Real-Time Motion Planning, Navigation, and Behavior for Large Crowds of Virtual Humans*. PhD thesis, École polytechnique fédérale de Lausanne, 2009.

[Zac06] John Zacharias. Exploratory spatial behaviour in real and virtual environments. *Landscape and Urban Planning*, 78(1-2):1–13, 2006.

[ZFP94] Uwe R. Zimmer, Cornelia Fischer, and Ewald Von Puttkamer. Navigation on topologic feature-maps. In *Proceedings of the 3rd International Conference on Fuzzy Logic, Neural Nets and Soft Computing*, pages 131–132, Iizuka, Japon, 1994.

[ZLS+08] Liangchena Zhou, Guoniana Lu, Yehuaa Sheng, Hongboa Xu, and Haixiaa Wang. A 3D GIS spatial data model based on cell complex. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXVII:905–908, 2008.