COMPUTER ANIMATION AND VIRTUAL WORLDS
Comp. Anim. Virtual Worlds 2012; **23**:87–99
Published online 21 March 2012 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cav.1428

SPECIAL ISSUE PAPER

Space-time planning in changing environments : using dynamic objects for accessibility

Thomas Lopez[1]*, Fabrice Lamarche[1] and Tsai-Yen Li[2]

[1] IRISA, Université de Rennes 1, Rennes, France
[2] Computer Science Department, National Chengchi University, Taipei 116, Taiwan

ABSTRACT

Navigation is a critical task for agents populating virtual worlds. In the last years, numerous solutions have been proposed to solve the path planning problem in order to enhance the autonomy of virtual agents. Those solutions mainly focused on static environments, eventually populated with dynamic obstacles. However, dynamic objects are usually more than just obstacles as they can be used by an agent to reach new locations. In this paper, we propose an online path planning algorithm in dynamically changing environments with unknown evolution such as physically based-environments. Our method represents objects in terms of obstacles but also in terms of navigable surfaces. This representation allows our algorithm to find temporal paths through disconnected and moving platforms. We will also show that the proposed method also enables several kinds of adaptations such as avoiding moving obstacles or adapting the agent postures to environmental constraints. Copyright © 2012 John Wiley & Sons, Ltd.

KEYWORDS

path planning; motion planning; dynamic environments; autonomous agents; accessibility

***Correspondence**

Thomas Lopez, MimeTIC, IRISA, Université de Rennes 1, Rennes, France.
E-mail: lopezthom@gmail.com

1. INTRODUCTION

In the last decades, path planning has been widely studied, and numerous solutions for static environments, eventually populated with dynamic obstacles, have been proposed. Our method focuses on a virtual agent navigating in dynamically changing environments where the evolution of the topology is not known *a priori*. This kind of situation can be found in environments by using physical simulation. The physics engine makes the environment highly dynamic, and the motions of objects cannot be known *a priori*, which makes the navigation task complicated. Moreover, we address a new kind of path planning problem by considering dynamic objects not only as obstacles but also as navigable parts of the environment. Those navigable elements can thus be used to access new locations. For instance, a plank can act as a bridge and connect two disconnected regions. Using the agent navigation capabilities, our method builds a dual representation of objects that identifies their impact in terms of accessibility and obstruction. This representation is used to characterize and track over time topological modifications in a global representation of the environment's topology. Our method, using temporal properties, is thus able to identify complex paths through disconnected and moving surfaces. Those paths are detected even if the surfaces are never directly connected all together at the same time. We will show that our solution is able to solve such complex cases in real time by generating collision-free paths and adapting the agent's postures among dynamic obstacles. Moreover, we will present a method to efficiently handle and represent highly dynamic objects, which navigability and obstruction properties are modified at runtime. This property makes our algorithm suitable for interactive applications where an external user, a script or a physics engine may act on the environment at runtime. Such an approach is useful in several application fields, including video games where non-player agents are evolving in dynamic environments where changes are not always known *a priori*.

In the following, we first introduce the related works. We then describe the precomputation steps associated to the agent representation and the dual representation of the dynamic objects. We also present how to update this dual representation in the context of highly dynamic objects. The next section focuses on the use of those representations to plan a path and adapt postures of a virtual agent evolving in a dynamic environment. Finally, we show and analyze some results.

2. RELATED WORK

Path planning has been widely studied in robotics where spatial reasoning provides robots with a crucial functionality : autonomy of navigation [1,2]. Given a character, its navigation capabilities and a description of the environment, the purpose is to plan a collision-free path for the character between two given locations. The general path planning problem is known to be PSPACE-complete [3,4]. Its formulation relies on the exploration of the configuration space (C-space). This C-space is defined as an N-dimensional space for which each of the N axes represents a degree of freedom (DOF) of the character. The C-space is generally divided in C-free, a subspace containing valid configurations, and C-obstacle, a subspace containing obstructed ones. Thus, planning a collision-free path for a character is equivalent to finding a path in C-free that links two configurations. The basic planning problem focuses on finding a valid path in a static environment. Proposed methods mostly fall into two categories: cell decomposition and roadmaps. The cell decomposition methods are either approximate, representing a subset of C-free with cells of predefined shapes [5,6], or exact, representing C-free using trapezoidal decomposition, Delaunay triangulations and variants [7,8]. Probabilistic methods, such as probabilistic roadmaps (PRMs) [9,10] or random trees (RRTs) [11,12], explore C-free by computing a roadmap in which nodes are non-colliding configurations randomly sampled over C-free, and edges are collision-free paths linking two nodes. Most of the methods generally consider navigation in a connected environment. However, few methods focused on static but disconnected environments [10,13,14]. The need of planning paths in dynamically changing environments arises in many application fields. Most of the proposed methods focus on avoiding dynamic obstacles. On the one hand, some methods consider that obstacles movements are known and use this knowledge to guide and speed-up the path planning process [15,16]. On the other hand, fast replanning techniques are used when an obstacle is detected along the planned trajectory [17,18]. Various methods based on PRMs [17] or RRTs [19] have been proposed. This kind of methods validates precomputed edges of the roadmaps during planning to take dynamic changes into account [20]. By taking time into account, other methods propose to define safe intervals where the agent can wait safely during the navigation and use those intervals to handle path planning among dynamic obstacles [21]. Velocity obstacles and extensions [22] propose a reactive approach that continuously updates the agent speed to generate trajectories avoiding collisions with dynamic obstacles. Finally, methods based on rapidly computed generalized Voronoï diagram have also been proposed [23,24].

To our knowledge, the first method handling space-time planning in a dynamic and disconnected environment has been recently proposed by Levine *et al.* [25]. The singularity of both methods is that moving platforms are no longer considered as obstacles but also as navigable regions used during navigation. However, their method is based on the strong hypothesis that the objects movements have known trajectories. This involves that the future evolutions of the environment are always known and thus that accessibilities between objects can be precomputed. On the basis of this hypothesis, they compute their path using a trial-and-error solver, trying different motion controllers until a correct motion is found.

Our method handles space-time planning in a dynamically changing environment with no *a priori* knowledge on the topology evolution. By observing this evolution at runtime, our algorithm characterizes and track topological relations over time. On the basis of this information, our method is able to compute in real time paths among moving and disconnected surfaces while avoiding dynamic obstacles and adapting the character's postures to environmental constraints.

3. AGENTS AND OBJECTS REPRESENTATIONS

Our path planning problem considers a non-flying agent evolving in a dynamic environment composed of non-deformable objects. No assumption is made on the spatio-temporal evolution of the environment that is only assumed to be observable. The dimension of the associated C-Space includes the agent's DOF and the moving objects DOF. Considering all of those DOF creates a large search space. Thus, in order to improve the path planning problem performance, the complexity of this problem must be reduced. To propose a solution compatible with interactive applications, the agent representation is simplified using bounding volumes, which reduce the number of DOF of the problem. Using the bounding volumes, we extract a dual representation of the objects that precisely defines the impact of an object in terms of accessibility and obstructions. Finally, in the context of dynamic environments, this representation will have to be dynamically updated in order to keep a consistent representation of the topology.

3.1. The Agent and its Navigation Capabilities

Considering that the motion planning problem is PSPACE-complete, bounding volumes are often used to represent an agent. Indeed, it decouples the animation and the path planning process that simplifies and speeds-up the path planning process as it reduces the number of DOFs [13,15,22]. We consider an agent with navigation capabilities mainly constrained on the floor. In order to speed-up path planning and decouple it from the animation process, we thus represent our agent by using bounding cylinder. For each navigation capability M_i, we define a cylinder C_i, centered on the agent's root, which bounds its geometry when playing the motion i. Given the cylinder C_i bounding a navigation capability of our agent, a configuration in this C-space

DOI: 10.1002/cav

represents the 3D-position of the agent's root located at the bottom center of the cylinder. When the agent's animation changes from a motion i to a motion j, we compute a transition cylinder $T_{i,j}$ bounding the postures of the agent during the transition. In order to handle an agent navigating on slopes [26], we associate to each navigation capability M_i and to each transition an interval of navigable slopes in which the agent is able to use this capability. Jump motions are particular as an agent could get hurt when it jumps down or might not be able to reach a high location. To model this, the jump motion is labeled with a maximum vertical impulse speed, a maximum horizontal impulse speed, and a maximum vertical landing speed. Finally, in order to represent an agent subject to the gravity, we assume that this agent follows a ballistic trajectory when jumping.

3.2. Augmenting geometry with Interaction Volumes

The considered environments, composed of geometric elements, define the workspace where the agent has to navigate. A configuration space (C-space), representing all the possible configurations between the agent and the environment, is associated to this workspace. The whole environment is thus represented in this C-space where the entire path planning process is performed. We thus introduce the definition of *Interaction Volumes*, which characterize subspaces of this C-space.

3.2.1. Definition of the Interaction Volumes.

From the agent's point of view, a geometric object impacts on the local topology in three different ways. It can obstruct a region, present navigable surfaces or create an access to other areas. Obstruction, navigability, and accessibility properties rely on the agent's navigation capabilities. Using those capabilities, we extract a dual representation of objects, called the *Interaction Volumes*, which characterizes feasible, colliding, and reachable configurations of the C-space. We then associate to each surface a precomputed roadmap. This roadmap allows to capture the local topology of all of the navigable surfaces in a single structure ready for path planning requests.

In the following, we assume that the (X, Y) axes represent the horizontal plane, and that the Z-axis is the height axis of the environment. Considering an object O and a navigation capability M_k associated to the motion k bounded by the cylinder C_k, we define three types of *Interaction Volumes* (Figure 1): the *Forbidden Volume*, the *Navigable Surface*, and the *Accessibility Volume*.

Forbidden Volumes, denoted $V_f(O, M_k)$, represent the set of configurations where the agent collides with the object O. This volume is obtained by extruding the object's shape along the Z-axis using the height of the cylinder C_k associated to M_k. As shown in Figure 1, this

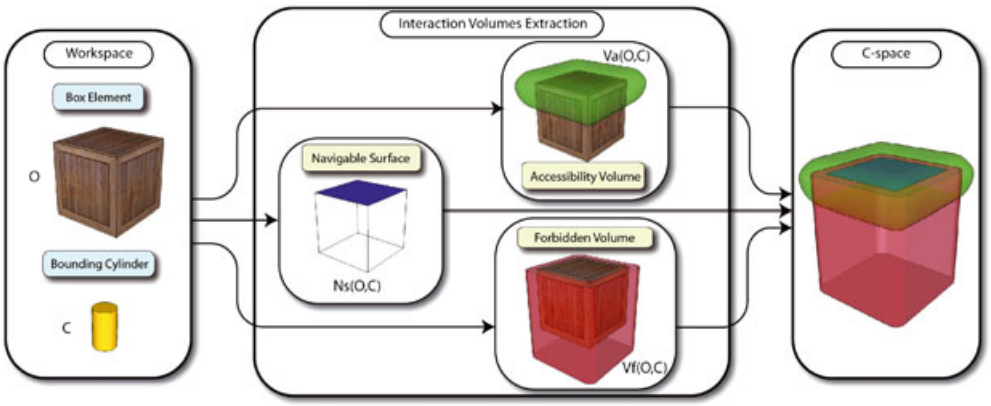

Figure 1. *Interaction Volumes*: given an original geometry *O* and a cylinder *C* bounding a considered motion capability, we compute three *Interaction Volumes*: the *Navigable Surface Ns*(*O*, *C*), the Accessibility Volume *Va*(*O*, *C*), and the *Forbidden Volume Vf*(*O*, *C*). On the right, the dual representation of *O* that is inserted in the C-space.

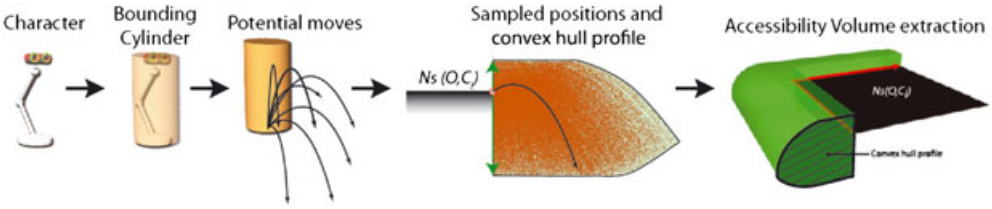

Figure 2. *Accessibility Volume* definition : regarding a character and its jumping capability, we extract a profile representing its potential jumps from a start configuration. The last scheme presents the profile extraction along edges of *Ns*(*O*, *C*).

shape is then extended along the (X, Y)-axes by using the cylinder's radius.

Navigable Surfaces, denoted $Ns(O, M_k)$, represent the regions where the agent can stand. In order to navigate on uneven floor, we tag each navigation capability M_k with an interval of navigable slopes. We use this interval to determine whether or not an agent is able to stand on the considered surface. $Ns(O, M_k)$ is computed by grouping all triangles of the object's mesh with admissible slopes minus configurations lying in $V_f(O, M_k)$. A simple *Navigable Surface* is shown in Figure 1.

Accessibility Volumes, denoted $V_a(O, M_k)$, contain all configurations reachable from $Ns(O, M_k)$ when using the jump capability M_k. First, the maximum reachable height is used to extrude $Ns(O, M_k)$ along the height axis. Second, given the jump motion characteristics, we compute an Accessibility Profile (Figure 2) by randomly sampling the set of admissible jumping trajectories and computing the convex hull shape of the sampled trajectories. This profile is then extruded along the borders of $Ns(O, M_k)$ to finalize $V_a(O, M_k)$.

3.2.2. Local Roadmap Generation.

Because the global structure of a *Navigable Surface* $Ns(O, M_k)$ does not change, a local roadmap is precomputed on each surface. Different methods have been proposed in the literature to build a roadmap. We chose the well-known PRMs method to create local roadmaps [9]. We thus randomly sample configurations in $\bigcup_k Ns(O, M_k)$ and annotate each sampled configuration c with the set of motion capabilities that are valid, that is, $\{k,\ c \in Ns(O, M_k)\}$. Each sample is then connected to its k-nearest neighbors iff the configurations share at least one common motion capability M_k and that a collision free linear path lying in $Ns(O, M_k)$ exists. An example of roadmap's construction is shown Figure 3. In this example, two navigation capabilities are used, and we can see how the roadmap is able to capture all of the navigation opportunities of the agent.

3.3. Dynamic Objects Representation

By considering dynamic environments, we create situations in which objects configurations evolve over time. This evolution will sometime be unpredictable for the agent. For example, the configuration of an environment using a physics engine changes depending on the actions of a user. In such an environment, we can easily imagine an object falling while rotating on itself, or an element that the user translates on the floor. Augmenting dynamic objects

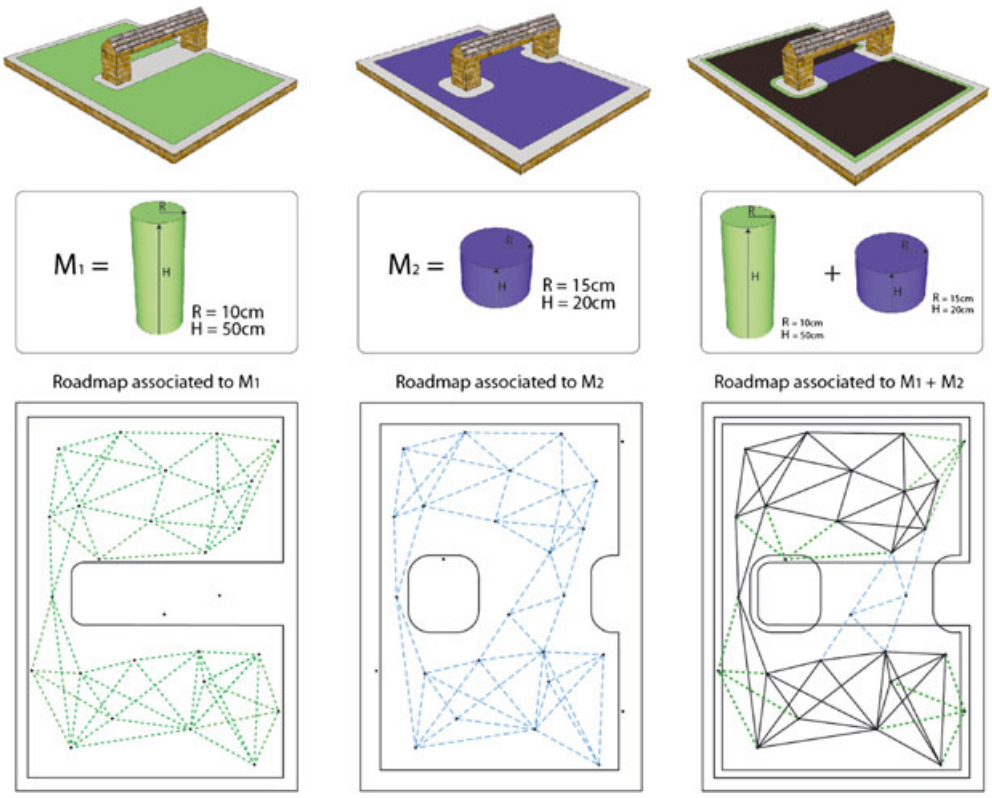

Figure 3. Roadmap definition. Two navigation are considered: M_1 and M_2. A Navigable Surface is associated each of them and configuration of the roadmap are sampled in the union of those Navigable Surfaces. Edges are then build when a collision-free path lying in at least one Navigable Surface exists.

DOI: 10.1002/cav

with Interaction Volumes allows identifying their topological impact. However, in order to handle highly dynamic elements, those Interaction Volumes have to be updated from time to time during the simulation.

A dynamic object possesses six DOF in the virtual environment. Those six DOF consist in three translations (along the axes X, Y, and Z) and three rotations (ψ around X, ϕ around Y, and θ around Z). Three different cases arise.

(1) Translations modify the position of the object but not its orientation. In this case, the same translations are applied directly on the Interaction Volumes in order to update their positions in the configuration space but their definition will remain the same. Then the *Interaction Volumes* are persistent to translation.
(2) A θ rotation modifies the object as well, but would not change the inclination of the *Navigable Surfaces*. Thus, the Interaction Volumes definition does not change, and the θ rotation is applied to those volumes in the C-Space.
(3) Finally, ψ and ϕ rotations change the definition of the *Interaction Volumes* as they modify the angle between the object and the vertical axis of the environment. This modification impacts on the *Navigable Surface* definition and, consequently, on the *Accessibility Volume*. As the *Forbidden Volumes* are defined by an extrusion of the object's shape along the vertical axis of the environment, this definition change too.

When a ψ or ϕ rotation modifies an object configuration, new *Interaction Volumes* are computed to preserve a valid representation of the topological impact of this object. In order to simplify these updates and the creation of new *Interaction Volumes*, we make the observation that an object rotating around the X or Y axis can hardly be navigable during its rotation and will become navigable again after its stabilization. Consequently, a rotating object will be considered only as an obstacle during its rotation as the agent would not be able to use it for its navigation task. Thus, as long as this object is rotating, we simply represent it as a *Forbidden Volume* to denote this property. When the object stabilizes on the ground, new *Interaction Volumes* corresponding to its current situation are defined and reassigned to this object.

Moreover, in order to reduce the number of new definition of *Interaction Volumes*, we associate to the global simulation a rotation parameter Δ. Thus, if the object rotated of an angle inferior to Δ since the last *Interaction Volumes* update, those volumes are kept. On the opposite, if this angle is superior to Δ, the volumes are discarded and new volumes are created. This parameter lightens the cost due to the update of those volumes but also scatters those creations over time as the dynamic objects might certainly not rotate with the same angular speed.

The update of those Forbidden Volumes is shown Figure 4. We observe the new Interaction Volumes associated to the object during its rotation. While the object is rotating (center), it is represented by a simple *Forbidden Volume* and the new *Interaction Volumes* are computed after stabilization.

3.4. Properties of the Representation

The *Interaction Volumes* represent the impact of objects on their local environment's topology. Identifying a topological relation between two objects in the workspace is thus equivalent to detecting an intersection between their respective *Interaction Volumes* in the C-space. This important property enables an accurate characterization of topological relations and is a key point in our method. Given two objects (O_i, O_j) and a motion capability m, accessibility and obstruction relations are defined as follows:

- **Accessibility**: $A(O_i, O_j, C_m)$, holds when $V_a(O_i, C_m) \cap Ns(O_j, C_m) \neq \emptyset$ and characterizes an access from O_i to O_j with the motion capability m. This relation is not a bijection as the agent may have more difficulties to climb on objects than to go down (Figure 2).
- **Obstruction**: $O(O_i, O_j, C_m)$, holds when $V_f(O_i, C_m) \cap V_a(O_j, C_m) \neq \emptyset$, that is, O_i obstructs some navigable parts of O_j for the given navigation capability m.

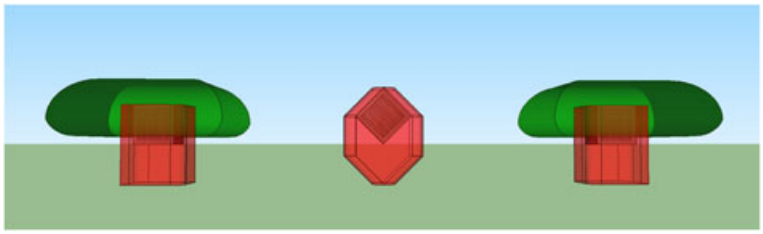

Figure 4. Rotation of a simple box around the X or Y axis. The Interaction Volumes are updated during the movement. The object is represented with a single Forbidden Volume during its motion, and the three Interaction Volumes are computed again when it becomes stable.

The identification of those relations coupled with local roadmaps allows us to locate the topological impact of the detected relations at runtime. An accessibility relation results in a connection between two distinct roadmaps, through a jumping capability, whereas an obstruction invalidates some parts of the roadmap. Obstruction relations have an impact on obstacle avoidance but also on posture adaptation as an obstruction might, for instance, force the agent to adopt a special capability to navigate along a path. Consider, for example, an agent endowed with a walk (w_1) and a crouched walk (w_2) capabilities and navigating on an object O. Now, suppose that a beam B appears in front of the agent. In this situation, $Ns(O, w_1) \cap V_f(B, w_1) \neq \emptyset$ and $Ns(O, w_2) \cap V_f(B, w_2) = \emptyset$. Therefore, an obstruction relation is detected between O and B for the navigation capability w_1 but not for w_2. Consequently, the agent is forced to adapt its posture and to use the crouched walking.

Finally, by reducing our problem to a three dimensional C-Space, several requests relating to path planning can be efficiently handled with classical algorithms proposed in the computer graphics community. The identification of topological relations is thus reduced to a collision detection between *Interaction Volumes* and the path validation to a simple ray casting between the local path and the relevant *Forbidden Volumes*. Those properties are intensively used in our algorithm.

4. FINDING A PATH IN A DYNAMIC ENVIRONMENT

In dynamic environments, the topology evolves and moving objects act as obstacles, bridges or elevators for instance. Topological relations need to be tracked in order to take them into account during navigation. We now present how the *Interaction Volumes* are used to track topology modifications. Moreover, taking time into account allows the agent to accurately avoid moving obstacles and detect moving platforms that link disconnected surfaces. This path planning problem is solved by a two-level path planner. The first level computes a path between *Navigable Surfaces* at the topological level, whereas the second level plans a local path on each *Navigable Surface*.

4.1. Representing the Global Topology

In order to identify the global topology, we introduce the *Topological Graph*. This directed graph aims at building a global representation of the environment's topology by representing each object as a node and each topological relation as a link between the two involved objects (Figure 5). This graph allows to represent and characterize all of the topological relations existing in the environment at a given time and can be viewed as a snapshot of the topology. As described previously, detected collisions between *Interaction Volumes* allow us to identify topological relations existing at a given time. Thanks to the 3-dimensionality of the *Interaction Volumes*, those collisions are detected using a tuned collision detection (CD) algorithm [27]. This algorithm reports interaction between relevant *Accessibility Volumes*, *Navigable Surfaces*, and *Forbidden Volumes*. Every time a relation is detected by the CD, the corresponding edge is created in the *Topological Graph*, and this edge is labeled with the nature of the detected relation: *accessibility* or *obstruction*.

Nevertheless, this graph is a coarse representation of the global topology as it only identifies relations between pairs of objects. Thus, the impact of a relation on another, such as the impact of an obstruction relation on an accessibility relation is not represented, although it can modify it. Regarding the definition of accessibility, an object O_j is reachable from O_i with a jump capability M_k iff $V_a(O_i, M_k) \cap Ns(O_j, M_k) \neq \emptyset$. To refine this relation and avoid potential collisions with *Forbidden Volumes*, we randomly sample a set of jumps from O_i to O_j as follows:

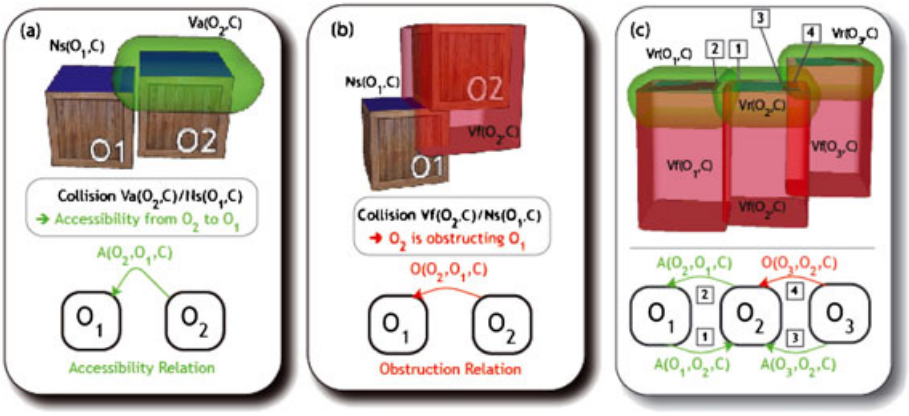

Figure 5. *Topological Graph* construction. Characterization of an accessibility (a), and of an obstruction (b). Example of a more complex situation (c) with four detected relations.

DOI: 10.1002/cav

(1) a target configuration c_t is selected from the local roadmap (PRM) associated to $N_s(O_j, M_k)$;
(2) the nearest source configuration c_s belonging to the local roadmap associated to $N_s(O_i, M_k)$ is then selected (Figure 6(a)); and
(3) the second order polynomial corresponding to the unique ballistic trajectory passing through those configurations is then defined by

$$P(X) = -\frac{1}{2}gX^2 + v_z^0 X + z_s$$

where X represents the abscissa of the agent on the trajectory, whereas $P(X)$ corresponds to its altitude. This trajectory remains by definition in the vertical plane defined by c_s and c_t. In the equation, g represents the gravity constant, v_z^0 is the vertical impulse speed associated to this trajectory, and z_s corresponds to the height of the start configuration c_s.

The obtained jump is then validated iff the impulse and landing speeds satisfy the constraints associated to the jumping capability M_k and the trajectory does not collide with a *Forbidden Volume* (Figure 6(b)). Each validated jump is then stored in the corresponding accessibility edge of the *Topological Graph*.

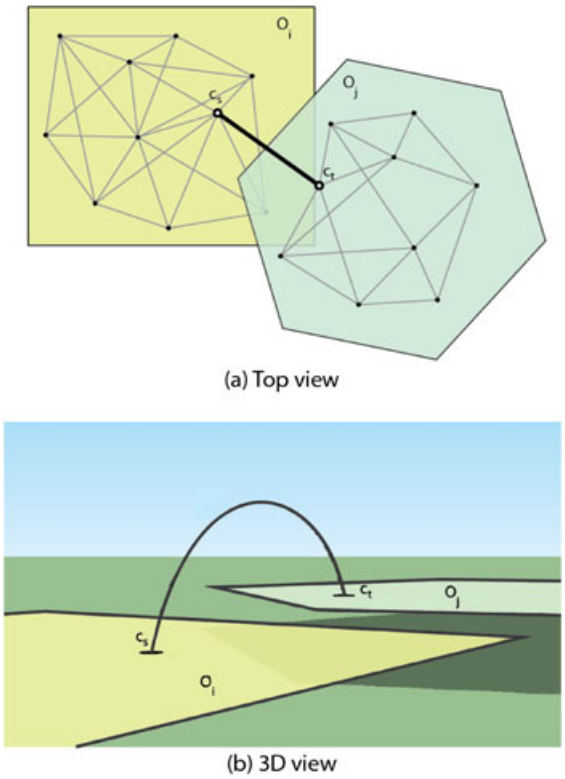

Figure 6. Reachability from O_i to O_j (top view). For each sampled trajectory, a source configuration c_s and a target configuration c_t are selected in the corresponding roadmaps. A second order polynomial, representing the jump capability, is then computed to link those two configurations.

4.2. Tracking a Dynamic Topology

The *Topological Graph* builds a static representation of the environment topology. However, considering the temporal aspect, topological relations between dynamic elements are going to evolve during simulation. By taking time into account, we can thus identify temporal properties about accessibilities and obstructions between dynamic elements. All of those temporal indications are useful to find paths between dynamic elements and using their movements to travel from place to place. The *Topological Graph* is then augmented with temporal information. This allows the agent to build, with no knowledge *a priori*, its own representation of the dynamic environment by observing the evolution of topological relations over time. Thus, edges are labeled with the number of times the relation was valid, the mean validity and non-validity times of the relation and the mean relative speed of the objects. The mean validity time of the relation and the mean relative speed of the objects give an estimate of the relation's stability. The sum of mean validity and non-validity times gives an estimate of the periodicity of the relation and of the waiting time. Finally, the number of times the relation was valid allows the *Topological Graph* to automatically identify and characterize periodic and punctual relations between objects.

The *Topological Graph* thus contains statistical information about validity and stability of observed topological relations. This is crucial as we use this information to characterize relations over time. Thanks to the coupling with the CD algorithm, the *Topological Graph* is automatically updated creating an anytime representation of the topology. Moreover, the temporal information associated to relations enables to automatically represent periodical relations in space and time (Figure 7), but also to better characterize relations in order to optimize the path planning phase. When considering highly dynamic environments, such as physical worlds, we have seen in the previous section that *Interaction Volumes* associated to an object might be deleted and recomputed. However, the *Topological Graph* represents relations between objects regardless of their associated volumes. Thus, a new definition of those volumes does not change the representation of the object in the graph. Nevertheless, old topological relations between dynamic objects might become invalid and never be valid again. In order to prune those useless relations, a time parameter can be used to delete those relation after a given invalidity period.

As mentioned before, a sampling process is used to validate or invalidate static and periodic accessibility relations. Within dynamic environments, this process can be costly as relations evolve over time. To limit the impact of this sampling on performance, a sampling budget is used at each

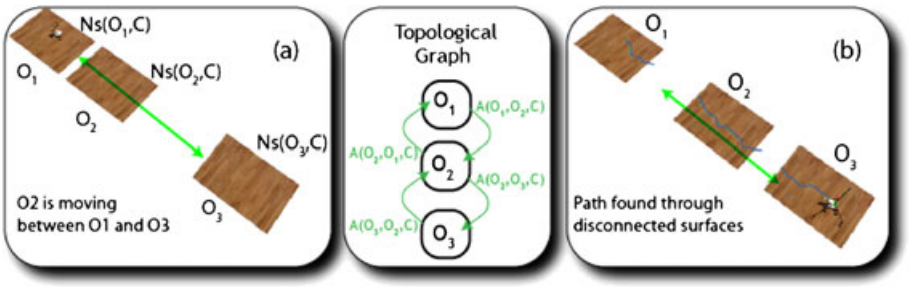

Figure 7. Navigation through disconnected surfaces using the *Topological Graph*.

time step of the simulation. This budget is then distributed among the currently identified accessibility relations.

4.3. A Two-Level Path Planner

In order to find a path in the environment, we designed a two-level path planner. Our planner first selects *Navigable Surfaces* that must be crossed by the agent, using the *Topological Graph* and the temporal information. Then local trajectories are computed on each surface using the associated local roadmaps. Those local trajectories handle posture adaptation and obstacle avoidance.

In order to identify *Navigable Surfaces* on the way, we first filter the *Topological Graph*. This filtering process discard invalidated accessibility relations for which a feasible jump has not been identified. For safety reasons, we also invalidate accessibility relations for which either the mean relative speed of the objects exceeds a given threshold or the mean validity time is lower than a time threshold. This time threshold is important as it makes a difference between an agent that takes risks (low threshold) and an agent that prefers safer paths (high threshold). To compute the global path, a Dijkstra algorithm is run on this filtered view of the *Topological Graph*. Costs associated to the accessibility relations are set to their periodicity (sum of the mean validity and non-validity time) for dynamic relations and to an ϵ-value for stable relations (relations that are always valid). This cost function tends to favor paths through stable links and minimizing the waiting time. The global path planner finally provides a sequence of *Navigable Surfaces* leading the agent toward the global destination.

Once a global path is computed, the local path planner has to generate local trajectories on each roadmap associated to the identified *Navigable Surface* while handling obstacle avoidance and posture adaptation. This local path planning process is run every time the agent reaches a new *Navigable Surface*. To compute this trajectory in the local PRM, we use a multi-target A* algorithm that starts from the current configuration of the agent and finds a path to the nearest target configuration. The target can be either the global target or the source of a jump to access the next *Navigable Surface*. An edge of the roadmap is valid if at least one navigation capability M_k associated to the edge does not collide with local *Forbidden Volumes* and if M_k is compatible with the navigation capability used to reach this edge. Edges validity is checked during the local planning in the space-time domain. In this domain, we anticipate objects positions using a linear extrapolation of their current movements over time [17]. As the evolution is not known *a priori*, we limit the impact of the extrapolation error by setting a maximum extrapolation time. If the time needed to reach the currently explored configuration is greater than this limit, dynamic obstacles are not considered as the agent is not able to anticipate collisions that far in time. This approximation creates an error that will be handled later during navigation. Once the path is computed, the agent follows it. If a new potential collision is detected during navigation, a local replanning is executed to handle errors due to mis-predictions of the obstacles trajectories. When a local target is reached, the agent waits to access the next surface. Using the jump properties, a jump decision is taken by extrapolating the position of the targeted surface and nearby obstacles. If the landing site lies on the targeted surface and the trajectory does not collide with obstacles, the agent jumps. The local planning is repeated on each identified surface until the final target is reached.

Based on the *Topological Graph* and on the analysis of temporal information, our two-level planner solves complex planning problems. Consider, for instance, an environment composed of different moving and disconnected platforms. Those platforms may have periodic movements and give accessibility from time to time to other surfaces in the environment. However, if the temporal aspect is not taken into account, the path planning process is run on a static representation of the topology. This static representation corresponds to a snapshot of the topology at the time the path planning request was sent. The addition of the temporal statistics allows, with no addition of *a priori* knowledge, to automatically detect periodical relations between dynamic elements but also to characterize them. Thus, it makes possible to easily identify a sequence of moving platforms disconnected in space but connected to each other from time to time and that can be crossed to reach a given goal. The search space is also reduced for the local planner that only plans local trajectories and adapts postures on relevant *Navigable Surfaces*.

Comp. Anim. Virtual Worlds 2012; **23**:87–99 © 2012 John Wiley & Sons, Ltd.
DOI: 10.1002/cav

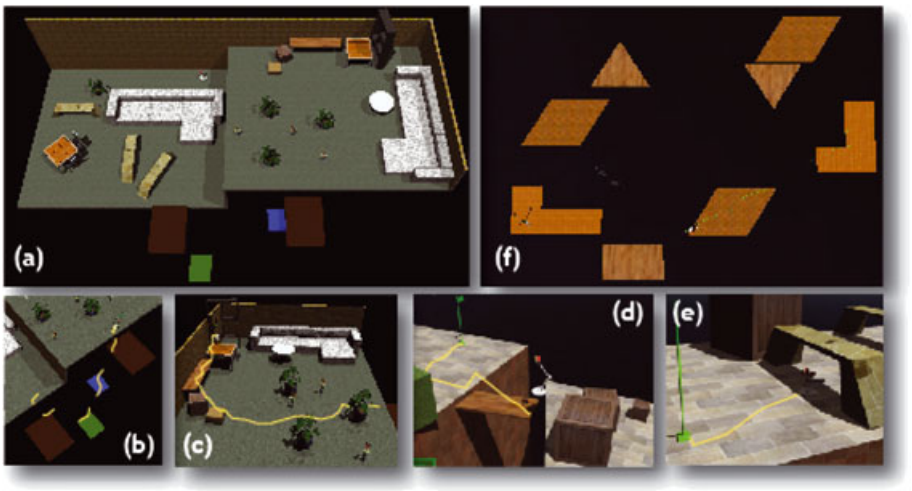

Figure 8. We present our environments: the living room (a) and the disconnected environment (f). Some results show such as: navigation between disconnected and moving surfaces (b), the dynamic obstacle avoidance (c), posture adaptation regarding the motion capabilities (d), and the environmental constraints (e).

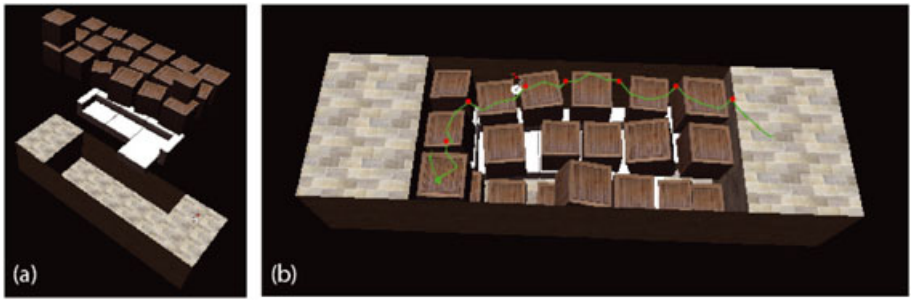

Figure 9. Physics 1 : This is a test environment using a physics simulation. The elements start in configuration (a), the user can then add new elements on the fly. The agent finally has to navigate in this environment and find its path among the navigable surfaces of the physical elements (b).

Figure 10. Physics 2 : Elements are added on the fly by the user. The agent has to find its path among those elements and to use them as soon as they are added to reach new places. A large number of elements is used to test the performance of our method.

Table I. Benchmarks.

Environment name	Collision detection (ms)	Topological graph update (ms)	Average path planning time (ms)	Obstruction tests (% path planning)
Disconnected Env	0.48	0.06	7.8	–
Living-room	14.12	6.46	93.98	67.77
Physics 1	10.01	0.71	25	37.5
Physics 2	14.15	1.25	311	94.6

5. RESULTS

In order to evaluate our method, we designed several testing environments. Those environments aim to highlight different aspects of our works. All of them are composed of dynamic elements. In some environments, dynamic elements have scripted or random trajectories. The last environments are using a physics engine in order to increase the dynamicity of the scene and the agent can add on the fly new elements to this environment.

In our test cases, the agent uses three navigation capabilities: sliding on the ground while (1) standing or (2) crouching, and (3) jumping. The jumping capability allows the agent to reach disconnected navigable surfaces of the workspace. The heights of the bounding cylinders are set to 50 cm for capabilities (1) and (3), 15 cm for capability (2). The radius of those cylinders is set to 20 cm for capability (1) and (3), 30 cm for capability (2). The maximum speed for motions (1) and (3) has been set to 2 $m.s^{-1}$ and the one for the motion (2) has been set to 1 $m.s^{-1}$. When using the jumping capability, our agent is able to jump with maximum vertical and horizontal impulse speeds of 2 $m.s^{-1}$, and we limit the landing speed to 3 $m.s^{-1}$. For the three navigation capabilities, the maximum navigable slope has been set to 30°. We evaluated our method by using different dynamic environments.

Disconnected environment. This environment is composed of disconnected and moving platforms (Figure 8(f)). By jumping from platform to platform when accessibilities are identified, the agent is able to reach every parts of the environment. This example shows how temporal information is used to detect paths in space and time even though the *Navigation Surfaces* are never directly connected. There is no obstacle in the environment.

Living room. This environment, presented in Figure 8(a), demonstrates the various properties of our method. It is composed of numerous complex objects: tables, chairs, sofas, shelves, plants... Those objects all act as obstacles or navigation surfaces and some can constraint the agent's postures. The environment is highly constrained leaving only a few room for navigation. Moreover, *flying books* are used as elevators that connect the two floors together. This environment focuses on connections between distinct surfaces, path obstructions, replanning and posture adaptation during navigation.

Physics 1. This environment, shown in Figure 9, uses a physics engine during the simulation. A couch and some boxes are falling in a hole, and the agent has to find its way between those dynamic elements to reach its destination.

Physics 2. In this environment, using a physics engine, the user adds elements on the fly, and the agent has to find its way between those elements. The navigation task becomes harder and harder as the number of elements in the environment increase as shown in Figure 10. A lot of obstructions are detected during navigation. Moreover, by throwing elements in the environment, the user modifies the configuration of present elements. This test environment has been created in order to test the robustness of our algorithm.

Benchmarks have been realized on an Intel(R) Core(TM)2 Extreme, CPU X7900, 2.80 GHz. We used Bullets Physics CD library to identify *Interaction Volumes* collisions. Our implementation is currently monothreaded. Our benchmarks results are presented in Table I. This table summarizes average times of: the CD between *Interaction Volumes*, the *Topological Graph* update, the connections computation between *Navigable Surfaces*, and the path planning process. The last column presents the percentage of time spent in testing the validity of roadmap edges during the path planning step. Results presented in Table I show that our algorithm performs topology detection and processes path planning requests at interactive frame rates in our testing environments even in the environments using a physics engine for simulation. A video presenting our results is also available online[†].

Our algorithm continuously tracks the collisions between *Interaction Volumes* to identify the evolution of the topology. This process performance is directly correlated with the CD library that is used. The time spent in the graph update is negligible but Table I shows that the use of numerous objects with complex geometries (such as in the *living room*) decreases algorithm performance. In order to reduce the time spent in the collision detection, simplified versions of original meshes (called collision meshes) are often used in real time physics simulations. We could extend those methods to simplify the shapes of the *Interaction Volumes*. In the *Physics 2* environments, path planning is really costly and most of the time (95%) is spent in the collision detection. The use of another collision detection library or of collision detection algorithms parallelized on CPU/GPU may help to improve those performances [28,29]. Second, we defined a dynamic path planner that is

[†]http://people.irisa.fr/Thomas.Lopez/demoCAVW.html

DOI: 10.1002/cav

able to provide temporal trajectories through disconnected surfaces while avoiding predicted collisions. However, the local path planner is the most time consuming process as it needs to test the validity of trajectories using the future obstacle locations. The validity tests use the majority the computation time. Whenever an unexpected obstacle appears on the agent's trajectory, a new local path planning request is emitted. To increase performance and avoid redundant computation, we could try to reuse previously computed paths by using a planning algorithm such as a D*-like, which could be adapted to our problem.

6. CONCLUSION AND FUTURE WORK

In this paper, we presented our approach to online path planning in dynamic environments with unknown evolution. The originality of our approach is that dynamic objects are not only obstacles, but can also be used to navigate and reach previously unreachable locations. The characterization offered by *Interaction Volumes* makes possible to track the evolution of the environment's topology. Our method can be viewed as an observation-based approach in which the dynamic topology representation is built by observing the evolution of the environment's topology over time and by deducing temporal properties. The identification of those properties allows the agent to use the inner motion of dynamic objects during its navigation in order to reach new locations. Our method is thus able to identify paths through surfaces that are connected together over time, but which are not connected all together at a given time. Moreover, the performances of our algorithm show that the method is suitable for interactive applications with highly dynamic objects in complex virtual worlds. Finally, our solution allows to handle any kind of environments with no addition of *a priori* knowledge on the evolution of the world. Using the navigation capabilities of the agent, the corresponding representation of the environment is automatically computed and updated over time by using observed evolutions.

The collision detection algorithm, used for topology tracking and local path planning, is a major bottleneck. However, some recent work focusing either on collision detection parallelization on CPU/GPU [28,29] or on parallelized planning algorithms [30,31] are promising for scaling our algorithm to very complex environments. Another aspect is that the character may sometimes miss the targeted surface and fall down due to an extrapolation error, if the targeted object has chaotic movements for instance. This can be viewed as a limitation of our technique or as something realistic, because the same case can arise in a real situation.

Future work will focus on scalability studies of the method. We are interested in path planning of different characters with individual motion capabilities in the same environment at the same time. It would also be interesting to combine our approach with the one proposed by Levine *et al.* [25]. The combination of their motion controllers, producing paths through disconnected and moving navigable surfaces, with our representation, handling dynamic environments with no *a priori* knowledge could lead to an efficient method. Finally, we intend to analyze the challenging problem of endowing a virtual character with the ability to move objects so as to access a previously unreachable location (by creating stacks of objects for instance). The key idea is to enhance existing solutions to Navigation Among Movable Obstacles problem [32] with the capabilities of our method in terms of topology characterization and path planning.

REFERENCES

1. Latombe JC. *Robot Motion Planning*. Kluwer Academic Publishers, Norwell, MA, USA, 1991.
2. LaValle SM. *Planning Algorithms*. Cambridge University Press, Cambridge, United Kingdom, 2006.
3. Reif HJ. Complexity of the mover's problem and generalizations., In *IEEE Symposium on Fondations of Computer Sciences*, Mayaguez, Puerto Rico, 1979; 421–427.
4. Canny J. Some algebraic and geometric computations in pspace, In *Annual ACM Symposium on Theory of Computing*, Chicago, Illinois, United States, 1988; 460–469.
5. Kuffner JJ. Goal-directed navigation for animated characters using real-time path planning and control. *Lecture Notes in Computer Science* 1998; **1537**: 171–186.
6. Shiller Z, Yamane K, Nakamura Y. Planning motion patterns of human figures using a multi-layered grid and the dynamics filter, In *Proceedings of the 2001 IEEE International Conference on Robotics and Automation, ICRA 2001*, Seoul, Korea, 2001; 1–8.
7. Kallmann M, Bieri H, Thalmann D. Fully dynamic constrained Delaunay triangulations. *Geometric Modelling for Scientific Visualization* 2003; **3**: 1–18.
8. Lamarche F. Topoplan: a topological path planner for real time human navigation under floor and ceiling constraints. *Computer Graphics Forum*; **28**(2): 649–658.
9. Kavraki LE, Svestka P, Latombe J-C, Overmars MH. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation* August 1996; **12**(4): 566–580.
10. Choi MG, Lee J, Shin SY. Planning biped locomotion using motion capture data and probabilistic roadmaps. *ACM Transactions on Graphics* 2003; **22**(2): 182–203.
11. LaValle SM. Rapidly-exploring random trees: a new tool for path planning, Computer Science departement, Iowa State University, 1998: 995–1001.

12. Kuffner JJ, LaValle SM. Rrt-connect: An efficient approach to single-query path planning, In *Proceedings of the 2000 IEEE International Conference on Robotics and Automation, ICRA 2000,* vol. 2, San Francisco, CA, USA, 2000; 995–1001.
13. Li T-Y, Huang P-Z. Planning humanoid motions with striding ability in a virtual environment, In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation, ICRA 2004,* vol. 4, Barcelona, Spain, 2004; 3195–3200.
14. Safonova A, Hodgins JK. Construction and optimal search of interpolated motions graphs. *ACM Transactions on Graphics* 2007; **26**(3): 106.1–106.10.
15. Lau M, Kuffner JJ. Behavior planning for character animation, In *Proc. of Symposium on Computer Animation*, Los Angeles, California, 2005; 271–280.
16. Gayle R, Sud A, Lin MC, Manocha D. Reactive deformation roadmaps: motion planning of multiple robots in dynamic environments, In *IEEE Intelligent Robots and Systems, IROS*, San Diego, California, USA, 2007; 3777–3783.
17. Van den Berg J, Ferguson D, Kuffner JJ. Anytime path planning and replanning in dynamic environments, In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation, ICRA*, Orlando, Florida, USA, 2006; 2366–2371.
18. Zucker M, Kuffner J, Branicky M. Multipartite rrts for rapid replanning in dynamic environments, In *Proceedings of the 2007 IEEE International Conference on Robotics and Automation, ICRA*, Roma, Italy, 2007; 1603–1609.
19. Jaillet L, Simeon T. A PRM-based motion planner for dynamically changing environments, In *IEEE Intelligent Robots and Systems, IROS,* vol. 2, Sendai, Japan, 2004; 1606–1611.
20. Kallmann M, Matarić M. Motion planning using dynamic roadmaps, In *Proc. of the International Conference on Robotics and Automation*, Barcelona, Spain, 2004; 4399–4404.
21. Phillips M, Likhachev M. Sipp: Safe interval path planning for dynamic environments, In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, Shanghai, China, 2011; 5628–5635. IEEE.
22. Van den Berg J, Lin M, Manocha D. Reciprocal velocity obstacles for real-time multi-agent navigation, In *Proceedings of the 2008 IEEE International Conference on Robotics and Automation, ICRA*, Pasadena, California, USA, 2008; 1928–1935.
23. Hoff III KE, Keyser J, Lin M, Manocha D, Culver T. Fast computation of generalized voronoi diagrams using graphics hardware. *Computer Graphics* 1999; **33**: 375–376.
24. Sud A, Gayle R, Andersen E, Guy S, Lin M, Manocha D. Real-time navigation of independent agents using adaptive roadmaps, In *Symposium on Virtual Reality Software and Technology*, Fairmont Newport Beach, California, USA, 2007; 99–106.
25. Levine S, Lee Y, Koltun V, Popović Z. Space-time planning with parameterized locomotion controllers. *Transactions on Graphics (TOG)* 2011; **30**(3): 23:1–23:11.
26. Rusu RB, Sundaresan A, Morisset B, Hauser K, Agrawal M, Latombe J-C. Leaving flatland: efficient real-time three-dimensional perception and motion planning. *Journal of Field Robotics* 2009; **26**(10): 841–862.
27. Teschner M, Kimmerle S, Heidelberger B, Zachmann G, Raghupathi L, Fuhrmann A, Cani MP, Faure F, Magnenat-Thalmann N, Strasser W, *et al.* Collision detection for deformable objects. *Computer Graphics Forum* 2005; **24**(1): 61–81. Wiley Online Library.
28. Pabst S, Koch A, Straßer W. Fast and scalable cpu/gpu collision detection for rigid and deformable surfaces. *Computer Graphics Forum* 2010; **29**(5): 1605–1612. Wiley Online Library.
29. Avril Q, Gouranton V, Arnaldi B. Dynamic Adaptation of Broad Phase Collision Detection Algorithms, In *International Symposium on VR innovation*, Singapore, 2011; 41–47.
30. Devaurs D, Siméon T, Cortés J. Parallelizing RRT on distributed-memory architectures, In *Robotics and automation (ICRA), 2011 IEEE International Conference on*, Shangai, China, 2011; 2261–2266. IEEE.
31. Pan J, Lauterbach C, Manocha D. g-planner: Real-time motion planning and global navigation using GPUs, In *Aaai Conference on Artificial Intelligence*, Atlanta, Georgia, USA, 2010; 1245–1251.
32. Stilman M, Kuffner JJ. Planning among movable obstacles with artificial constraints. *International Journal of Robotics Research* 2008; **27**: 1295-1307.

AUTHORS' BIOGRAPHIES

Thomas Lopez is a research engineer at the INSA de Rennes, IRISA, France. He is a former member of the MimeTIC research team and currently a member of the VR4I team at the IRISA, Rennes, France. He received his PhD in Computer Science from the INSA de Rennes in 2012. His research interests focus on path planning, spatial reasoning, and computer animation.

Fabrice Lamarche is an associate professor at the University of Rennes 1, in France, since September 2004. He received his PhD in Computer Science from the University of Rennes 1 in 2003. He is a member of the MimeTIC research team at IRISA/INRIA Rennes. His research interests include behavioral animation, crowd simulation, and automated analysis of 3D virtual environments for path planning and spatial reasoning. He is involved in several French projects/platforms dealing with virtual human behavior modeling and is cofounder of the Golaem Company focusing on populating 3D worlds with autonomous virtual humans.

Tsai-Yen Li is a Professor in the Computer Science Department of National Chengchi University in Taiwan. He earned his MS and PhD from Stanford University. His research interests include computer animation, digital character, crowd simulation, robot motion planning, virtual environment, intelligent user interface, and interactive storytelling. He is currently a member of ACM, IEEE, TAAI and IICM in Taiwan.