Space-time planning in dynamic environments with unknown evolution.

Thomas LOPEZ[1] and Fabrice LAMARCHE[2] and Tsai-Yen LI[3]

[1]MimeTIC Lab, IRISA / INSA Rennes, France
[2]MimeTIC Lab, IRISA / University of Rennes 1, France
[3]IM Lab, National Chengchi University, Taipei, Taiwan, R.O.C.
{thomas.lopez,fabrice.lamarche}@irisa.fr
li@nccu.edu.tw

Abstract. Numerous path planning solutions have been proposed to solve the navigation problem in static environments, potentially populated with dynamic obstacles. However, in dynamic environments, moving objects can be used to reach new locations. In this paper, we propose an online planning algorithm for dynamically changing environments with unknown evolution. This method focuses on accessibility and on the use of objects movements to reach a given target. Among other examples, we will show that this algorithm is able to find a path through moving platforms to reach a target located on a surface that is never directly accessible. We will also show that the proposed representation enables several kind of adaptations such as avoiding moving obstacles or adapting the character postures to environmental constraints.

Keywords: Path Planning, Dynamic Environments, Autonomous Characters, Accessibility

1 Introduction

In the last decades, path planning has been widely studied and numerous solutions for static environments, eventually populated of dynamic obstacles, have been proposed. Our method addresses a new kind of path planning problem by focusing on a virtual character navigating in dynamically changing environments where the evolution of the topology is not known a priori. Moreover, dynamic objects are not only considered as obstacles but also as navigable parts of the environment that can be used to access new locations such as a plank acting as a bridge and connecting two disconnected regions. Using the character capabilities, our method builds a dual representation of objects which identifies the impact of objects in terms of accessibility and obstruction. This representation is used to characterize and track topological modifications while considering the temporal aspect. Two disconnected regions of the environment can thus be linked thanks to a moving elevator even though no explicit or static path exists between them. We will refer to this situation as the *elevator problem.* We will show that our solution is able to solve such complex cases in real time by generating collision-free paths and adapting the character's postures among dynamic obstacles whose movements are not known a priori. This property makes our algorithm suitable for interactive applications where an external user or a script may act on the environment at runtime. Such an approach is useful in several application fields including video games where non-player characters 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 character representation and the dual representation of the dynamic objects. The next section focuses on the use of those representations to plan a path and adapt postures of a virtual character 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 the critical functionality of autonomy of navigation [13, 15]. 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 specific locations. The general formulation of this problem relies on the exploration of the configuration space (C-space). This C-space is defined as a N-dimensional space for which each of the N axis represents a degree of freedom (DOF) of the character. The C-space is generally divided in C-free, containing valid configurations, and C-obstacle, containing obstructed ones. Thus, planning a collision-free path for a character is equivalent to finding a path in C-free that links two specific 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 [10, 21], or exact, representing C-free using trapezoidal decomposition, Delaunay triangulations and variants [7, 12]. Probabilistic methods, such as PRMs [9, 3] or RRTs [11], 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 [3, 17, 20].

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 considered that obstacles movements are known and use this knowledge to guide and speed-up the path planning [14, 4]. On the other hand, fast replanning techniques are used when an obstacle is detected along the planned trajectory [1, 24]. Various reactive methods based on PRMs [1] or RRTs [6] have been proposed. This kind of methods validates precomputed edges of the PRMs during planning to take dynamic changes into account [8]. Velocity obstacles and extensions [2] propose a reactive approach which 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 [5, 22].

To our knowledge, the first method handling space-time planning in a dynamic and disconnected environment has been recently proposed by Levine et al. [16]. The singularity of both our and their 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. Based on this hypothesis they compute their path using a trial-and-error solver, trying different motion controllers until a correct motion to adopt along the path 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 characterize and track topological relations over time. Based on 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 Characters and objects representations

Our path planning problem considers a non-flying character evolving in a dynamic environment composed of non-deformable objects. We make no assumption on the spatio-temporal evolution of the environment that we only assume to be observable. To propose a solution compatible with interactive applications the character representation is simplified using bounding volumes, which reduce the number of DOF of the problem. Those volumes are used to extract a dual representation of the objects which characterizes the topological impact of an object and enables to rapidly identify accessibility and obstructions.

3.1 The character and its navigation capabilities

We consider a character with navigation capabilities mainly constrained on the floor. In order to speed-up path planning and uncouple it from the animation process, we represent our character using bounding volumes [14, 2, 17] that reduce the number of DOF. For each navigation capability i, we define a cylinder C_i, centered on the character's root, which bounds its geometry when playing the motion M_i. Jump motions are particular as a character 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, we assume that our character follows a ballistic trajectory when jumping.

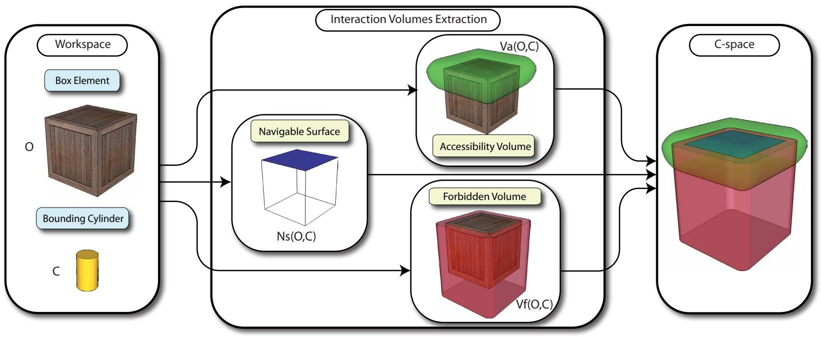

Fig. 1. *Interaction Volumes*: given an original geometry O and a cylinder C bounding a considered motion capability, we compute three *Interaction Volumes*: the *Navigable Surface* $Ns(O, C)$, the Accessibility Volume $Va(O, C)$, and the *Forbidden Volume* $Vf(O, C)$. On the right, the dual representation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

3.2 Augmenting geometry with Interaction Volumes

Definition of the Interaction Volumes From the character's point of view, a geometric object impacts on the local topology in three different ways. It can obstruct a region, present navigable areas or create an access to other surfaces. Obstruction, navigability and accessibility properties rely on the character's navigation capabilities. Regarding those capabilities, we extract a dual representation of objects, called the *Interaction Volumes*, which characterizes feasible, colliding and reachable configurations of the C-space. Once navigable areas of an object have been identified, we associate to each surface a precomputed roadmap.

Given the cylinder bounding a navigation capability of our character, a configuration in this C-space represents the position of the character's root located at the bottom center of the cylinder. 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 cylinder C_m bounding the navigation capability m, we define three types of *Interaction Volumes*(see Fig. 1): the *Forbidden Volume*, the *Navigable Surface* and the *Accessibility Volume*.

Forbidden Volume, denoted $V_f(O, C_m)$, represents the set of configurations where the character 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_m. This shape is then extended along the (X,Y)-axes using the cylinder's radius.

Navigable Surface, denoted $Ns(O, C_m)$, represents the surface where the character can stand. We use an interval of navigable slopes associated to m to determine whether or not a character is able to stand on the considered surface. $Ns(O, C_m)$ is computed by grouping all triangles of the object's mesh with navigable slopes minus configurations lying in $V_f(O, C_m)$.

Accessibility Volumes denoted $V_a(O, C_m)$, contains all configurations reachable from $Ns(O, C_m)$ when jumping. First, the maximum reachable height is

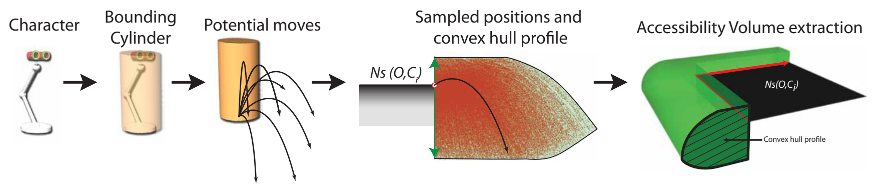

Fig. 2. *Accessibility Volume* definition : regarding a character and its jumping capability, we extract a profile representing its potential jumps from a start configuration. The last scheme presents the profile extraction along edges of $Ns(O, C)$.

used to extrude $Ns(O, C_m)$ along the height axis. Second, given the jump motion characteristics, we compute an Accessibility Profile (see Fig. 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 the $Ns(O, C_m)$ to finalize $V_a(O, C_m)$.

Local roadmap generation Since the global structure of $Ns(O, C)$ 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 Probabilistic RoadMaps method (PRM) to create local roadmaps [9]. We thus randomly sample configurations in $\bigcup_m Ns(O, C_m)$ and annotate each sampled configuration c with the set of motion capabilities that are valid i.e. $\{m | C \in Ns(O, C_m)\}$. Each sample is then connected to its k-nearest neighbors iff the configurations share at least one common motion capability m and that a linear path lying in $Ns(O, C_m)$ exists.

3.3 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. Given two objects (O_i,O_j) and a motion capability m, accessibility and obstruction relations are defined as follow:

- **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 character may have more difficulties to climb on objects than to go down (see Fig. 2).
- **Obstruction**: $O(O_i, O_j, C_m)$, holds when $V_f(O_i, C_m) \cap V_a(O_j, C_m) \neq \emptyset$, i.e., O_i obstructs some navigable parts of O_j for the given capability m.

The identification of those relations coupled with local roadmaps allow us to locate the topological impact of the detected relations at runtime. Thus, an accessibility relation results in a connection between two distinct roadmaps while

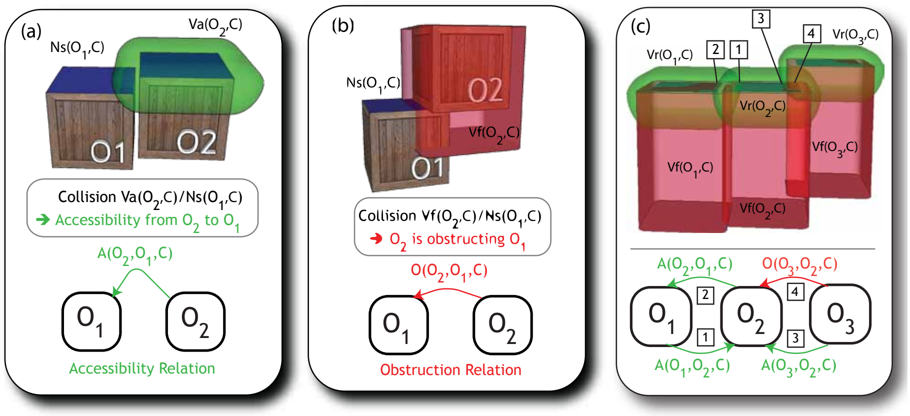

Fig. 3. *Topological Graph* construction. Characterization of an accessibility (a), and of an obstruction (b). Example of a more complex situation (c) with 4 detected relations.

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 character to adopt a crouching capability to navigate along its path. The identification of topological relations is 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. Topology relations need to be tracked in order to consider them during navigation. We now present how the *Interaction Volumes* representation is used to track topology modifications while taking time into account to avoid moving obstacles but also detect moving platforms linking disconnected surfaces. Our two level path planner is then presented. The first level computes a path between *Navigable Surfaces* at the topological level, while the second level plans a local path on each *Navigable Surface*.

4.1 Tracking topology modifications

In order to track the topology over time, 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 concerned objects (see Fig.3). As the character has no a priori knowledge on the environment's evolution, this graph allows it to build

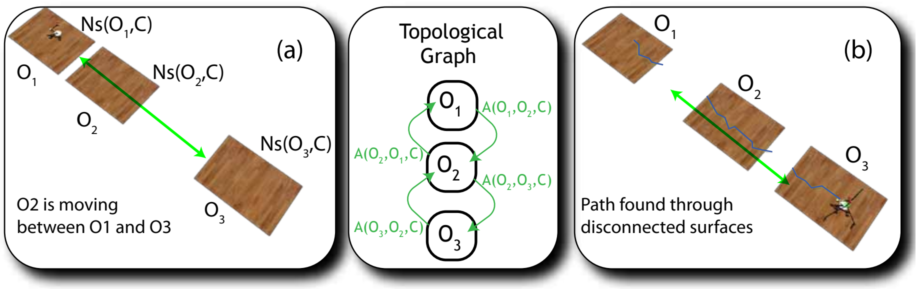

Fig. 4. Navigation through disconnected surfaces using the *Topological Graph*.

its own representation of the environment by observing the evolution of topological relations over time. As described previously, detected collisions between *Interaction Volumes* allows 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 [23]. Every time a relation is detected by the CD, the corresponding edge is added or updated in the *Topological Graph*. 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 its periodicity. 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 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 an anytime representation of the topology. Moreover, the temporal information on relations enables to automatically represent periodical relations in space and time (see Fig. 4).

Nevertheless, this graph is a coarse representation of the global topology as it only identifies relations between pairs of objects. Regarding the definition of accessibility, an object O_j is reachable from O_i with a jump capability C_m iff $V_a(O_i, C_m) \cap Ns(O_j, C_m) \neq \emptyset$. To refine this relation and avoid potential collisions with *Forbidden Volumes*, we randomly sample a set of jumps from O_i to O_j. First, a target configuration c_t is selected from the local roadmap (PRM) associated to O_j. Second, its nearest source configuration c_s belonging to the local roadmap associated to $N_s(O_i, C_m)$ is also selected. Finally, a random configuration c_r is sampled such as it vertically projects on the segment (c_s, c_t) and its Z-coordinate is greater than the maximum Z coordinate of c_s and c_t. Given the configurations c_s, c_r and c_t, the second order polynomial corresponding to the unique ballistic trajectory passing through those three configurations is computed (see Fig. 5(a)). The obtained jump is validated iff the impulse and

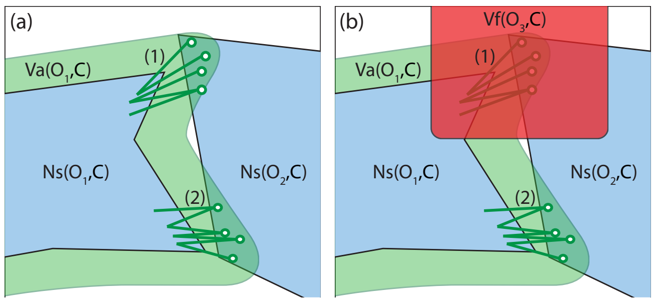

Fig. 5. Reachability from O_1 to O_2 (top view). Several valid configurations are sampled in $Va(O_1, C) \cap Ns(O_2, C)$ and linked to the O_1's roadmap (a). Then relevant *Forbidden Volumes*, here $Vf(O_3, C)$, are retrieved from the *Topological Graph* and obstructed jumps are filtered (b), such as links from the region (1).

landing speeds satisfy the constraints associated to the jumping capability C_m and the trajectory does not collide with a *Forbidden Volume* (see Fig. 5(b)). Each validated jump is then stored in the corresponding accessibility edge of the *Topological Graph*. To amortize the cost of the sampling phase, a sampling budget is allocated at each time step. This budget is then distributed among the currently valid accessibility relations.

4.2 A two-level path planner

In order to find a path in the environment, we designed a two-level path planner. This planner first selects *Navigable Surfaces* on the way, using the *Topological Graph* and the temporal information. Then local paths are computed on each surface using the associated local roadmaps.

In order to identify *Navigable Surfaces* on the way, we first filter the *Topological Graph*. Thus, we discard obstruction relations as well as unfeasible accessibility 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. To compute the global path, we run a Dijkstra algorithm 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 which are always valid). This cost function thus tends to favor paths through stable links and minimizing the waiting time. The global path planner finally provides a sequence of *Navigable Surfaces* that must be crossed in order to reach the goal.

Then, the local path planner has to generate local paths on each roadmap associated to the identified *Navigable Surface* while handling obstacle avoidance and posture adaptation. To compute this path in the local PRM, we use a multi-target A* algorithm that starts from the current configuration of the character 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 is valid if at least one motion capability m associated to the edge does not collide with local *Forbidden Volumes* and if m is compatible with the motion capability used to reach this edge. Edges validity is checked during planning in the space-time domain. In this domain, we anticipate objects positions using a linear extrapolation of their current movements over time [1]. 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, the object is assumed to be static. Once the path is computed, the character follows it. If a new potential collision is detected during navigation, a replanning is executed. When a local target is reached, the character waits to access the next surface. Using the jump properties, the jump decision is taken by extrapolating the position of the targeted surface and nearby obstacles. If the landing area lies on the targeted surface and the trajectory does not collide with obstacles, the character 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 such as detecting a sequence of moving platforms disconnected in space and time that must be crossed to reach a given goal. The search space is also reduced for the local planner which only plans paths and adapts postures on relevant *Navigable Surfaces*.

5 Results

In our test cases, the character uses three navigation capabilities: sliding on the ground while (1) standing or (2) crouching and (3) jumping. The jumping capability allows the character to reach disconnected areas of the workspace. The heights of the bounding cylinders are set to 50 cm for capabilities (1) and (3), 20 cm for capability (2). The radius of those cylinders is set to 20 cm for capability (1) and (3), 30 cm for capability (2). When using the jumping capability, our character is able to jump with maximum vertical and horizontal impulse speeds of $2m.s^{-1}$ and we limit the landing speed to $3m.s^{-1}$. We evaluated our method using different dynamic environments. A demo video presenting our results is available online[1].

Disconnected environment. This environment is composed of disconnected and moving platforms (see Fig. 6(f)). By jumping from platform to platform when connections are identified, the character 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 not directly connected. There is no obstacles in the environment.

Living room. This environment, presented in Fig. 6(a), demonstrates the various properties of our method. It is composed of numerous complex objects:

[1] http://www.irisa.fr/bunraku/GENS/tlopez/video/MIG2011video/MIG2011video.html

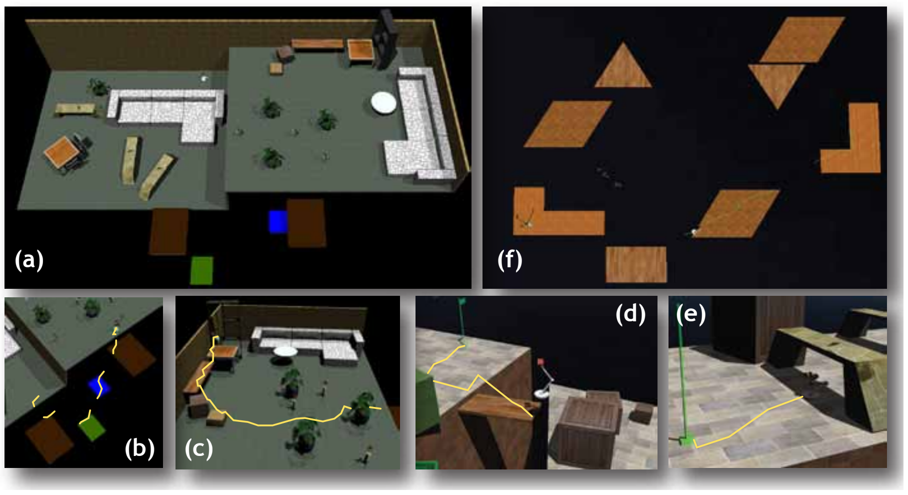

Fig. 6. We present our environments: the living room (a) and the disconnected environment (f). Some results are show such as: navigation between disconnected and moving surfaces (b), the dynamic obstacle avoidance(c), posture adaptation regarding the motion capabilities (d) and the environmental constraints (e).

Table 1. Benchmarks.

Environment Name	Collision Detection	Topological Graph update	average Path Planning time	Obstruction Tests (%path planning)
Disconnected Env	0,75 ms	0,24 ms	6,22 ms	-
Living-room	14,12 ms	6,46 ms	93,98 ms	67,77 %

tables, chairs, sofas, shelves, plants... Those objects all act as obstacles or navigation surfaces and some can constraint the character's postures. The environment is highly constrained leaving only a few room for navigation. Moreover, *flying books* are used as elevators to connect the two floors together. This environment focuses on connections between distinct surfaces, path obstructions replanning and posture adaptation during navigation.

Benchmarks have been realized on an Intel Core i7, CPU X920, 2GHz. We used Bullets Physics CD library to identify *Interaction Volumes* collisions. Our implementation is currently mono-threaded. Our benchmarks results are presented in Table 1. This table summarizes average times of: the CD between *Interaction Volumes*, the *Topological Graph* update and 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 1 show that our algorithm performs topology detection and processes path planning requests at interactive frame rates in our testing environments.

Our algorithm first continuously tracks the collisions between *Interaction Volumes* to identify the evolution of the topology. This process performances are directly correlated with the CD library that is used. The time spent in the graph update is negligible but Table 1 shows that the use of numerous objects with complex geometries (such as the living room) decreases algorithm performances. 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 physical simulations. We could extend those methods to simplify the shapes of the *Interaction Volumes*. Second, we defined a dynamic path planner which is 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 regarding the future obstacle locations. The validity tests represent around 70% of the computation time. Whenever an unexpected obstacle appears on the character's trajectory, a new local path planning request is emitted. To increase performances and avoid redundant computations, a D* algorithm could be used.

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. The analysis of this evolution is then used to solve complex planning problems such as finding a path between regions disconnected in space and time. Moreover, the same representation is used to adapt postures to environmental constraints and locally plan collision-free paths avoiding dynamic obstacles. Finally, contrary to Levine et al. approach [16], our method does not require a prior knowledge on the evolution of the world but builds its knowledge at runtime by observing the environment's evolution to propose a real time path planner.

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 [18] or on GPU-based planning algorithms [19] 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, since 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. Finally, we intend to increase the dynamic and the unpredictability of the environment by considering navigation in physical worlds with physical objects and destructible structures as it can be seen in numerous video games.

References

1. Van den Berg, J., Ferguson, D., Kuffner, J.: Anytime path planning and replanning in dynamic environments. In: Proc. IEEE ICRA (2006)
2. Van den Berg, J., Lin, M., Manocha, D.: Reciprocal velocity obstacles for real-time multi-agent navigation. In: Proc. IEEE ICRA (2008)
3. Choi, M.G., Lee, J., Shin, S.Y.: Planning biped locomotion using motion capture data and probabilistic roadmaps. ACM Transactions on Graphics 22(2) (2003)
4. Gayle, R., Sud, A., Lin, M., Manocha, D.: Reactive deformation roadmaps: motion planning of multiple robots in dynamic environments. In: IEEE IROS (2007)
5. Hoff III, K.E., Keyser, J., Lin, M., Manocha, D., Culver, T.: Fast computation of generalized voronoi diagrams using graphics hardware. Computer Graphics 33 (1999)
6. Jaillet, L., Simeon, T.: A prm-based motion planner for dynamically changing environments. In: IEEE International Conf. IROS (2004)
7. Kallmann, M., Bieri, H., Thalmann, D.: Fully dynamic constrained delaunay triangulations. Geometric Modelling for Scientific Visualization (2003)
8. Kallmann, M., Matarić, M.: Motion planning using dynamic roadmaps. In: Proc. of the International Conference on Robotics and Automation (2004)
9. Kavraki, L., Svestka, P., Latombe, J.C., Overmars, M.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces (1994)
10. Kuffner, J.J.: Goal-directed navigation for animated characters using real-time path planning and control. Lecture Notes in Computer Science 1537 (1998)
11. Kuffner, J.J., LaValle, S.M.: Rrt-connect: An efficient approach to single-query path planning. In: IEEE Int. Conf. on Robotics and Automation (2000)
12. Lamarche, F.: Topoplan: a topological path planner for real time human navigation under floor and ceiling constraints. Computer Graphics Forum (2)
13. Latombe, J.C.: Robot Motion Planning. Kluwer Academic Publishers (1991)
14. Lau, M., Kuffner, J.: Behavior planning for character animation. In: Proc. of Symposium on Computer animation (2005)
15. LaValle, S.M.: Planning Algorithms. Cambridge University Press (2006)
16. Levine, S., Lee, Y., Koltun, V., Popović, Z.: Space-time planning with parameterized locomotion controllers. Transactions on Graphics (TOG) 30(3) (2011)
17. Li, T.Y., Huang, P.Z.: Planning humanoid motions with striding ability in a virtual environment. In: Int. Conf. on Robotics and Automation (2004)
18. Pabst, S., Koch, A., Straßer, W.: Fast and scalable cpu/gpu collision detection for rigid and deformable surfaces. In: Computer Graphics Forum. vol. 29 (2010)
19. Pan, J., Lauterbach, C., Manocha, D.: g-planner: Real-time motion planning and global navigation using gpus. In: AAAI Conference on Artificial Intelligence (2010)
20. Safonova, A., Hodgins, J.K.: Construction and optimal search of interpolated motions graphs. ACM Transactions on Graphics 26(3) (2007)
21. Shiller, Z., Yamane, K., Nakamura, Y.: Planning motion patterns of human figures using a multi-layered grid and the dynamics filter. In: IEEE ICRA (2001)
22. 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 (2007)
23. Teschner, M., Kimmerle, S., Heidelberger, B., Zachmann, G., Raghupathi, L., Fuhrmann, A., Cani, M., Faure, F., Magnenat-Thalmann, N., Strasser, W., et al.: Collision detection for deformable objects. In: CGF. vol. 24 (2005)
24. Zucker, M., Kuffner, J., Branicky, M.: Multipartite rrts for rapid replanning in dynamic environments. In: IEEE ICRA (2007)