Space-time planning in unknown dynamic environments.

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 unknown dynamic environments that 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 kinds 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, presented in this paper, addresses a new kind of path planning problem by focusing on unknown dynamic environments where dynamic objects are not only obstacles but also navigable parts of the environment that can be used to access new locations. Our algorithm identifies the impact of objects in terms of accessibility, such as a plank which can act as a bridge and thus connect two disconnected regions. Moreover, as we consider moving objects, the temporal aspect is also taken into account. Thus, two disconnected regions of the environment can 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*. 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.

Our algorithm has been designed to tackle the problem of a virtual character navigating in a dynamic environment. The environment's evolution over time

is not known a priori as this algorithm is designed for interactive applications where a external user or a script may act directly on the environment at runtime. Our method uses a classical simplification of the character's geometry, known as bounding volumes, to build a dual representation of objects. This representation characterizes the objects topological impacts and is used to identify and track topological modifications over time. This information enables path planning but also posture adaptation in dynamic environments. We will show that our solution is able to solve complex cases, such as the elevator problem, while generating collision-free paths and adapting the character's postures among dynamic obstacles.

In the following, we first present related works. We then present the precomputation steps associated to the character representation and the dual representation of the dynamic objects. We then focus on the use of those representations to plan a path and adapt postures of a virtual character evolving in a dynamic environment. Finally, we present some results and analyze them.

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 collision-free path in a static environment. Proposed methods mostly fall into two categories: cell decomposition and probabilistic methods. The cell decomposition methods are either approximate [10, 20] or exact [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 the C-free and edges are collision-free paths linking two nodes. Most of the methods generally consider navigation in a connected environement. However, few methods focused on static but disconnected environments[3, 17, 19].

The need of planning paths in dynamically changing environments arises in many application fields. Most of the proposed methods focuses on avoiding dynamic obstacles. Various probabilistic methods based on PRMs [1] or RRTs [6] have been proposed. This kind of methods validates pre-computed paths of the PRMs during planning to take dynamic changes into account[8]. Velocity obstacles and extensions [2] focus on the agent speed to generate trajectories avoiding collisions with dynamic obstacles. Methods based on rapidly computed generalized Voronoï diagram have also been proposed [5, 21]. Two techniques

can be distinguished. On the one hand, obstacles movements are considered to be known, using 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, 23].

To our knowledge, the first method for 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 make the strong hypothesis that the objects movements have known trajectories which involves that the future evolutions of the environment are always known. Based on this hypothesis they compute their path using a trial-and-error solver, trying different controllers until a correct motion along the path is found.

Our algorithm identifies and tracks topological relations linking objects in an unknown environment. Thanks to a space-time representation of those relations, the proposed algorithm is able to find paths among moving platforms that links disconnected surfaces while avoiding dynamic obstacles and adapting the character's postures to environmental constraints with no a priori knowledge on the dynamic of the environment.

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 unknown but observable. To propose a solution compatible with interactive applications the number of DOF of the problem is reduced using bounding volumes. In this section, we introduce our character representation and then explain how it is used to extract a dual representation of the objects. This three dimensional representation characterizes the topological impact of an object and enables to rapidly identify accessibility and obstructions.

3.1 The character and its navigation capabilities

By containing all the character's geometry and decoupling animation and path planning, representing a character using bounding volumes simplify and speed-up the path planning process as it reduces the number of DOF [14, 2, 17]. We consider here a character with navigation capabilities mainly constrained on the floor. For each navigation capability i, we define a cylinder C_i, centered on the character's root, by bounding its geometry when playing the motion M_i. However, 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.

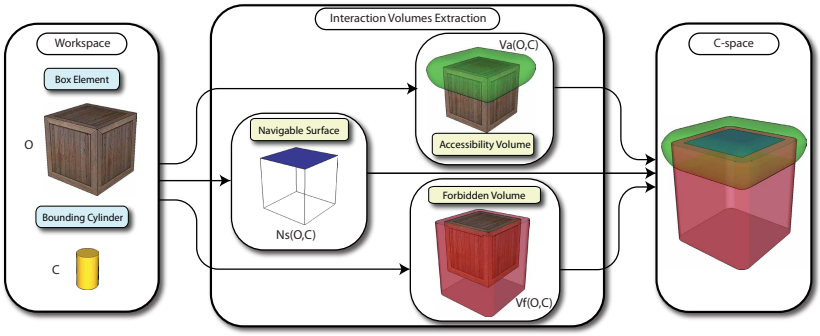

Fig. 1. *Interaction Volumes*: Given an original geometry O and a cylinder C bounding a considered motion capability, we compute three *Interaction Volumes*: the *Navigable Surface* $Ns(O, C)$, the Accessibility Volume $Va(O, C)$, and the *Forbidden Volume* $Vf(O, C)$. On the right, the dual representation 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 impact on the local topology in three different ways. It can obstruct a region, offer 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 pre-computed 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. 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*(cf. Fig.1):

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 used to extrude $Ns(O, C_m)$ along the height axis. Second, given the jump mo-

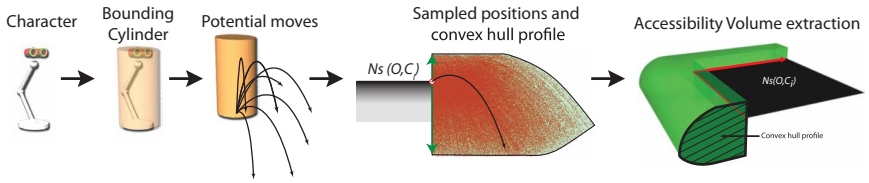

Fig. 2. *Accessibility Volume* definition : Regarding a character and its jumping capability, we extract a profile representing its potential jumps from a start configuration. The last scheme presents the profile extraction along edges of $Ns(O, C)$.

tion characteristics, we compute an Accessibility Profile (cf. 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 a *Navigable Surface* 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**: denoted $A(O_i, O_j, C_m)$., 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 some characters may have more difficulties to climb on objects than to go down (fig.2).
- **Obstruction**: denoted $O(O_i, O_j, C_m)$, 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 allows us to locate the topological impact of the detected relations at runtime. Thus, an accessibility relation results in a connection between two distinct roadmaps whereas

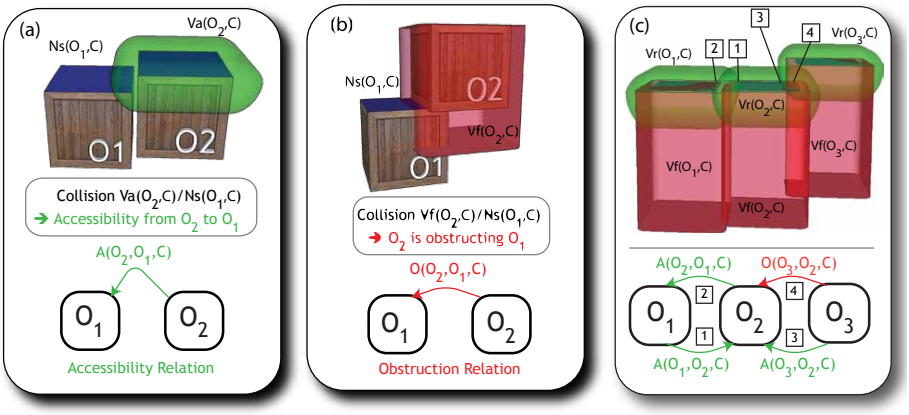

Fig. 3. *Topological Graph* construction. Characterization of an accessibility (a), and of an obstruction (b). Example of a more complex situation (c) with 4 detected relations.

an obstruction invalidates some parts of the roadmap. Obstruction relations 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. Topological relations identification 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 linking surfaces or elevators. 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, when 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 complete representation of the environment's topology by representing each object as a node and each topological relation by a link between the concerned objects (cf Fig.3). As the character has no a priori knowledge on the environment's evolution, this graph allows it to build

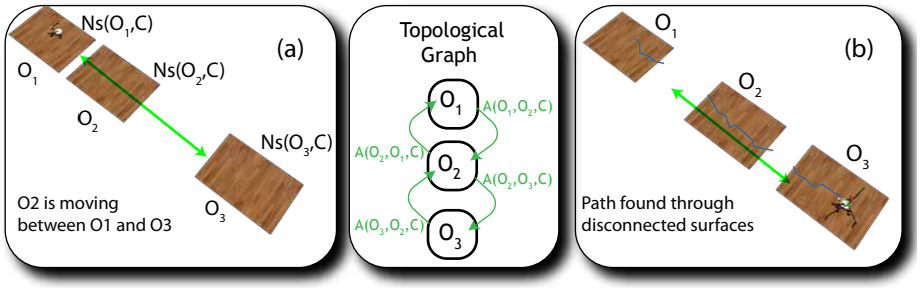

Fig. 4. Navigation through disconnected surfaces is possible thanks to the *Topological Graph* information.

its own representation of the environment by observing the evolution of topological relations over time. As describe previously, collisions between Interaction Volumes at a certain time allows us to identify topological relations present in the environment at this time. Thanks to the 3-dimensionality of the Interaction Volumes, we detect those collisions using a tuned collision detection (CD) algorithm [22]. Every time a relation is detected by the CD, the corresponding edge is detected or updated. Edges are labeled with: the number of times the relation was valid, the mean validity time and the mean non-validity time of the relation and the mean relative speed of the objects. Information concerning the evolution of the environment over time is thus stored in the Topological Graph edges. This point is crucial as we can use this information to characterize topological relations over time. So, the mean validity time of the relation and the mean relative speed of the objects give an estimate of the stability of a relation, while the sum of mean validity and non-validity times give an estimate of its periodicity. Moreover, using also the number of times the relation was valid, this allows the *Topological Graph* to automatically identify and characterize periodic and punctual relations between objects. Thanks to the coupling with the CD algorithm, the *Topological Graph* is an anytime representation of the environment's topology. Moreover, creating edges as soon as a relation is identified and storing inside temporal information on the relation enables to identify objects which periodically link over time regions of the environment (cf Fig.4). Finally, this temporal information can also be used as optimization criteria by the planning algorithm.

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 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 roadmap associated to O_j. Second, its nearest source configuration c_s belonging to the roadmap associated to $N_s(O_i, C_m)$ is also selected. Finally, a random configuration c_r is

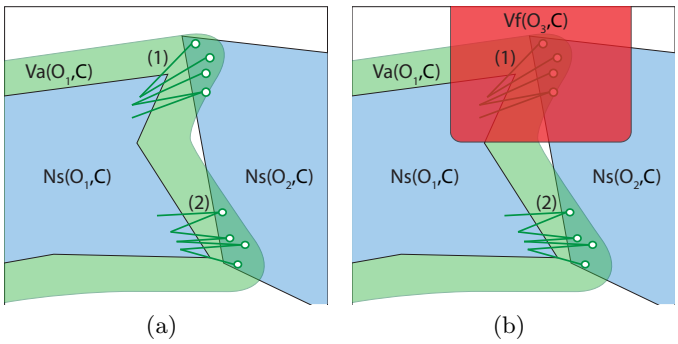

Fig. 5. Reachability from O_1 to O_2 (top view). Several valid configurations are sampled in $Va(O_1, C) \cap Ns(O_2, C)$ and linked to the O_1's roadmap (cf. 5(a)). Then relevant *Forbidden Volumes*, here $Vf(O_3, C)$, are retrieved from the *Topological Graph* and obstructed jumps are filtered (cf.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

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 (cf Fig.5(a)). The obtained trajectory is validated iff the impulse and landing speeds satisfy the constraints associated to the jumping capability C_m and that the trajectory does not collide with a *Forbidden Volume* (cf Fig.5(b)). Each validated jump is then stored in the corresponding accessibility edge of the *Topological Graph*.

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 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 the 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 identified *Navigable Surface* while handling obstacle avoidance and posture adaptations. To compute this path, 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*. Edges validity is checked during planning in the space-time domain. An edge is valid if at least one motion capability m associated to the edge does not collide with local *Forbidden Volumes* and that m is compatible with the motion capability used to reach this edge. In the space-time domain, we predict objects positions using a linear extrapolation of their current motion [1]. However, if the time needed to reach an explored configuration is greater than a defined time anticipation window, the object is considered to be static in order to limit extrapolation errors. When the path is computed, the character follows it. If a new potential collision is detected during navigation, a replanning is executed. Once 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: translating 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 :

Disconnected environment. This environment is composed of disconnected and moving platforms (cf 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:

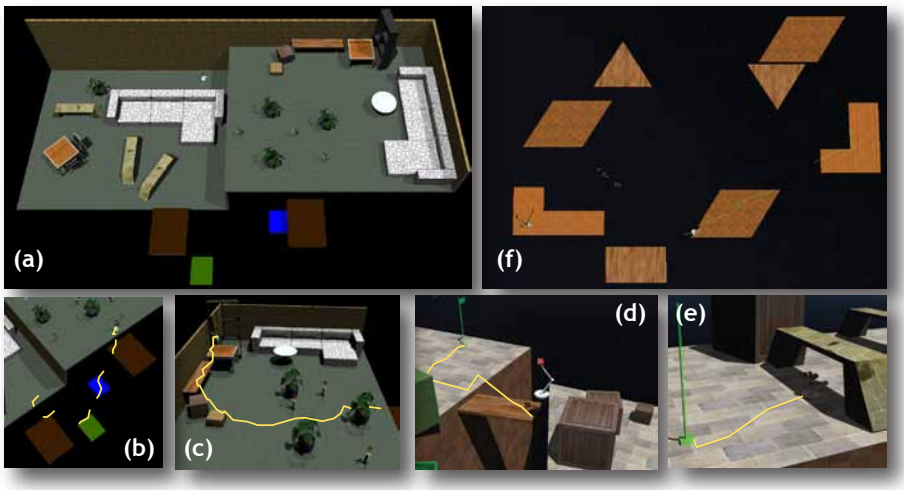

Fig. 6. We present our environments: the living room (a) and the disconnected environment (f). Some results are show such as: navigation between disconnected and moving surfaces (b), the dynamic obstacle avoidance(c), posture adaptation regarding the motion capabilities (d) and the environmental constraints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

Table 1. Benchmarks.

Environment Name	Collision Detection	Topological Graph update	average Path Planning time	Obstruction Tests (%path planning)
Disconnected Env	0,62 ms	0,43 ms	79 ms	-
Living-room	8,72 ms	5,77 ms	163,35 ms	95,6 %

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(R) Core(TM)2 Extrem, CPU X7900, 2.80GHz. 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 95% 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 more adapted.

6 Conclusion and future works

In this paper, we presented our approach to online path planning in dynamic environments. The originality of our approach is to assume that dynamic objects are not only obstacles but can also be used to navigate and reach previously unreachable locations. We showed that, the characterization offered by *Interaction Volumes*, it is possible to track the evolution of the environment's topology. The analysis of this evolution can then be 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 handle posture adaptation to environmental constraints and to locally plan a collision-free path avoiding dynamic obstacles. Finally, our method does not require a prior knowledge on the evolution of the world but build its knowledge on observation at runtime.

The collision detection algorithm, used for topology tracking and local path planning, is a major bottleneck. However, some recent work focusing on collision detection parallelization on CPU / GPU [18] are promising for scaling our algorithm to very complex environments. Another aspect is that sometimes, the character may 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.

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: 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: Proceedings of the IEEE International Conf. on Intelligent Robots and Systems (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. Boston: Kluwer Academic Publishers, Boston (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. Safonova, A., Hodgins, J.K.: Construction and optimal search of interpolated motions graphs. ACM Transactions on Graphics 26(3) (2007)
20. 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)
21. 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)
22. 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)
23. Zucker, M., Kuffner, J., Branicky, M.: Multipartite rrts for rapid replanning in dynamic environments. In: IEEE ICRA (2007)