HAL
archives-ouvertes.fr

# TopoPlan: a topological path planner for real time human navigation under floor and ceiling constraints

Fabrice Lamarche

▶ To cite this version:

Fabrice Lamarche. TopoPlan: a topological path planner for real time human navigation under floor and ceiling constraints. Computer Graphics Forum, Wiley, 2009, 28 (2), http://www3.interscience.wiley.com/journal/122288567/abstract . 10.1111/j.1467-8659.2009.01405.x . inria-00432184

HAL Id: inria-00432184
https://hal.inria.fr/inria-00432184

Submitted on 18 Nov 2009

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# TopoPlan: a topological path planner for real time human navigation under floor and ceiling constraints

F. Lamarche *[1]
[1] Université de Rennes 1

## Abstract

In this article we present TopoPlan, a topological planner dedicated to real-time humanoid path-planning and motion adaptation to floor and ceiling constraints inside complex static environments. This planner analyzes unstructured 3D triangular meshes in order to automatically determine their topology. The analysis is based on a prismatic spatial subdivision which is analyzed, taking into account humanoid characteristics, in order to extract navigable surfaces and precisely identify environmental constraints such as floors, ceilings, walls, steps and bottlenecks. The technique also provides a lightweight roadmap computation covering all accessible free space. We demonstrate the properties of our topological planner within the context of two reactive motion control processes: an on-the-fly trajectory optimization and footprint generation process that correctly handles climbing of complex staircases, and a reactive ceiling adaptation process that handles beam avoidance and motion adaptation to irregular floors and ceilings. We further show that the computation cost of these processes is compatible with the real time animation of several dozens of virtual humans.

## 1 Introduction

One of the goal of behavioral animation is to automate the process of populating a virtual environment with autonomous virtual humans. Models used to describe humanoid behavior have to combine long term planning and reactivity to enable local adaptation to different environmental constraints. One of the most important behaviors is the ability to navigate inside a virtual environment. Like humans, humanoids should be able to walk upstairs or downstairs, avoid beams and adapt their posture to avoid collisions with an irregular ceiling. This natural behavior actually needs a huge effort to be mimicked.

Virtual environments are provided as 3D triangular meshes modeled by designers (architects, graphics designers...). In order to endow a virtual human with a navigation process, this 3D triangular mesh needs to be analyzed in order to produce efficient structures enabling fast and accurate path planning inside constrained environments. On the other hand, humanoid motion should be adapted to match constraints imposed by the environment (beam avoidance, stair climbing...). To the best of our knowledge, few models propose to solve all those problems in a unified architecture and none is capable of generating such motions for several dozens of humanoids in real time.

In this paper, we address the problem of path planning and motion adaptation in complex 3D environments. We propose an architecture handling spatial analysis, accurate path planning and motion adaptation to environmental constraints. Proposed algorithms make a compromise between exact environment representation and performance. Environment representation is based on an exact spatial subdivision which is analyzed in order to extract environment topology and generate a concise and accurate roadmap. Based on this representation, several runtime algorithms are proposed: optimized path generation, accurate footprint placement and posture adaptation to floor and ceiling constraints. Those algorithms have been designed to match behavioral animation requirements: performance and reactivity.

This paper is organized as follows. In the next section, previous works centered on environment representation, path planning and motion adaptation during locomotion are presented. The following sections detail the environment representation and proposed mo-

*e-mail: fabrice.lamarche@irisa.fr

tion control processes. Finally, results and performance analysis are discussed.

## 2 Related works

Path planning and environment representation have been widely studied in the field of robotics where navigation is a necessary task to achieve [Lat91, LaV06]. The aim of the proposed methods is to represent the environment free space. Three general approaches, differing in their representation of free space, can be distinguished: roadmaps, cell decomposition and potential fields. The **roadmap** approach captures the connectivity of the free space by using a network of standardized paths (lines, curves). Different approaches can be used to compute roadmaps. The visibility graph [ACF01] connects together vertices of the environment geometry if and only if they see each other. This approach ensures finding the shortest path between two configuration but can be memory consuming in open environments with sparse obstacles. An alternative is to compute the generalized Voronoi diagram [HIKL*99] inside the free space [SGA*07]. The property of such a method is to generate paths maximizing clearance with obstacles. The probabilistic roadmap consists of generating non colliding configurations by randomly sampling the free space and linking those configurations if a collision-free linear path connects them. Such method has been used to plan paths inside 2D virtual environments [BLA02] or huge 3D environments [SGLM03] and has demonstrated its capacity to deal with relatively high dimensional problems [KSLO94]. A combination of random sampling and Voronoi diagrams have also been used in the AERO system to compute adaptative roadmaps in order to interactively plan paths inside dynamic environments [SAC*07]. The Rapidly exploring Random Trees (RRT) [KL00] is a single query method which tries to cover free space with a tree of random configurations. The root of the tree is the initial configuration whereas leaves converge to the goal. The **cell decomposition** method consists of decomposing free spaces into cells. Two general methods can be distinguished: the approximate cell decomposition and the exact cell decomposition. The approximate cell decomposition consists of using predefined cell shapes, whose union is strictly included in the free space (uniform grids, quadtrees, circles) [BT98, Kuf98, SYN01, PLT06, ST06]. The exact cell decomposition consists of computing cells such that their union is exactly the free space (constrained Delaunay triangulation, convex polygons, trapezoidal) [KBT03, LD04]. The representation complexity of the latter method is directly dependent on the input geometry complexity. Finally, the **potential field** method consists of computing an artificial potential function in order to guide the entity to its goal by using gradient methods. The main problem of this technique is that it is subject to the local minimum problem and does not necessary reach the goal. This problem can be partially solved by providing good potential functions or by executing random moves [KKL96].

Coupled with path planning, a second problem arises: how can we generate plausible motions inside complex environments such as environments with stairs, beams or pillars? In order to generate plausible animations, multi stage methods have been proposed. A humanoid is approximated by a bounding volume (generally a cylinder) in order to compute a collision free path inside the environment. Coupled with global path planning strategies, motion controllers [Kuf98] or local planning methods are used to adapt the humanoid motion to environmental constraints. The approach of Pettré et al. [PLS03] consists of using a probabilistic roadmap in order to plan a global path. Once the path is computed, a global motion is generated and body parts colliding with obstacles are identified. Those colliding parts are then modified by using a randomized search in order to avoid obstacles colliding with the upper part of the body. The resulting motion is then warped with the initial motion. In [CLS03], sequences of valid footprints are searched through a probabilistic roadmap augmented with a posture transition graph. Each pair of footprints is then substituted with corresponding motion clips. This approach tends to generate realistic motions and is able to generate locomotion on uneven grounds as well as jumps over holes.The approach of Li consists of representing the free space by using regular multi-layered grids and a cost function optimization to compute footprints [LCH03] even on unconnected surfaces [LH04]. An inverse kinematics (IK) method is then employed to generate human locomotion. In the work of Shiller et al. [SYN01], a multi-layer grid is used to represent the configuration space for a humanoid with different locomotion such as walking and crawling. The humanoid is able to change its posture along a global path by using several bounding volumes extracted from the predefined locomotion behavior. The approach proposed in [LK05] uses finite state machines in order to encode possible transitions between motions traducing possible motion / state transitions. This state

machine is used by a search technique ($A^*$ and variants) in order to provide possible states. The environment is represented by a 2D height field grid map. This map encodes free space, obstacles and their proximity and also contains information about special obstacles the character can crawl under. This environment representation is used to prune the exploration tree. This technique has been extended in [LK06] by using a precomputed exploration tree to improve the system performance.

Techniques handling motion adaptation during locomotion use a two step process. The first step relies on a specific environment representation on which global path planning is based. The second step uses this representation in conjunction with specific algorithms to select and / or adapt motion to environmental constraints. Most approaches use either probabilistic roadmaps or regular grids to represent the environment. Probabilistic roadmaps enable path planning in high dimensional configuration spaces but raise several problems. The most important one is related to the probability density during sampling. The process may fail to capture the connectivity of a very constrained environment if probability densities are not carefully tuned [KKL96, SLCP05]. Moreover, suppose that the environment contains a constrained spiral staircase. Without further environment analysis, it is hard to ensure that each step will be identified. On the other hand, using regular grids may provide better results but this technique is very sensitive to the grid resolution. A precise grid resolution, as well as high probability densities during sampling, generates precise representations. But it also increase memory consumption and decrease planning performance. Techniques used to select / adapt humanoid motions fall into one of the three following categories: search of an adequate motion in order to enforce the quality of the resulting animation [CLS03, LK05, LCL05, SH07]; use of a bounding volume to avoid collision [SYN01]; use of a randomized search in order to generate collision free motion [PLS03, YKH04]. Processes using motion adaptation are the more general but are generally non real-time for one character and do not scale to the animation of several dozens of characters in real time. Finally some hard navigation problems such as climbing a constrained spiral staircase, walking on a curved ground while avoiding collisions with a curved ceiling or avoiding a beam while climbing a step have not been demonstrated.

In the following, we will show that our approach based on an exact 3D spatial subdivision of a static environment is able to deal with such problems. It generates accurate and concise roadmaps covering all accessible zones and enables the creation of reactive motion control processes handling path optimization, accurate footprint placement and posture adaptation under floor and ceiling constraints in real time.

# 3 Environment representation

The aim of the proposed environment representation is to exactly represent the free space of an unstructured $3D$ database while providing efficient data structures dedicated to environment analysis, environmental queries and path planning. Our representation is defined by several complementary visions of the same environment. Firstly, an exact $3D$ prismatic spatial subdivision of the 3D database is computed. Based on this $3D$ spatial subdivision and some humanoid characteristics, a topology identifying interconnected continuous navigable zones is computed. This topology is then summarized in $2D$ topological maps that accurately identify obstacles and steps borders as well as bottlenecks. Those topological maps enable the reduction of $3D$ computational problems to simpler $2D$ problems and are a prerequisite for the computation of lightweight roadmaps which automatically cover all accessible free space. In the following, each component of this environment representation is discussed.

## 3.1 Input geometry and preprocessing

Our spatial subdivision process does not make any assumption on the input geometry. However, some requirement has to be fulfilled before computation. Firstly, the geometry has to be described by a set of unorganized triangles. Secondly, this mesh has to be clean, i.e. it does not contain degenerate triangles and all triangles intersections have to be materialized by a shared vertex or a shared edge. The required cleaning step can be achieved by classical $3D$ modelers or by using computational geometry libraries such as CGAL[1]. Once a clean mesh is computed, we apply a simplification step consisting in representing the mesh with complex 3D planar polygons instead of triangles. Those polygons are computed by partitioning the set of mesh triangles into sets of coplanar and connected triangles knowing that two triangles are connected if and only if

[1] http://www.cgal.org

they share a common edge and if this edge is not shared by another triangle of the input geometry.

## 3.2 Prismatic spatial subdivision

The aim of our prismatic spatial subdivision process is to organize a set of $3D$ polygons in order to capture ground connectivity and identify floor and ceiling constraints. It represents the environment by a set of vertical $3D$ prisms dividing the $3D$ database into layers.

In order to explain our algorithm, we will use the example presented Fig. 1. Step (1) presents a simple environment composed of two triangles. The first step of the prismatic spatial subdivision computation consists of projecting the boundaries of each 3D planar polygon on the XY plane (the ground plane). This produces a set of 2D segments (3) on which a constrained Delaunay triangulation [BY98] is applied (4). This triangulation defines a triangle partition $T_2$ of the convex hull of the environment projected onto the XY plane. An important resulting property is that each 3D polygon of the environment mesh exactly projects itself on the union of one or several triangles of $T_2$. The prismatic subdivision is then obtained by associating to each 2D triangle of $T_2$, the set of 3D polygons partially projecting on it. Technically, the latter relation can be computed by using ray casting techniques. Once this relation is computed, for each triangle $t$ of $T_2$ and for each associated polygon $p$, we compute a triangle (called 3D cell) such as this triangle is supported by the plane supporting $p$ and its projection on XY plane is exactly $t$. The resulting subdivision is illustrated by step (5) in Fig. 1. Let $prism(t)$ be the relation giving the list of all 3D cells which are exactly projecting on a triangle $t$ of $T_2$. The relation *prism*, coupled with the triangular map $T_2$ defines the prismatic spatial subdivision. In this spatial subdivision, vertical polygons are not represented as they project on a single segment. Vertical polygons constrain the accessibility between cells, thus they will be taken into account during the topology extraction phase. As shown on the example of Fig. 1, the extracted 3D cells perfectly identify overlapping parts of the environment geometry with cells belonging to the same prisms. Each prism defines several layers of 3D cells, knowing that if a 3D cell is surmounted by another one, the lower cell can be identified as a potential floor whereas the upper one corresponds to its associated ceiling.

Due to the properties ensured by the geometry preprocessing (all faces intersections are materialized

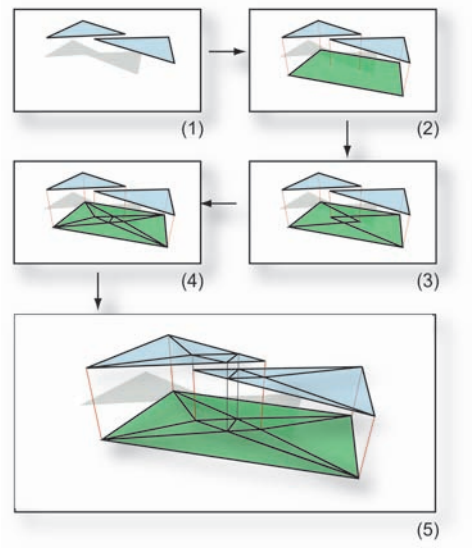


Figure 1: Different steps of the 3D subdivision process.

thanks to a shared vertex or a shared edge) and to the spatial subdivision construction, cells belonging to $prism(t)$ can be ordered along the Z axis. Thus, for each triangle $t \in T_2$, we sort the list of 3D cells of $prism(t)$ in the increasing order of the average Z coordinates of the vertices of the 3D cells. Let consider a triangle $t \in T_2$, and two cells $(c_1, c_2)$ belonging to $prism(t)$. If $c_2$ is after $c_1$ in $prism(t)$, then $c_2$ is over $c_1$. Moreover, if $c_2$ is immediately after $c_1$ in $prism(t)$ then $c_2$ is the ceiling corresponding to the floor defined by $c_1$. The Fig. 3 presents a top view of the prismatic spatial subdivision of our testing environment along with two sample prims identifying floor and ceiling constraints under a beam.

This spatial subdivision has two good properties in terms of topology computation. Firstly, the triangle map $T_2$ coupled with the *prism* relation identifies floor and ceiling enclosing each point of the environment. Secondly, the triangle partition $T_2$ defines a 2D connectivity between prisms of the environment. This property can be used to extract the environment topology by possibly connecting cells belonging to two adjacent prisms.

## 3.3 Topology and 2.5D surfaces

The aim of the topology computation is to characterize navigable zones of the environment with respect to humanoid characteristics. This process determines 3D cells spatial relations and identifies homogeneous and continuous navigable surfaces.

In order to characterize the topology with respect to humanoid characteristics, several parameters have to be defined: the minimal navigable height ($H_m$), the maximal navigable slope ($S_m$) and the maximal height of a surmountable step ($H_m^s$). Moreover, several functions related to the cells of the $3D$ subdivision are also defined. Let $\mathcal{C}_S(c)$ be a function returning the slope of cell $c$ and $\mathcal{C}_H(c)$ be a function returning the minimal height between the floor cell $c$ and its ceiling. Let $up(c)$ be a function returning the ceiling $3D$ cell associated to $3D$ cell $c$ and $E(c,s)$ a function returning the segment of the cell $c$ projecting on the $2D$ segment $s$.

The topology of a 3D environment is represented by a graph in which nodes are navigable 3D cells and edges represent the accessibility between cells. Thus, two cells $c_1$ and $c_2$ belong to the graph and are connected if and only if:

- Cells $c_1$ and $c_2$ are navigable, i.e. $\forall c \in \{c_1, c_2\}$, $\mathcal{C}_H(c) \geq H_m \wedge \mathcal{C}_S(c) \leq S_m$
- The $2D$ projections of $c_1$ and $c_2$ are two triangles of $T_2$ sharing a common edge $S$. This property verifies the $2D$ accessibility between cells.
- The maximal vertical distance between $E(c_1,S)$ and $E(c_2,S)$ is less than the maximum surmountable step height $H_m^s$.
- The minimal vertical distance between $E(c_1,S)$ and $E(up(c_2),S)$ and between $E(up(c_1),S)$ and $E(c_2,S)$ is greater than the minimal navigable height. This property ensures that the humanoid can pass under the ceiling located on the frontier of the two cells.
- The vertical zone delimited by the upper segment between $E(c_1,S)$ and $E(c_2,S)$ noted $V_s$, and segment $V_s$ translated along the Z axis with an offset equivalent to the minimal navigable height, do not contain any vertical obstacle. This property compensates the lack of representation of vertical obstacles in the 3D spatial subdivision by verifying if a vertical polygon of the environment mesh is situated on the common boundary of the two cells.

An edge added in the topological graph is then tagged *continuous* if the maximal distance between $E(c_1,S)$ and $E(c_2,S)$ is lesser than a given threshold or *step* otherwise. Each connected component of the topological graph defines a surface. Let us consider the environment of Fig. 2 containing several stairs (regular, irregular, spiral), beams and narrow passages. Figure 3

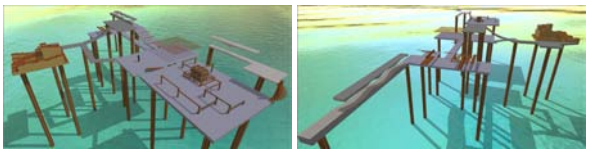

Figure 2: Views of our testing environment.

presents the 2D projection of all 3D cells belonging to a connected component of the topological graph which defines the navigable surface of this testing environment.

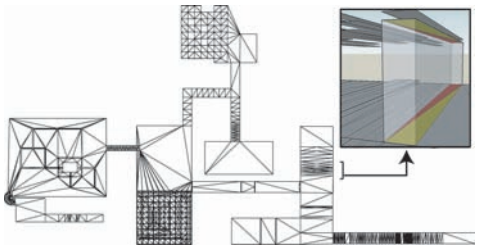

Figure 3: Projection of the 3D cells compounding the $2.5D$ surface extracted from the environment of fig. 2 along with two sample prisms of the prismatic subdivision.

Once the topological graph is computed, zones representing homogeneous groups of 3D cells can be extracted. Those zones are defined as groups of cells linked by a *continuous* relation, i.e. an edge tagged as *continuous*. Those zones are then grouped into a set of $2.5D$ surfaces. A $2.5D$ surface is defined as the union of interconnected zones that do not overlap, i.e. none of the 3D cells belong to the same prism of the spatial subdivision. In case of 3D environments, a connected component of the topological graph will then be represented as a set of layers defining several $2.5D$ surfaces. This property reduces problem dimensionality and enables some data structure optimization and constraint extraction that will be presented in the following sections. Fig. 4 presents the different zones corresponding to the navigable surface presented in figure 3. Each zone is colored and dark lines represent zones borders. As shown in this example, extracted zones match the different continuous surfaces and identify each step of a stair, each platform of the block maze and continuous surfaces. Moreover, it appears that $2.5D$ surface and zone computation is a first step toward a drastically

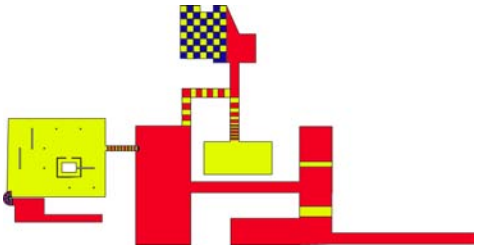

Figure 4: Example of zones computed on the navigable part of our testing environment.

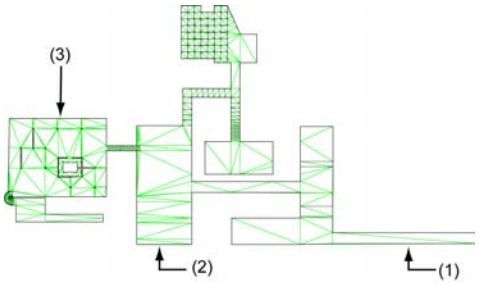


Figure 5: Computed topological map.

simpler representation of the environment.

## 3.4 Topological maps

The aim of topological maps is to provide a compact and accurate representation of a $2.5D$ surface by precisely characterizing obstacle and step borders while improving the environment representation with the computation of bottlenecks, i.e. the most constrained part of the environment. It is a $2D$ representation of the environment topology enabling to reduce a large part of $3D$ problems to simpler $2D$ problems.

As presented in the previous section, a $2.5D$ surface is composed of several interconnected zones. The definition of the topological map relies on the spatial subdivision of segments defining the boundaries of zones compounding a given $2.5D$ surface. Zone boundaries represent either a *step* border if they correspond to an edge linking two cells in the topological graph or an *obstacle* border otherwise. Let $B_s$ be the 2D segments corresponding to the projection of *step* borders and $B_o$ be the 2D segments corresponding to the projection of *obstacle* borders. Sets $B_s$ and $B_o$ are simplified by merging collinear segments sharing a common point. This tends to highly reduce the number of segments and reduce the complexity of the topological map. The topological map is then computed by using a constrained Delaunay triangulation computed on the segments belonging to sets $B_s$ and $B_o$. In order to conserve *step* and *obstacle* border information, the membership of a segment to set $B_o$ or $B_s$ is kept during computation. Once the triangulation is computed, the algorithm proposed in [LD04] is used to identify bottlenecks. The resulting $2D$ triangulation defines the topological map. Triangular cells compounding this map are delimited by three types of segments: $obstacle$, $step$ and $free$ segments. It also accurately identify bottlenecks constraining the navigation on even or uneven floors.

Figure 5 shows the topological map computed on our testing environment: black segments are obstacle borders, dashed segments are step borders and green segments represent $free$ edges, added by the Delaunay triangulation. Compared to the complexity of the surface presented in Fig. 3 we can rapidly conclude that the topological map is drastically simpler (2406 cells vs 7038 original $3D$ cells). In zones (1) and (2), the projection of ceiling constraints and floor irregularities has been removed. Those results are due to the zone border identification combined with aligned segment simplification. Moreover, in zone (3), bottlenecks have been clearly identified.

Due to the analysis of the environment topology, the topological map abstracts the environment spatial subdivision by removing the projection of ceiling constraints, which results in a simpler representation. Furthermore, this map precisely represents the environment and contains all information needed by path planning and footprint generation processes: it identifies steps, obstacles and bottlenecks.

## 3.5 Roadmap

The aim of the roadmap generation process is to limit the number of waypoints and paths while ensuring a precise coverage of the environment. Classically, this process uses two computational steps: the generation of waypoints on the edges of the topological map and the generation of linear paths connecting them.

In order to generate waypoints, each *border* and *free* segment of the topological map is sampled in order to compute a histogram $H(S,u)$, where $S$ is the sampled segment, $u$ is the curvilinear abscissa on segment $S$ and $H(S,u)$ is the distance to the nearest *obstacle* segment. Once the histogram computed, all values of $u$ such as $H(S,u)$ is a local maximum and $H(S,u)$ is greater than

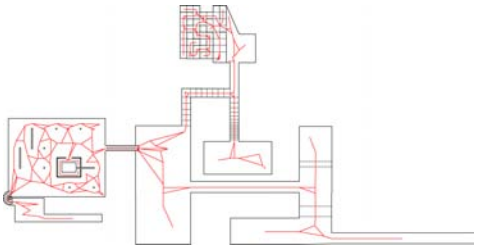

Figure 6: Topological map with associated roadmap.

the humanoid radius are extracted. A waypoint is then generated for each extracted value of $u$. Once all waypoints are generated, a linear path connecting two waypoints ($w_1$, $w_2$) is generated if and only if the distance between the segment ($w_1$,$w_2$) and *obstacle* segments is greater than the humanoid radius and if $w_1$ and $w_2$ have been generated on two different segments delimiting the same triangular cell of the topological map. This waypoint generation process tends to generate waypoints maximizing clearance with obstacles. The generated roadmap is thus a coarse approximation of the generalized Voronoï diagram computed on the set of *obstacle* segments. An example of the roadmap generation is presented in fig. 6. On the spiral staircase (on the bottom left of the figure), this example demonstrates the capacity of our algorithm to find paths within narrow passages.

# 4 Virtual human navigation

Built upon the representation described in the previous sections, three animation processes will be described: optimized path generation, footprint generation and ceiling obstacle avoidance. Those processes have been designed to respect behavioral animation requirements: reactivity and performance. Optimized path generation and footprint generation have been designed to be compatible with producer / consumer architecture. This makes them compatible with other reactive processes such as, for instance, reactive navigation. The ceiling avoidance algorithm is described through and reactive process endowed with anticipation capabilities. It acts independently by modifying the posture to match environmental constraints.

## 4.1 Path generation and optimization

Given a topological map, the aim of our path generation process is to generate a pseudo optimal trajectory linking a source and a destination point. The path planning process starts by localizing the source and destination points inside the topological map, i.e. finding the cells containing them. Those points are then connected to the way points generated on the segments delimiting their respective cells. The path is then computed using an $A^*$ algorithm [**?**]. The resulting path is a list of waypoints which is transformed in a parameterized poly-line $P(u)$ with $u$ varying in the interval $[0;L]$; $L$ being the path length.

Given the position of the virtual human inside the environment, the aim of this algorithm is to find the furthest accessible position belonging to the planned path $P(u)$. Let $t$ be the curvilinear abscissa of the current target on the path (with $t = 0$ when initializing the algorithm). In order to find the furthest accessible target along the path $P(u)$, we use a dichotomy to maximize the parameter $\alpha \in [0;1]$ such as $P(t+\alpha(L-t))$ is accessible. Then $t$ is given the value $t+\alpha(L-t)$ and the new target becomes $P(t)$. The notion of accessibility is defined as follow: a target position is accessible from a source position (generally the humanoid position) if the distance between the segment (source, target) and obstacles is greater than the humanoid radius (this property is verified within the topological map). This ensures that the virtual human can move along a straight line toward the target without colliding with obstacles. Given a new position of the humanoid, a new optimization starting with the current value of $t$ can be called in order to generate a new target.

The performance of this algorithm is limited by the cost of the accessibility query. This spatial query is efficiently implemented by locating a point of the segment inside the topological map and by using a flood fill algorithm to search for obstacles. This algorithm starts from the triangle containing the selected position and explores triangles lying in a zone situated at the given distance from the segment.

## 4.2 Footprint generation

Footprint generation aims at correctly handling footprint placement over steps while following the optimized trajectory. It uses the topological map to extract environmental constraints and correct footprint position.

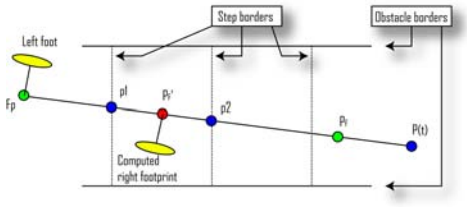


Figure 7: Footprint computation on a regular stair.

This process uses two parameters that can be automatically computed knowing the humanoid morphology and the motion capture in use: the preferred step length $S_l$ and the preferred step width $S_w$. During the following explanation, the reader is invited to refer to Fig. 7 which presents the computation of an adapted footprint. The first step of the algorithm is to compute the new target $P(t)$ using the trajectory optimization process, using $F_p$ as the humanoid position. Once the target computed, a maximal step length $S^l$ is defined such as $S^l = \min(|F_p - P(t)|, S_l)$. Let $D$ be the normalized vector defining the movement direction between $F_p$ and $P(t)$. The next potential position $P_F$ is then defined as $P_F = F_p + S^l D$. Within the topological map, all *border* edges intersecting segment $(F_p, P_F)$ are selected. If the number of intersections is greater or equal to 2, there exists a step between $F_p$ and $P_F$. In that case, intersection points $p_1$ and $p_2$ minimizing the distance to $F_p$ are computed and $P_F$ (noted $P'_F$ in Fig. 7) is defined as the middle point of the segment $(p_1, p_2)$. This position is thus located on the step. The final footprint position is then defined as $P_F \pm S * D \times R(\pi/2)$ where $R(\pi/2)$ is a rotation matrix and $S$ is the minimum value between the preferred step width $S_w$ and the distance to the nearest *border* edge minus the foot width.

## 4.3 Virtual ceiling

The ceiling avoidance algorithm is based on the definition of a virtual ceiling whose role is to smooth transitions between ceiling obstacles. This smoothing process spatially extends ceiling constraints and thus solves anticipation problems during ceiling collision avoidance.

The height of the virtual ceiling is evaluated by a local function taking into account ceiling obstacles situated within a given volume. Let $H_h$ be the humanoid height, $P$ a point of the environment, $r$ a radius and $D_A$ an anticipation distance. Let $\mathcal{C}_H(c)$ be a function returning the minimal height between cell $c$ and its associated ceiling. Firstly a set $C_o$ of floor cells belonging to a given $2.5D$ surface is extracted. Each cell $c$ of $C_o$ respects the following constraints: $c$ lies in an infinite vertical cylinder centered on $P$ with a radius of $r + D_A$ and $\mathcal{C}_H(c) < H_h$. Let $\mathcal{D}_2(a,b)$ be a function returning the minimal distance between the $2D$ projection of $3D$ primitives $a$ and $b$. The height of the virtual ceiling is defined as follow:

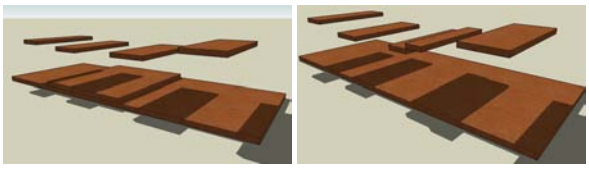

Figure 8: 3D view of the environment (left). Flatten floor and morphed ceiling (right).

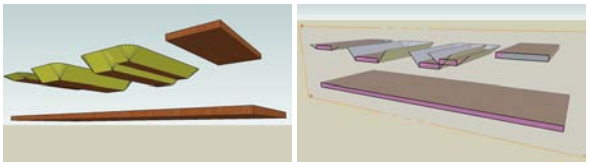

Figure 9: Computed virtual ceiling from morphed environment of fig. 8(b) and associated section.

$$H(P,r) = \min_{c \in C_o} \frac{\max(\mathcal{D}_2(P,c) - r, 0)}{D_A} * (H_h - \mathcal{C}_H(c)) + \mathcal{C}_H(c) \quad (1)$$

Let consider the example presented in Fig. 8. On the left, an environment composed of a flat floor with a rectangular step surmounted by several ceiling obstacles is presented. The previous equation uses the projection of 3D cells compounding the $2.5D$ surface and the height between the floor and ceiling. During the computation, the algorithm reduces the problem to a flatten floor with an associated ceiling deformation such as presented on the right of figure 8. Finally, equation 1 creates a virtual ceiling smoothing constraints using a linear height interpolation around constrained zones of the environment. Figure 9 presents the virtual ceiling and its section, computed on the basis of the environment of fig. 8.

Given the position of a body part of a humanoid, the corresponding height of the virtual ceiling can be rapidly computed. By applying constraints on this body

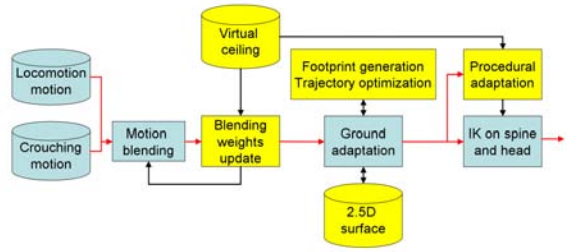


Figure 10: MKM (blue) and TopoPlan (yellow) processes in animation pipeline. Red arrows correspond to posture communication between processes.

part, a motion anticipating and avoiding collision with the ceiling can be generated. Finally, as the height of the virtual ceiling is computed by using the height between floor and ceiling, uneven surfaces surmounted by irregular ceilings are automatically handled and do not need specific computation.

### 4.4 Virtual human animation

To animate our virtual human, we use MKM [KM05, KMA05]. This system proposes several functionalities such as motion blending, real time motion retargeting, fast IK solver and ground adaptation. Coupled with MKM, two control processes are used. The first process handles locomotion and footprint generation while the second one handles ceiling avoidance. The structure of the animation pipeline is presented Fig. 10.

**Locomotion**. When a moving request is received, the global path is computed. Then, the trajectory optimization and the footprint generation processes are started. A cyclic locomotion motion is played and the trajectory optimization and the footprint generation processes are activated when a new footprint is required.

**Ceiling collision avoidance**. The ceiling collision avoidance process combines two techniques: the control of blending between locomotion and crouching locomotion and a procedural control of the humanoid head position. If the head position resulting from the last blending is under the virtual ceiling, the weight of the crouching locomotion is decreased otherwise the weight of the crouching locomotion is increased. This computation tends to smoothly adapt the humanoid animation to ceiling constraints. As a result, the head can collide the virtual ceiling. In such a case, we generate a new head position in front of the humanoid. Let $D_H(p)$ be a function returning the distance between a 3D point $p$ and its projection on the floor. Let $P_H$ be the head position and $r_H$ be the radius of a sphere approximating the head shape. The distance between the top of the head and the virtual ceiling is given by the following equation:

$$\Delta_H(P_H, r_H) = |H(P_H, r_H) - D_H(P_H) - r_H| \quad (2)$$

Let $\delta = \Delta_H(P_H, r_H)$ be the distance after blending between the head and the virtual ceiling. Let $f$ be the normalized front vector of the humanoid and $Z$ be the normalized Z-up vector. The head position is computed as follow:

$$P'_H = P_H + \delta f - \Delta_H(P_H + \frac{1}{2}\delta f, \frac{1}{2}\delta + r_H)Z \quad (3)$$

In this equation, two virtual ceiling height queries are used. The first one $\delta = \Delta_H(P_H, r_H)$ select minimal ceiling height over the head of the humanoid. The second one $\Delta_H(P_H + \frac{1}{2}\delta f, \frac{1}{2}\delta + r_H)$ uses a selection over a cylinder centered on the middle of the original head position and the translated head position with a radius covering all the zone possibly occupied by the head between those two positions. This ensures that the head cannot collide the ceiling. An IK on spine and head is then used to modify the posture of the virtual humanoid.

The Fig. 11 presents some results of motion adaptation to floor and ceilings constraints. One important characteristic of the presented motion control techniques is that they automatically adapt to new motions and character morphologies. This enables to use the same algorithm on a wide variety of humanoids and motions without any pre-computation. The other important characteristic is their reactivity. Computation is achieved online and do not require long term anticipation nor planning. Footprint generation only requires a target which can be computed by the proposed trajectory optimization process but also by an external reactive navigation model for example. Ceiling adaptation does not use randomized search nor motion bounding boxes and do not need a fully planned path to adapt postures. It uses a mix between motion blending and procedural animation which accelerates computation and increase model reactivity.

## 5 Results and performance analysis

We tested our algorithm on two environments. The first environment is the platform presented Fig. 2. It

| Scene | Floors | Polygons | Number of prisms | Number of 3D cells | Number of topological maps | Topological map size | Roadmap size |
|---|---|---|---|---|---|---|---|
| Platform | 1 | 7094 | 7038 | 24502 | 1 | 2406 cells | 439 nodes, 1054 paths |
| House | 3 | 120160 | 120664 | 1180834 | 3 | (1) 6871 cells<br>(2) 20222 cells<br>(3) 20046 cells | 2079 nodes, 5066 paths<br>4027 nodes, 9556 paths<br>4056 nodes, 9636 paths |

Table 1: Environment representation characteristics on tested environments.

| Scene | Floors | Polygons | Geometry preprocessing | Prismatic subdivision | Topology | Topological map | Roadmap | Total time |
|---|---|---|---|---|---|---|---|---|
| Platform | 1 | 7094 | 1.73 *s* | 12.22 *s* | 0.7 *s* | 1.34 *s* | 0.55 *s* | 16.54 *s* |
| House | 3 | 120160 | 118.84 *s* | 255.84 *s* | 63.78 *s* | (1) 33.08 *s*<br>(2) 110.52 *s*<br>(3) 94.32 *s* | 9.56 *s*<br>109.21 *s*<br>109.17 *s* | <br><br>904.32 *s* |

Table 2: Benchmark of environment computation on tested environments.

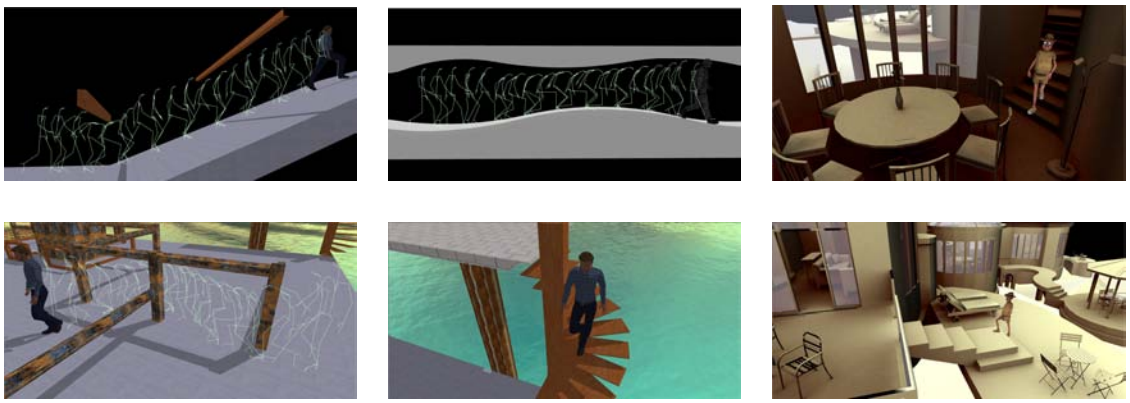

Figure 11: Motion adaptation to floor and ceiling constraints.

is compounded of 7952 triangles and contains several test cases: a regular staircase, an irregular staircase (with varying step length), a spiral staircase, a block maze, several beams and a tunnel with a curved floor surmounted by a curved ceiling. The second environment is a house with two floors and a tunnel, several staircases and several rooms with furniture (Cf. right images of Fig. 11). The geometry is compounded of 120160 triangles. Tables 1 and 2 respectively present the size of the data structures generated during environment analysis and the time taken to generate each representation (prismatic spatial subdivision, topology, topological maps and roadmaps). Computation time as well as representation complexity increase with the size of the input mesh and the number of floors. In the worst case, the number of generated prisms is a polynomial function of the number of overlapping triangles (two overlapping triangles can generate up to height prisms). This is a drawback of our approach. However, our preprocessing phase tends to reduce the complexity of the input geometry, and in most cases, complex geometries which do not influence navigation can be removed or simplified before environment processing.

Figure 11 presents some result concerning motion adaptation: climbing a slope while avoiding beams, walking on an curved floor under an curved ceiling, avoiding beams along a curved trajectory and climbing several types of staircases. Our reactive avoidance process, due to the use of the virtual ceiling, is endowed with anticipation capabilities. Humanoids can anticipate collision on uneven floor surmounted by an irregular ceiling without any specific computation such as presented fig. 11. Moreover, the footprint generation process is able to generate climbing motions even in very constrained stairs such as in the narrow spiral staircase for example. This demonstrates the generic aspect of our approach. In terms of performance, the average path planning time using an $A^*$ algorithm is about 1.2 *ms*. Our animation control processes are coupled with the approach proposed in [KM05, KMA05] to compute the virtual human animation. The humanoid animation only takes about 0.5 *ms* per time step with the following distribution:

- Motion blending: 0.041 *ms* per time step.
- Path optimization: 0.21 *ms* per footprint.
- Footprint correction: 0.05 *ms* per footprint.
- Floor height evaluation and ground adaptation (inverse kinematics on legs, root position correction): 0.012 *ms* per time step.
- Virtual ceiling computation and head position computation 0.236 *ms* per time step.
- IK on spine and head: 0.217 *ms* per time step.

Those results are compatible with the real time animation of several dozens of characters adapting their motion to ceiling constraints.

# 6 Discussion

We have presented TopoPlan, our topological planner. The proposed approach uses the 3D environment description to automatically generate an accurate spatial representation. It makes no assumption on the input geometry and proposes an exact $3D$ spatial subdivision (the prismatic subdivision) from which environment topology is extracted. Free space and navigable zones are thus exactly identified. The $2D$ representation provided by the topological map is compact and contains all required topological information such as obstacles, steps but also bottlenecks. It provides a suitable structure enabling to reduce some $3D$ problems to $2D$ problems and improves performance. Finally, the generated roadmap automatically captures the environment connectivity even in very constrained zones and is described by a small number of waypoints and paths. Based upon this representation, two motion adaptation algorithms have been proposed: a footprint generation process and posture adaptation to floor and ceiling constraints. Footprints are generated on demand by using a process compatible with reactive navigation algorithms. Posture adaptation is handled by a separate and independent process and can be used in several contexts and not only navigation. Those algorithms rapidly adapt to different humanoid morphologies by using high level parameters and do not require any complex collision checking. Finally, those algorithms have been designed to respect behavioral animation requirements: reactivity and performance.

Future works will focus on several aspects. Firstly, we will use TopoPlan properties to provide a new reactive navigation process taking into account constraints imposed by the environment. Secondly, we will focus on the generalization of our approach on posture adaptation within dynamic environments. Finally, we will explore the link between environment information and

topology determination in order to feed high level decision processes with abstract topological information automatically extracted from an informed environment geometry.

# References


[ACF01] ARIKAN O., CHENNEY S., FORSYTH D. A.: Efficient multi-agent path planning. In *Computer Animation and Simulation '01* (2001).

[BLA02] BAYAZIT O. B., LIEN J.-M., AMATO N. M.: Roadmap-based flocking for complex environments. In *Pacific Conference on Computer Graphics and Applications* (2002), pp. 104–113.

[BT98] BANDI S., THALMANN D.: Space discretization for efficient human navigation. *Computer Graphics Forum 17*, 3 (1998), 195–206.

[BY98] BOISSONNAT J. D., YVINEC M.: *Algorithmic Geometry*. Cambridge University Press, 1998.

[CLS03] 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), 182–203.

[HIKL*99] 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), 277–286.

[KBT03] KALLMANN M., BIERI H., THALMANN D.: Fully dynamic constrained delaunay triangulations. *Geometric Modelling for Scientific Visualization* (2003).

[KKL96] KAVRAKI L., KOLOUNTZAKIS M., LATOMBE J.: Analysis of probabilistic roadmaps for path planning. In *IEEE Int. Conf. on Robotics and Automation* (1996).

[KL00] 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).

[KM05] KULPA R., MULTON F.: Fast inverse kinematics and kinetics solver for human-like figures. In *IEEE-RAS Int. Conf. on Humanoid Robots* (2005).

[KMA05] KULPA R., MULTON F., ARNALDI B.: Morphology-independent representation of motions for interactive human-like animation. *Computer Graphics Forum 24*, 3 (2005), 343–351.

[KSLO94] KAVRAKI L., SVESTKA P., LATOMBE J.-C., OVERMARS M.: *Probabilistic roadmaps for path planning in high-dimensional conc̆guration spaces*. Tech. rep., 1994.

[Kuf98] KUFFNER J. J.: Goal-directed navigation for animated characters using real-time path planning and control. *Lecture Notes in Computer Science 1537* (1998), 171–179.

[Lat91] LATOMBE J. C.: *Robot Motion Planning*. Boston: Kluwer Academic Publishers, Boston, 1991.

[LaV06] LAVALLE S. M.: *Planning Algorithms*. Cambridge University Press, 2006.

[LCH03] LI T.-Y., CHEN P.-F., HUANG P.-Z.: Motion planning for humanoid walking in a layered environment. In *IEEE Int. Conf. on Robotics and Automation* (2003).

[LCL05] LEE K. H., CHOI M. G., LEE J.: Motion patches: buildings blocks for virtual environments annotated with motion data. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Sketches* (2005), ACM.

[LD04] LAMARCHE F., DONIKIAN S.: Crowd of virtual humans: a new approach for real time navigation in complex and structured environments. *Computer Graphics Forum 23*, 3 (2004).

[LH04] LI T.-Y., HUANG P.-Z.: Planning humanoid motions with striding ability in a virtual environment. In *IEEE Int. Conf. on Robotics and Automation* (2004).

[LK05] LAU M., KUFFNER J. J.: Behavior planning for character animation. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (2005).

[LK06] LAU M., KUFFNER J. J.: Precomputed search trees: Planning for interactive goal-driven animation. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (2006).

[PLS03] PETTRÉ J., LAUMOND J.-P., SIMÉON T.: 3d collision avoidance for digital actors locomotion. In *Intelligent Robots and Systems* (2003).

[PLT06] PETTRE J., LAUMOND J. P., THALMANN D.: A navigation graph for real-time crowd animation on multilayered and uneven terrain. In *V-Crowds* (2006).

[SAC*07] SUD A., ANDERSEN E., CURTIS S., LIN M., MANOCHA D.: Real-time path planning for virtual agents in dynamic environments. In *IEEE Virtual Reality conference* (2007).

[SGA*07] SUD A., GAYLE R., ANDERSEN E., GUY S., LIN M., MANOCHA D.: Real-time navigation of independent agents using adaptive roadmaps. In *ACM symposium on Virtual reality software and technology* (2007).

[SGLM03] SALOMON B., GARBER M., LIN M. C., MANOCHA D.: Interactive navigation in complex environments using path planning. In *symposium on Interactive 3D graphics* (2003).

[SH07] SAFONOVA A., HODGINS J. K.: Construction and optimal search of interpolated motions graphs. *ACM Transactions on Graphics 26*, 3 (2007).

[SLCP05] SAHA M., LATOMBE J.-C., CHANG Y.-C., PRINZ F.: Finding narrow passages with probabilistic roadmaps: The small-step retraction method. *Autonomous Robots 19* (2005), 301–319.

[ST06] SHAO W., TERZOPOULOS D.: Environmental modeling for autonomous virtual pedestrians. *SAE 2005 Transactions Journal of Passenger Cars: Electronic and Electrical Systems 114*, 7 (2006), 735–742.

[SYN01] SHILLER Z., YAMANE K., NAKAMURA Y.: Planning motion patterns of human figures using a multi-layered grid and the dynamics filter. In *IEEE Int. Conf. on Robotics and Automation* (2001).

[YKH04] YAMANE K., KUFFNER J., HODGINS J. K.: Synthesizing animations of human manipulation tasks. In *Proceedings of SIGGRAPH 2004* (2004).