Fast Behavioral Locomotion with Layered Navigation Meshes

Alain Juarez-Perez
University of California, Merced
Merced, California
ajuarez-perez@ucmerced.edu

Marcelo Kallmann
University of California, Merced
Merced, California
mkallmann@ucmerced.edu

ABSTRACT

We address the problem of efficient navigation planning for virtual characters with multiple locomotion behaviors. One main difficulty in this problem is the need to rely on expensive collision detection computations which typically lead to methods unsuitable for real-time applications. At the same time, the alternative of using reactive path following methods lead to characters less capable of addressing cluttered environments. In this paper we propose a locomotion planning approach that incorporates the behavioral capabilities of the character during the path planning stage without the need to rely on expensive collision detection computations. Our approach is based on computing paths and clearance tests in layered navigation meshes, where layers are positioned at different body heights in order to reduce 3D collision checking to fast 2D clearance determination per layer. The proposed approach significantly improves the overall planning time making our method suitable for real-time applications while successfully addressing tradeoffs between path length and selection of narrow passages according to preferred locomotion behaviors.

CCS CONCEPTS

• **Computing methodologies** → *Motion path planning*; *Motion processing*;

KEYWORDS

Path planning, motion planning, path following, navigation.

ACM Reference Format:
Alain Juarez-Perez and Marcelo Kallmann. 2018. Fast Behavioral Locomotion with Layered Navigation Meshes . In *I3D '18: I3D '18: Symposium on Interactive 3D Graphics and Games, May 4–6, 2018, Montreal, QC, Canada.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3190834.3190841

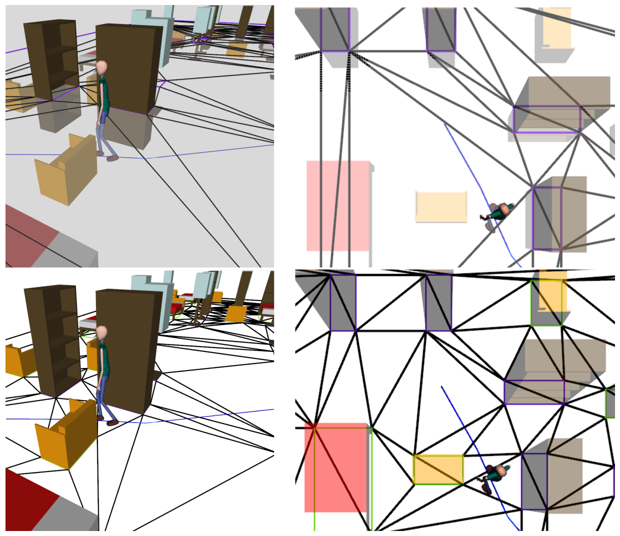

Figure 1: Two layers are used in our method. The bottom layer (bottom images) represents all obstacles while the top layer (top images) only represents the tall obstacles which are mostly important to detect torso collisions.

1 INTRODUCTION

One main difficulty faced by full-body motion planners is the need to rely on expensive collision detection computations in order to validate the execution of locomotion behaviors in narrow passages of a given environment. The character and the environment commonly need to be represented with significant detail and the involved collision checks often make these planning methods unsuitable for real-time applications.

The typical alternative for character navigation in virtual environments is to decouple the problem in two stages: first, a path is planned in the free portion of the environment, and second, steering behaviors are used to follow the path while reactively selecting locomotion behaviors according to the encountered constraints.

Unfortunately, when environments are cluttered with obstacles reactive approaches are less capable of addressing narrow areas and they can easily lead to difficult maneuvers in narrow passages while much more comfortable paths just slightly longer may be available. In decoupled approaches the path planning stage does not take into consideration the behavioral choices of the character.

We follow the approach of performing successive path planning queries in a navigation mesh in order to plan paths incorporating the behavioral capabilities of the character, as introduced in our recent previous work [Juarez-Perez and Kallmann 2016a]. The approach however requires determining the feasibility of performing different locomotion behaviors in the environment, what is usually performed with expensive 3D collision detection computations. In this paper we improve the approach and we replace 3D collision checking by layered 2D clearance tests significantly improving computation times. See Figure 1.

Our approach significantly improves the overall planning time making our method suitable for real-time applications and at the same time achieving paths addressing the trade-off between path length and using preferred locomotion behaviors instead of handling narrow passages with high-cost behaviors. Our benchmarks demonstrate a 20-fold computation time speed-up in comparison to relying on full 3D collision checking.

2 RELATED WORK

Different approaches are possible for full-body locomotion. For instance, physics-based methods are capable of achieving realistic motion adaptation to different events in the environment [Zimmermann et al. 2015] and to facilitate the coordination between concurrent upper-body motions [2012]; however, requiring expensive physics simulators and not addressing path planning taking into account different behaviors.

Data-based approaches are also very popular for locomotion planning. The first methods designed to re-use data from a motion capture database [Arikan and Forsyth 2002; Kovar et al. 2002; Lee et al. 2002] gave rise to *motion graphs*, which can be used to support concatenation planning of the motion clips in the graph in order to achieve navigation in a given environment. Many variants were developed, as for instance to improve connectivity [Mahmudi and Kallmann 2013; Zhao et al. 2009; Zhao and Safonova 2008] and to structure motions as parameterized behaviors, leading to pre-computed navigation behaviors [2006] able to support motion search at interactive performances. Pre-computed motion maps have also been introduced [Mahmudi and Kallmann 2012] to achieve interactive motion search and synthesis in an unstructured motion capture database. While these methods represent powerful approaches for achieving high-quality animations, they require pre-computation procedures that are time-intensive and memory-consuming, and still involve significant computation at running time.

A number of methods are based on other types of motion planning techniques. For example one approach is to deform animation data to cope with constraints while searching for paths using a probabilistic roadmap [Choi et al. 2003]. A more recent approach uses specialized motion deformation techniques that are used during navigation planning [Choi et al. 2011]. A large number of additional full-body motion planning techniques have been proposed from researchers from both the robotics and computer animation areas. While these methods are able to achieve impressive results, their focus is on the motion synthesis generation in complex environments and not on achieving a real-time solution for the underlying navigation trajectory selection. Our method focuses on this problem and on how to significantly improve the overall planning time.

The motion planning methodology proposed in this paper can be applied to any given locomotion controller able to follow paths with different navigation behaviors. Our motion controller is data-based and has the advantage of relying on only one deformable animation clip per behavior. There are however a number of possible data-based approaches relying on a larger number of motion clips [Heck and Gleicher 2007; Lee et al. 2010] which could be employed and which have the potential to generate better-quality results. Motion controllers can also be based on learning methods, such as reinforcement learning. For instance, Treuille et al. [2007] and Levine et al. [2011] have employed learning for achieving powerful real-time character controllers. While these methods are time-consuming during learning and controllers have to remain in a low dimensional space, these methods could also be controlled by our proposed multi-behavior path planner.

With respect to path planning techniques on navigation meshes, most of the methods have been developed independently from full-body locomotion behaviors, and only considering clearance on the 2D floor plan of a given environment [Kallmann and Kapadia 2016a,b]. In contrast we plan paths incorporating the behavioral capabilities of the character, following the method introduced in our previous work [Juarez-Perez and Kallmann 2016a]. The approach requires determining the feasibility of performing different locomotion behaviors in the environment and, instead of performing expensive 3D collision detection computations, in this paper we introduce the concept of using clearance tests in layered navigation meshes, significantly improving computation times, up to 20 times according to our benchmarks. Our layers are represented with an efficient navigation mesh called a Local Clearance Triangulation (LCT) [Kallmann 2014], which allows efficient path queries, clearance tests, and dynamic mesh changes during the process of searching for a path suitable for multi-behavior execution. While popular solutions used in many applications will typically build a mesh specifically for only one clearance value [Mononen 2018], a growing number of methods are being developed supporting arbitrary clearance queries [van Toll et al. 2016].

In summary, while powerful behavioral planning approaches have been proposed in the past for planning multi-behavior navigation in cluttered environments, they quickly become too expensive for interactive applications. On the other hand efficient methods based on path planning with navigation meshes have not been extended to take into account constraints or trade-offs emerging from multi-behavioral locomotion. The method proposed in this paper addresses these issues in an efficient way with the introduction of layered clearance tests, which replace full collision checks and achieve significant computation time improvements.

3 METHOD

The first step of our method is to determine the clearances required by each locomotion behavior to be considered.

3.1 Locomotion Behaviors

Our locomotion controller relies on a set of deformable motion capture clips and is implemented based on the techniques presented in [Juarez-Perez et al. 2014; Juarez-Perez and Kallmann 2016b]. The controller requires annotated transition points between data-based behaviors and as well the general direction of motion. Motion deformations are performed with small motion modifications applied incrementally at each frame of the motions, allowing us to precisely parameterize behaviors for achieving smooth path following.

In our current implementation we employ three locomotion behaviors: regular frontal walking, arm-constrained frontal walking and lateral walking. The motion capture set contains annotated clips of each of the behaviors as well as the necessary transition clips between each behavior. The arm-constrained frontal walking behavior is implemented by modifying the trajectory of the arms during the regular frontal walking so that the arms remain close to the body making the character require less clearance.

The controller can be instructed to switch to a different behavior at any time and the necessary transition will be automatically applied. Motion blending is applied to smooth out any possible misalignment introduced by the deformations. At each behavior concatenation point it is possible to start following a new path starting at the current position and orientation and/or to switch to a different behavior. The goal of the behavioral path planner (described in Section 3.3) is to generate a multi-behavior path that can be executed by the locomotion controller.

Since in the proposed approach collision detection is reduced to layered clearance tests, the locomotion behaviors taken into account are analyzed in order to determine their clearance requirements. For each frame of each motion we analyze the required clearances at several layers parallel to the ground plane. Clearances are computed as the minimum diameter completely enclosing the character in the direction orthogonal to the motion direction. A few representative layers are then selected for use during the path planning stage.

For the considered behaviors we have noticed that two layers are enough to capture the most important clearance requirements. One layer represents the clearance requirements of the legs and the other takes into account the movement of the arms. These layers are referred to as the legs and torso layers. Their respective clearances are illustrated in Figure 2.

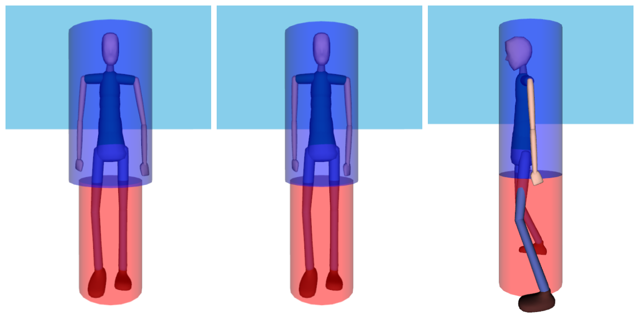

Figure 2: Left: the frontal walking behavior requires the torso clearance larger than the bottom clearance. Center: with arms constrained frontal walking requires only slightly more torso clearance than legs clearance. Right: lateral walking requires the least clearance and the torso and legs clearances are the same because arm movement occurs only along the motion direction.

We organize the available $n = 3$ locomotion behaviors as a set $\{\mathcal{B}_1, \ldots, \mathcal{B}_n\}$. Each behavior is associated with the values of the clearances computed in the considered layers. In our current case we have the legs clearance c_i^1 and the torso clearance c_i^2, $i \in \{1, \ldots, n\}$.

The behaviors are organized according to the preference in using them. Our last behavior $\mathcal{B}_n$ is the preferred locomotion behavior and in our case it represents regular frontal walking. Behavior $\mathcal{B}_1$ is defined as the least preferred behavior, which is an uncomfortable behavior to execute, but the most appropriate navigation behavior for narrow passages. In our case $\mathcal{B}_1$ performs lateral walking (or side-stepping). Any behavior in-between will represent a trade-off between capabilities of navigation and preference of execution. In our current setup, we only have one intermediate behavior, $\mathcal{B}_2$, which is the regular frontal walking with the constrained arm motion. A cost function will be later associated to each behavior in order to quantify the "preference" of using a behavior.

It is possible to observe that the required leg clearance is the same in all our behaviors while the torso clearance is higher in the preferred behaviors, such that $c_i^1 = c_j^1, \forall i, j$, and $c_i^2 < c_j^2$ when $i < j$, $i, j \in \{1, \ldots, n\}$. Our set of behaviors is therefore also ordered by increasing torso clearance requirements.

3.2 Layers

For maximum efficiency our path planner only relies on layer queries. The first layer (Figure 1-bottom) is a navigation mesh where its 2D obstacles are the ground projections of all of the 3D obstacles. This layer captures all the passages of the environment and will be used to perform path queries.

The second layer (Figure 1-top) is used to validate, label and select alternative behaviors that can be used to execute paths with different clearances. This layer only contains as 2D obstacles the polygonal sections obtained by the intersection of the layer plane with the 3D obstacles. It therefore only represents the obstacles that can generate a conflict with the torso of our character.

Additional layers can be incorporated as needed. The overall approach is to first compute low-clearance paths on the lower layer capturing all obstacles and then to check for additional clearance on the other layers.

3.3 Behavioral Layered Path Planner

Our planner computes a free path from the current position of the character to a given goal position, and returns it as a sequence of concatenated sub-paths, each to be executed with a specific locomotion behavior. A null path is returned if no path exists. The overall solution is therefore a sequence of sub-path sections $\Pi = \{\pi_1, \ldots, \pi_k\}$, where each section π_i satisfies the needed clearance value for its specific behavior.

The possibility of using different behaviors leads to several possible solutions. A cost value can be associated to each behavior such that the overall cost of a solution can be defined as:

$$\textsc{Cost}(\Pi) = \sum_k ||\pi_k|| \cdot cost_k\,(\mathcal{B}_k)\,,$$

where $||\pi_k||$ denotes the length of the k^{th} sub-path, and $cost_k(\mathcal{B}_k)$ returns the cost of locomotion per unit length using behavior $\mathcal{B}_k$. While the goal of our planner is to find a solution that minimizes this overall cost function, our current approach is heuristic-based and does not attempt to guarantee a global optimum.

We rely on the local clearance triangulation (LCT) navigation mesh [Kallmann 2014], which supports computation of paths with arbitrary clearance, clearance checks at any position, and dynamic insertion and removal of degenerate obstacles defined as single points. This last feature is used to add constraints that block passages in the navigation mesh, forcing the path planner to search for alternative paths with new requirements in search for lower-cost solutions.

Algorithm 1 summarizes the main steps of the layered path planner. It starts by finding an initial annotated path in the bottom layer using the smallest clearance c_1^1, which is obtained with a call to function FINDPATH(c_1^1, s, g). Without additional tests this path can only be safely executed using our least preferred behavior $\mathcal{B}_1$, which is the behavior of highest cost. c_1^1 is the only clearance that can guarantee that if a path is found, a solution with our available behaviors exists. The algorithm will then seek to lower the overall cost of this initial solution.

Function CONFLICTSECTIONS() returns all points $\{p_{obs}\}$, spaced by the minimum behavior concatenation length along the current annotated path Π_{cur}, where the path only has enough clearance for $\mathcal{B}_1$ to be executed. These points represent passages that are too narrow for other behaviors to be executed. In order to evaluate alternative solutions, a point subset $\{\hat{p}_{obs}\} \subset \{p_{obs}\}$ is selected to be inserted as point-obstacles, forcing the planner to search for a different path, which may have lower overall cost. While considering all sub-sets would lead to the evaluation of all alternative paths, too many combinations would be generated and we limit our search to every single point instead of every possible subset.

Overall, Algorithm 1 greedily explores the possible behaviors that can navigate the path, using CONFLICT SECTIONS() to detect the narrowest path sections that will likely generate collisions if the side-stepping behavior is not used. It then evaluates modifying the path in order to be able to use preferred behaviors and to lower the overall cost of the current solution. To force new paths to be evaluated, Algorithm 2 adds 2D point-obstacles to the bottom layer navigation mesh in order to block narrow passages and force new paths to be found and evaluated.

If using the least desired behavior cannot be prevented by path modification, the algorithm will eventually find a path that exceeds the cost of the current one, at which point it will finish its execution and return the current solution.

When Algorithm 1 succesfully terminates it returns a path annotated with behaviors to be executed. The annotation procedure is called for every path evaluated and it is described in Algorithm 2.

Algorithm 2 relies on TESTCLEARANCE($\Pi, p, \mathcal{B}$) which verifies the possibility from point p onwards, along path Π, to employ behavior $\mathcal{B}$. The function returns the first point that does not allow using $\mathcal{B}$ due clearance requirements. This procedure relies on clearance tests performed on the upper layers as required by the requested behavior. The upper layer doesn't contain short obstacles because they won't cause collisions with the torso or arms, therefore the upper layer has more clearance in certain passages than the bottom layer. Clearance tests are efficiently performed because the navigation mesh provides an exact triangular decomposition of the free space in the layer. This allows the quick verification if a desired clearance value around a point in the path remains covered by free triangles, which is executed by only locally checking the triangles around the query point.

An example of a solution path obtained is shown in Figure 3. While the available leg clearance is the same inside the corridor, due to the difference in height of the obstacles, different preferred behaviors with larger required clearance are selected along the path. The solution path in this example has the minimum possible cost.

Algorithm 1 - Behavioral Layered Path Planner

Input: initial position s, goal position g and the number of available behaviors n (3 in our current work.)
Output: path from s to g with annotated behavior, or null path if a feasible path does not exist.

```
1:  procedure BHPATH($s, g, n$)
2:      $cost_{old} = \infty$;
3:      $\Pi_{cur}$ = FINDPATH($c_1^1; s, g$);
4:      $\Pi_{cur}$ = ANNOTATEPATH($\Pi_{cur}, s, g$);
5:      $cost_{min}$ = COST ($\Pi_{curr}$)
6:      while ( $cost_{min} < cost_{old}$ ) do
7:          $cost_{old} = cost_{min}$;
8:          $\{p_{obs}\}$ = CONFLICTSECTIONS($\Pi_{cur}$);
9:          $\Pi_{min} = \emptyset$;
10:         for each $\{\hat{p}_{obs}\} \subset \{p_{obs}\}$ do
11:             Add $\{\hat{p}_{obs}\}$ as a point obstacle in first layer;
12:             $\Pi_{new}$ = FINDPATH($c_1^1; s, g$);
13:             $\Pi_{new}$ = ANNOTATEPATH($\Pi_{new}, s, g$);
14:             if (COST($\Pi_{new}$) < $cost_{min}$) then
15:                 $cost_{min}$ = COST($\Pi_{new}$);
16:                 $\Pi_{min} = \Pi_{new}$;
17:             Remove $\{\hat{p}_{obs}\}$ from the environment;
18:         if ($\Pi_{min} \neq \emptyset$) then
19:             $\Pi_{cur} = \Pi_{min}$;
20:     return $\Pi_{curr}$;
```

Algorithm 2 - Path Annotation

Input: Path Π, traversing the environment from s to g
Output: Annotated path.

```
1:  procedure ANNOTATEPATH($\Pi, s, g$)
2:      $p$ = TESTCLEARANCE($\Pi, s, \mathcal{B}_n$);
3:      $\Pi_{cur}$ = FINDPATH($c_n; s, p$);
4:      while ( $p \neq g$ ) do
5:          for ($i = n$ to 1 ) do
6:              $q$ = TESTCLEARANCE($\Pi, p, \mathcal{B}_i$);
7:              if ( $p \neq q$) then
8:                  Append FINDPATH($c_i^2; p, q$) to $\Pi_{curr}$;
9:                  break; //exit for loop
10:         $p = q$;
11:     return $\Pi_{curr}$;
```

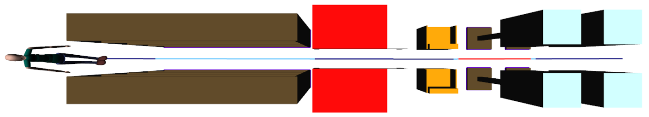
Figure 3: Example solution path found by our method. The blue path sections can be executed with $\mathcal{B}_3$, the cyan sections with $\mathcal{B}_2$, and the red sections with $\mathcal{B}_1$.

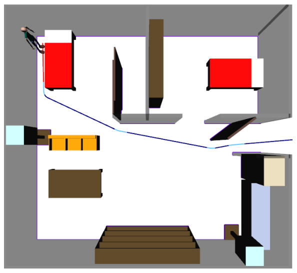

Figure 4: Apartment scenario.

After successful completion the overall algorithm returns a path

$$\Pi = \{\pi_1(c_{j_1}; \boldsymbol{a}_1, \boldsymbol{b}_1), \ldots, \pi_n(c_{j_n}; \boldsymbol{a}_n, \boldsymbol{b}_n)\},$$

where $\boldsymbol{b}_i = \boldsymbol{a}_{i+1}$, $i \in \{1, ..., n-1\}$. The path sequence implicitly defines a solution path with behavior transition in the following way: if $j_i = 3$ regular walking is used for that section; if $j_i = 2$, constrained walking is used; otherwise lateral walking is used.

In summary, the overall procedure starts by finding the minimum clearance path, which can be executed with lateral stepping, however since this behavior is slow and often unnecessary, path section replacements accommodating preferred behaviors are performed whenever the overall path cost can be lowered.

4 RESULTS AND DISCUSSION

We have evaluated the performance of our method and we demonstrate results obtained in three different scenarios.

4.1 Scenarios

In the first scenario (Figure 4) the character has to find a path for traversing an apartment cluttered with furniture. Multiple behaviors are required, for instance, the door is in a position that makes it impossible to enter the room using the preferred regular walking behavior.

The second scenario, illustrated in Figure 5, was created to trigger all the behaviors. It consists of a narrow corridor with specific areas designed to be traversed only by one of the available behaviors. The first area of the corridor requires the arms to be constrained. The second area has less clearance but because the obstacles do not collide with the arms the system allows a switch to the regular front walking behavior. The final section has the minimum acceptable clearance and lateral side-stepping is required.

The third scenario is a randomized environment with obstacles of varied dimensions (Figure 6). We can control the density of the obstacles in the scene, going from easy to traverse with the normal behavior to highly cluttered and often requiring non-preferred behaviors. Every time the user selects a reachable point in the environment, the system computes and concatenates a new annotated path to be executed in order to reach the new point. Smooth and responsive locomotion control is achieved and performed in real-time. We use this environment in the evaluations described next.

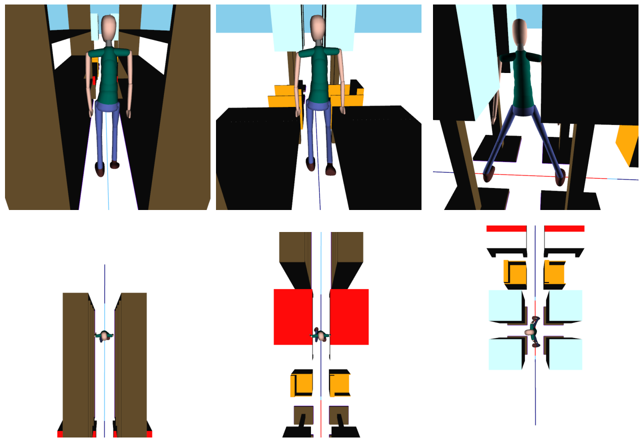

Figure 5: Narrow corridor behavior selection. From left to right: arm avoidance is employed in the first section, regular walking is correctly employed in the middle section given the high clearance at the torso layer, and lateral side-stepping is required in the last, narrowest, section.

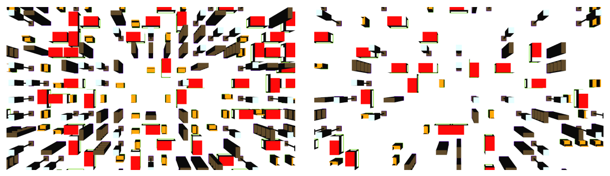

Figure 6: Dense (left) and Normal (right) environments.

Table 1: Average times in milliseconds to compute paths with layered queries versus full 3D collision checking.

	Dense environment		Normal environment	
	Layered	3D Collision	Layered	3D Collision
20 Steps	9.70	189.40	6.14	110.65
10 Steps	10.62	105.26	6.48	61.15
5 Steps	9.67	28.21	6.16	22.23

4.2 Performance Evaluation

We evaluate the performance of our planner on two different randomized environments, one with a *normal* density of obstacles and the other with a denser configuration, as illustrated in Figure 6.

In order to test our environments, we have generated 100 different goals at the same linear distance from the starting point of the path queries. The length of the planned paths was equivalent to 5, 10 and 20 character steps using the normal behavior. We then ran our layered locomotion planner and compared the results against using full 3D collision detection with the environment geometry as performed previously [Juarez-Perez and Kallmann 2016a]. The results are presented in Table 1 and Figure 7.

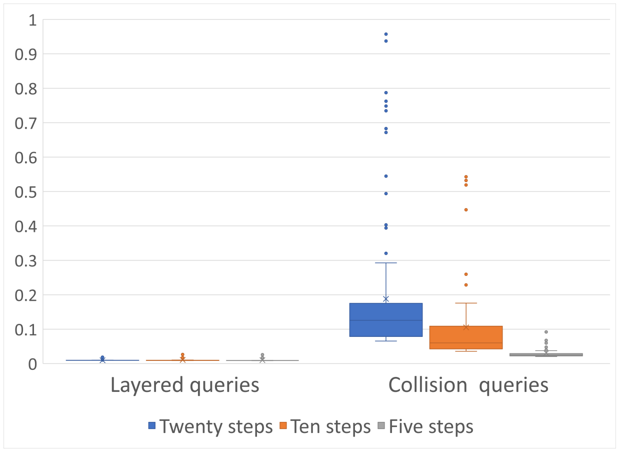

Figure 7: Average times, deviation and outliers when computing paths of different lengths on a dense environment.

4.3 Discussion

The proposed layered approach clearly overcomes the collision detection bottleneck, significantly speeding up computation times and allowing us to plan considerably long paths in real-time.

Our system can be easily adapted to operate with a larger set of behaviors and correspondent layers. While currently our approach of projecting all obstacles to the bottom layer cannot capture more complex movements such as crawling under a table, it is certainly possible to design specific uses of layers in order to capture such behaviors. Currently our layer mechanism is used only for capturing clearance along horizontal planes, which is appropriate for many locomotion behaviors. Applying layers in generic orientations for other types of movement planning is an interesting future work direction.

It is also possible to explore improvements to the overall approach of our method. Our current greedy approach based on blocking passages can be extended to evaluate all relevant passages and the underlying graph search can also be extended to generate and evaluate all relevant paths [Eppstein 1999] without requiring to block passages and to perform multiple independent queries, possibly leading to faster final times. However a combinatorial approach may generate too many candidates given that the global optimum of our problem lies on the Euclidean plane and not on the used adjacency graph representing the free space.

5 CONCLUSION

We have presented a path planning method that incorporates the automatic selection of behaviors with different costs. Our method computes paths which can be followed by locomotion behaviors with different clearance requirements, achieving fast multi-behavior navigation in cluttered environments. Our method introduces the concept of applying layered clearance tests instead of full 3D collision checking, providing a significant speed-up in computation time. The computed solutions adapt to dense environments with narrow passages and our results demonstrate real-time performances with solutions that represent natural behavioral choices.

REFERENCES

Okan Arikan and D. A. Forsyth. 2002. Interactive Motion Generation from Examples. *ACM Trans. Graph.* 21, 3 (July 2002), 483–490.

Yunfei Bai, Kristin Siu, and C. Karen Liu. 2012. Synthesis of Concurrent Object Manipulation Tasks. *ACM Transactions on Graphics* 31, 6, Article 156 (Nov. 2012).

Myung Geol Choi, Manmyung Kim, Kyung Lyul Hyun, and Jehee Lee. 2011. Deformable Motion: Squeezing into Cluttered Environments. *Computer Graphics Forum* (2011). https://doi.org/10.1111/j.1467-8659.2011.01889.x

Min Gyu Choi, Jehee Lee, and Sung Yong Shin. 2003. Planning Biped Locomotion Using Motion Capture Data and Probabilistic Roadmaps. *ACM Transactions on Graphics* 22, 2 (April 2003), 182–203. https://doi.org/10.1145/636886.636889

David Eppstein. 1999. Finding the K Shortest Paths. *SIAM J. Comput.* 28, 2 (Feb. 1999), 652–673. https://doi.org/10.1137/S0097539795290477

Rachel Heck and Michael Gleicher. 2007. Parametric motion graphs. In *Proceedings of the symposium on Interactive 3D graphics and games (I3D)*. ACM Press, New York, NY, USA, 129–136.

Alain Juarez-Perez, Andrew Feng, Marcelo Kallmann, and Ari Shapiro. 2014. Deformation, Parameterization and Analysis of a Single Locomotion Cycle. In *Proceedings of the Seventh International Conference on Motion in Games (MIG '14)*. ACM, New York, NY, USA, 182–182. https://doi.org/10.1145/2668064.2677082

Alain Juarez-Perez and Marcelo Kallmann. 2016a. Full-Body Behavioral Path Planning in Cluttered Environments. In *Proceedings of the ACM SIGGRAPH Conference on Motion in Games (MIG)*. ACM.

Alain Juarez-Perez and Marcelo Kallmann. 2016b. Modeling Data-Based Mobility Controllers with Known Coverage and Quality Properties. In *Digital Human Modeling*.

Marcelo Kallmann. 2014. Dynamic and Robust Local Clearance Triangulations. *ACM Transactions on Graphics (TOG)* 33, 5 (2014).

Marcelo Kallmann and Mubbasir Kapadia. 2016a. *Geometric and Discrete Path Planning for Interactive Virtual Worlds.* Morgan and Claypool Publishers.

Marcelo Kallmann and Mubbasir Kapadia. 2016b. Geometric and Discrete Path Planning for Interactive Virtual Worlds. In *SIGGRAPH Course Notes*.

Lucas Kovar, Michael Gleicher, and Frédéric Pighin. 2002. Motion graphs. *ACM Transactions on Graphics* 21, 3 (jul 2002), 473–482.

Manfred Lau and James J. Kuffner. 2006. Precomputed search trees: planning for interactive goal-driven animation. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA)*. 299–308.

Jehee Lee, Jinxiang Chai, Paul Reitsma, Jessica K Hodgins, and Nancy Pollard. 2002. Interactive Control of Avatars Animated with Human Motion Data. *Proceedings of SIGGRAPH* 21, 3 (July 2002), 491–500.

Yongjoon Lee, Kevin Wampler, Gilbert Bernstein, Jovan Popović, and Zoran Popović. 2010. Motion Fields for Interactive Character Locomotion. *ACM Transactions on Graphics* 29, 6, Article 138 (Dec. 2010), 138:1–138:8 pages.

Sergey Levine, Yongjoon Lee, Vladlen Koltun, and Zoran Popović. 2011. Space-time planning with parameterized locomotion controllers. *ACM Trans. Graph.* 30, 3 (May 2011).

Mentar Mahmudi and Marcelo Kallmann. 2012. Precomputed motion maps for unstructured motion capture. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '12)*. 127–136.

Mentar Mahmudi and Marcelo Kallmann. 2013. Analyzing Locomotion Synthesis with Feature-Based Motion Graphs. *IEEE Transcations on Visualization and Computer Graphics (TVCG)* 19, 5 (2013), 774–786.

M. Mononen. 2018. Recast Navigation. (2018). https://github.com/memononen/recastnavigation/

A. Treuille, Y. Lee, and Z. Popović. 2007. Near-optimal Character Animation with Continuous Control. In *Proceedings of ACM SIGGRAPH*. ACM Press.

Wouter van Toll, Roy Triesscheijn, Marcelo Kallmann, Ramon Oliva, Nuria Pelechano, Julien Pettré, and Roland Geraerts. 2016. A Comparative Study of Navigation Meshes. In *Proceedings of the ACM SIGGRAPH Conference on Motion in Games (MIG)*.

Liming Zhao, Aline Normoyle, Sanjeev Khanna, and Alla Safonova. 2009. Automatic Construction of a Minimum Size Motion Graph. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*.

Liming Zhao and Alla Safonova. 2008. Achieving Good Connectivity in Motion Graphs. In *Proceedings of the 2008 ACM/Eurographics Symposium on Computer Animation (SCA)*. 127–136.

Daniel Zimmermann, Stelian Coros, Yuting Ye, Robert W. Sumner, and Markus Gross. 2015. Hierarchical Planning and Control for Complex Motor Tasks. In *Proceedings of the 14th ACM SIGGRAPH / Eurographics Symposium on Computer Animation (SCA '15)*. ACM, New York, NY, USA, 73–81.