PRECISION: Precomputing Environment Semantics for Contact-Rich Character Animation

Mubbasir Kapadia[1,*], Xu Xianghao[2], Maurizio Nitti[3], Marcelo Kallmann[4], Stelian Coros[5], Robert W. Sumner[2,3], Markus Gross[2,3]
[1]Rutgers University, [2]ETH Zurich, [3]Disney Research, [4]UC Merced, [5]Carnegie Mellon University

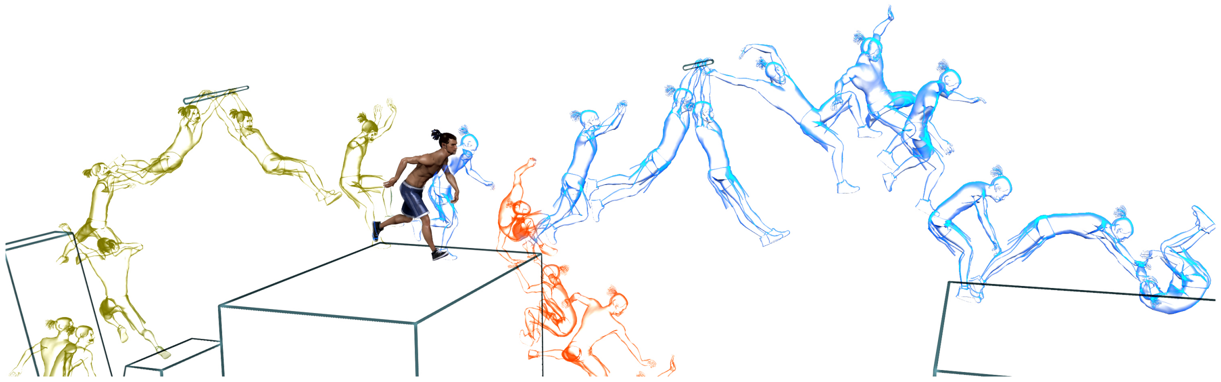


Figure 1: *An autonomous virtual character harnessing a wide array of contact-rich movement capabilities while navigating in a complex virtual environment.*

Abstract

The widespread availability of high-quality motion capture data and the maturity of solutions to animate virtual characters has paved the way for the next generation of interactive virtual worlds exhibiting intricate interactions between characters and the environments they inhabit. However, current motion synthesis techniques have not been designed to scale with complex environments and contact-rich motions, requiring environment designers to manually embed motion semantics in the environment geometry in order to address online motion synthesis. This paper presents an automated approach for analyzing both motions and environments in order to represent the different ways in which an environment can afford a character to move. We extract the salient features that characterize the contact-rich motion repertoire of a character and detect valid transitions in the environment where each of these motions may be possible, along with additional semantics that inform which surfaces of the environment the character may use for support during the motion. The precomputed motion semantics can be easily integrated into standard navigation and animation pipelines in order to greatly enhance the motion capabilities of virtual characters. The computational efficiency of our approach enables two additional applications. Environment designers can interactively design new environments and get instant feedback on how characters may potentially interact, which can be used for iterative modeling and refinement. End users can dynamically edit virtual worlds and characters will automatically accommodate the changes in the environment in their movement strategies.

Keywords: character animation, environment semantics, contact-rich motion

*e-mail:mubbasir.kapadia@rutgers.edu

I3D '16,, February 27-28, 2016, Redmond, WA,
ISBN: 978-1-4503-4043-4/16/03
DOI: http://dx.doi.org/10.1145/2856400.2856404

1 Introduction

Humanoid character animation is a well-studied topic with many solutions available to animate fully articulated virtual humans. Animated characters that rely on motion capture [Kovar et al. 2002; Arikan and Forsyth 2002] or artist-created content move in natural and pleasing ways, and can be used to create complex data-driven controllers [Lau and Kuffner 2005; Levine et al. 2011] for animating interactive, responsive virtual characters.

The widespread availability of high-quality motions, and the maturity of character animation systems have enabled the design of large, complex virtual worlds [Ubisoft 2014] which require complex interactions between the environment geometry and the virtual characters that inhabit it. Current automated approaches for motion synthesis [Lau and Kuffner 2005; Lau and Kuffner 2006] cannot be easily extended to address the increase in environment and motion complexity, while being suitable for interactive applications such as games. To mitigate this difficulty, environment designers are burdened with the task of manually annotating motion semantics in the environment, which helps inform character controllers which motion(s) a character is able to perform and how, thus pruning the search space for motion synthesis. This is very tedious, especially for motions that involve intricate contacts between the character and the environment.

This paper presents an automated approach for computing the semantics of how a character can interact with the environment geometry. Given a database of motion clips representing arbitrarily complex contact-rich behaviors of a virtual character, we first analyze the motions in order to define its *motion signature*, which characterizes the contacts between the character and environment surfaces, spatial relationships between contacts, and the collision representation of the character.

For any 3D environment, our system automatically identifies a sequence of proximal surfaces that satisfy the constraints of a specific motion, using a projection and intersection method that efficiently identifies the spatial relationships between surfaces without the need for global sampling or discretization, and taking into account surface-hand contacts. These motion transitions, along with their associated semantics, can be easily integrated into standard navigation systems to enhance the movement repertoire of virtual characters, to rely on contact-rich behaviors while navigating in dynamic, complex virtual worlds.

The computational efficiency of our method enables interactive authoring sessions where environment designers may iteratively design environments and get instant feedback of how characters may potentially interact with it. Additionally, end users may dynamically change game worlds and observe virtual characters automatically modify their movement strategies in order to accommodate motion transitions which may have been invalidated or made possible as a result of the user interaction. We demonstrate several examples of both static and dynamic scenes with multiple characters that were authored using our system. Our approach can be seamlessly integrated into existing animation and navigation pipelines.

2 Related Work

Example-based Motion Synthesis. The widespread availability of high-quality motion capture data has led to the prominence of data-driven approaches such as motion graphs [Kovar et al. 2002; Arikan and Forsyth 2002] for synthesizing motions for virtual characters. Building on top of these foundational contributions, the computer animation community has developed solutions for parameterizing motion graphs [Heck and Gleicher 2007], optimizing its structure for extended types of search [Safonova and Hodgins 2007], annotating them with semantics for efficient querying [Min and Chai 2012], and developing controllers that produce responsive animated characters [McCann and Pollard 2007] which can be interactively controlled [Lee et al. 2002; Lee et al. 2009] and suitable for interactive applications such as games. Lee and Hoon [2004] precompute avatar behavior from unlabeled motion data and adopt dynamic programming in order to animate and control avatars at minimal runtime cost.

Motion Planning. Planning-based approaches have been extensively applied to generate complex motion sequences around obstacles by searching among possible concatenations and blending between motion clips in a library. Lau and Kuffner [2005] propose a method that searches over animation sequences generated by a finite state machine, in order to generate animations that traverse complex environments. Follow-up work [Lau and Kuffner 2006; Lau and Kuffner 2010] describes a precomputed acceleration structure that enables the method to be used in real-time applications. These approaches however only address locomotion behaviors or specific behaviors that avoid collisions with the environment (e.g., jump over, crouch), and do not process contact-rich motions or determine motion fitting to complex environments. Other approaches include planning using probabilistic roadmaps [Choi et al. 2003], and approaches suitable for uneven and sloped terrain [Hauser et al. 2005]. Planning-based approaches have also been extended to dynamic environments by generating motion plans in both space and time [Levine et al. 2011]. In a different direction, the work in [Lo and Zwicker 2008] proposes a real-time planning approach based on reinforcement learning for parameterized human motion. The common disadvantage of these methods is that they do not consider contact-rich motions in complex environments. Our method exactly addresses this limitation by focusing on processing and representing complex motions and environments for real-time planning.

Contact-aware Motion Synthesis In addition to planning in the space of motions, researchers have also explored motion synthesis techniques that explore the space of viable contacts between characters and the environments. Lie *et al.* [2010] reconstruct the implicit contact forces from motion capture data and propose a randomized sampling strategy to synthesize dynamically plausible contact-rich motions. The work in [Tonneau et al. 2014] synthesizes skeletal configurations for the virtual character that satisfy task and contact constraints by sampling the reachable workspace in the environment. The work in [Kalisiak and van de Panne 2001] presents a grasp-based motion planning algorithm for character animation. Mordatch *et al.* [2012] automatically discover complex contact-rich behaviors by using an offline contact-invariant optimization process. Kang and Lee [2014] generate contact-rich character poses by sampling points on the character and the environment to identify admissible support points. Ho and Komura [2009] use *topology coordinates* to synthesize motions that involve close contacts such as wearing a t-shirt. Shape2Pose [Kim et al. 2014] predicts a corresponding human pose including contact points and kinematic parameters by analyzing geometric structures of shapes and local region classification combined with global kinematically-constrained search. The work in [Al-Asqhar et al. 2013] adopts an interactive motion adaptation scheme for close interactions between skeletal characters and mesh structures by introducing a new spatial relationship-based representation which describes the kinematics of the body parts by the weighted sum of translation vectors relative to points selectively sampled over the surface of the mesh.

Dynamics. Physics-based approaches [Coros et al. 2009; Liu et al. 2012] synthesize physically plausible motions that can locally generalize to unknown environments. [Fang and Pollard 2003] optimizes input motion data by introducing physically-based objective functions and constraints, but do not address global behavior planning in complex environments. Our work is complementary to physics-based approaches in which we provide global plans based on input behaviors, which could be specified from the control constraints of physics-based behaviors. In this case a solution from our method could be executed by physics-based controllers.

Path Planning and Multi-Character Navigation. Traditional planning-based navigation approaches [Kallmann and Kapadia 2014] compute a discrete representation of the free space in the environment which can then be efficiently queried to generate walkable paths. Motion Patches [Kim et al. 2012] orchestrates interactions among multiple characters by collecting deformable motion patches which each describes an episode of multiple interacting characters and till them spatially and temporally using a combination of stochastic sampling. These approaches scale to handle hundreds, even thousands of agents at interactive rates, however are restricted to agents with simple movement capabilities. Our work complements these methods by extending navigation meshes to include transitions for contact-rich motions in complex environments.

Contributions. While previous work has explored complex interactions between characters and environments, none has addressed planning motions that require contact-rich behaviors in order to overcome obstacles with detailed hand-surface contact constraints required to use the surfaces in the 3D environment. Our work introduces an environment processing method that is suitable to handle complex animations at interactive rates.

3 Overview

Fig. 2 illustrates the main steps of our framework. Given raw motion data and geometry data of the environment, our system analyzes both of them in order to find valid motion transitions in the environment. Motion transitions are then annotated in the environ-

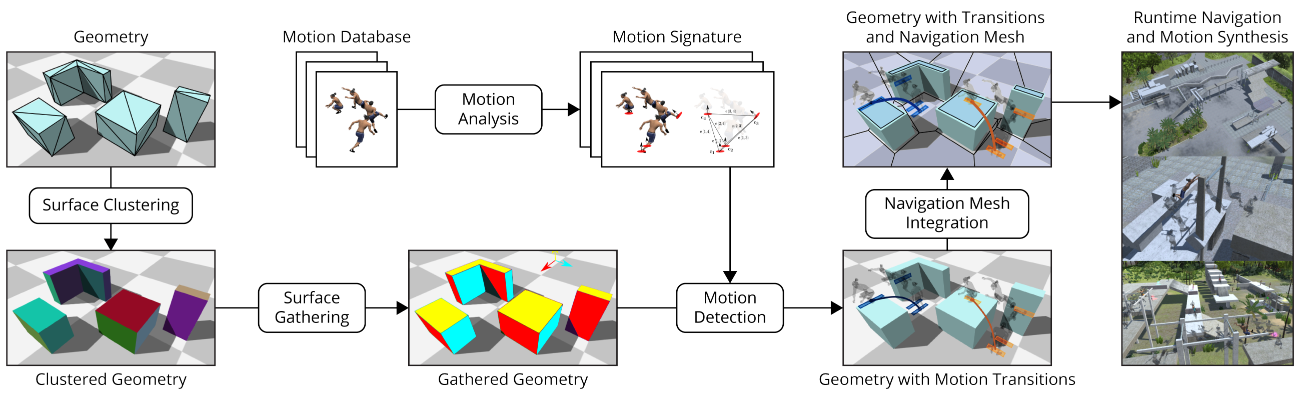

Geometry
Motion Database
Motion Analysis
Motion Signature
Geometry with Transitions and Navigation Mesh
Runtime Navigation and Motion Synthesis
Surface Clustering
Navigation Mesh Integration
Surface Gathering
Motion Detection
Clustered Geometry
Gathered Geometry
Geometry with Motion Transitions

Figure 2: *Framework Overview.*

ment combined with standard navigation mesh representations for runtime pathfinding and motion synthesis.

Motion Analysis. Given a database of motion clips representing different motion skills for a character, we analyze each motion (or a set of similar motions) to define its *motion signature* – which characterizes the different contacts between the character's body parts and the environment during the motion, the spatial constraints between pairs of contacts, as well as the collision bounds of the character while performing the motion.

Environment Analysis. Given an arbitrarily complex 3D environment, our system identifies surfaces that can be used to support the character using different kinds of contacts. For each motion signature (a sequence of contact configurations), a corresponding sequence of proximal contact surfaces is identified which satisfy the spatial constraints between the contacts in the motion. We use a "projection and intersection" based method that efficiently identifies spatial relationships between surfaces and can handle deformable motion models. The set of valid surfaces are pruned to ensure that the resulting motion performed in that part of the environment is collision-free, producing an annotated environment that identifies areas where different motion skills are possible.

Runtime Pathfinding and Motion Synthesis. We extend traditional navigation graph approaches to include additional edges for connecting disconnected surfaces using a particular motion skill, with semantics that codify motion, contact, and collision constraints. During runtime, standard search techniques can be used on this navigation graph to generate a path that includes these behaviors. A data-driven animation system is used to animate characters to navigate along this path, and the annotated motion transitions are used to animate the character to perform complex animations that satisfy the exact contact constraints without the need for expensive configuration-space motion planning.

4 Motion Analysis

The movement repertoire of an animated character is defined by a set of motion skills $\mathbf{m} \in \mathbf{M}$. A motion skill may be a single motion clip or a set of similar motion clips which can be blended together to create a parameterized motion skill that can produce new contact variations that are not present in the original motions.

Each motion skill is abstracted as $\mathbf{m} = \langle \mathbf{G}, \mathbf{V}, \mathbf{L} \rangle$ where: (1) the contact constraint graph $\mathbf{G} = \langle \mathbf{C}, \mathbf{E} \rangle$ is defined by a set of contacts $\mathbf{c} \in \mathbf{C}$, and spatial constraints $\mathbf{e}(i,j) \in \mathbf{E}$ between pairs of contacts $(\mathbf{c_i}, \mathbf{c_j}) \in \mathbf{C}$. (2) $\mathbf{V}$ represents the collision bounds of the character over the course of the motion. (3) $\mathbf{L}$ is a data-driven controller that is used for animating the character and may be a single motion clip coupled with inverse kinematics, or a parameterized blend tree of similar motions.

4.1 Contact Constraint Graph

Over the course of the motion, different parts of the characters body might come in contact with different surfaces of the environment. We define a contact pose $\mathbf{P} = \{\mathbf{c_i}\}$ as a body part (e.g., hands or feet) coming in contact with one or more surfaces in the environment. Each contact $\mathbf{c} = \{\mathbf{p}, \mathbf{n}\}$ denotes the contact of a body part with one environment surface, defined using the relative position $\mathbf{p}$ and surface normal $\mathbf{n}$ of the contact, with respect to the initial position of the center of mass of the character. Fig. 3 illustrates example contact poses.

A constraint $\mathbf{e}(i,j) \in \mathbf{E}$ defines the distance interval (d_{min}, d_{max}) between the two contacts $\mathbf{c_i}, \mathbf{c_j} \in \mathbf{C}$. For a motion skill that is a single motion clip without any procedural deformation, this will be a single numeric value. We do not consider temporal constraints as they are not necessary in a static snapshot of an environment.

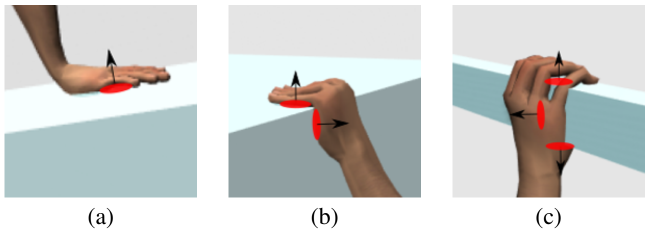


Figure 3: *Contact Poses. (a) Planar contact pose. (b) L-shape contact pose: two surfaces with angle between normals close to 90 degrees. This corresponds to placing hands or other body part on a perpendicular edge in the environment. (c) Grasping contact: three planar surfaces with normals spanning 180 degrees, corresponding to placing hands on a curved surface.*

For each motion, we identify the set of keyframes during which a contact takes place using a semi-automated method, as described in [Xiao et al. 2006]. A keyframe may have multiple body parts in contact at a time, leading to simultaneous contact poses. Fig. 4 illustrates the keyframes of selected motions with their contacts highlighted. Fig. 5 illustrates an example of a contact constraint graph for a tic-tac motion. Our constraint graph allows to represent constraints between all pairs of contacts. This is useful to specify global constraints, for example, such as lower and upper limits

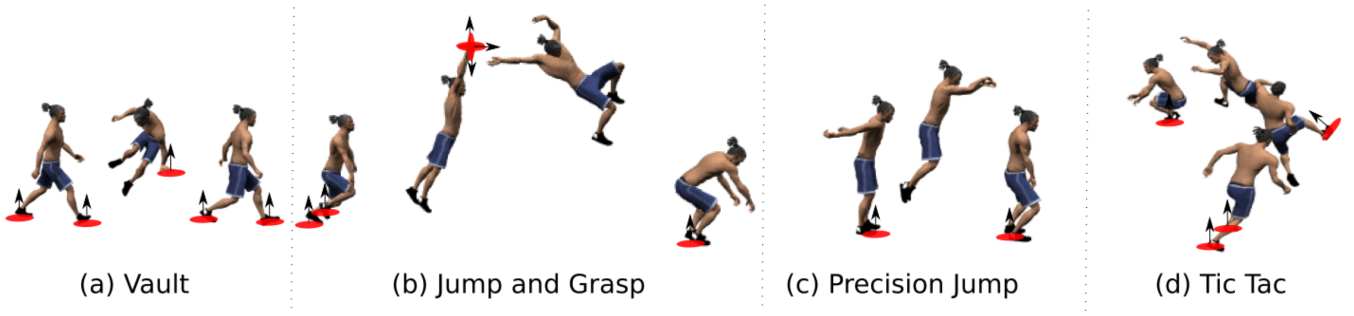


Figure 4: *Keyframes of selected motions with annotated contacts.*

between the first and last contacts for the behavior to be executed.

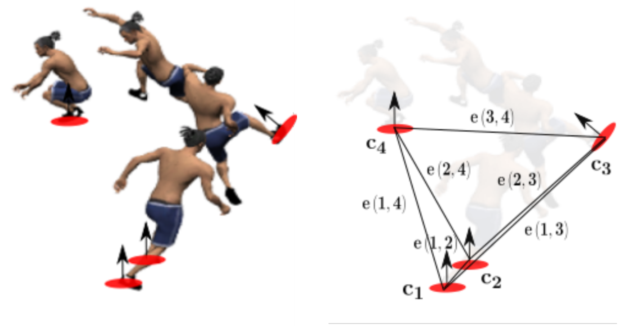


Figure 5: *Contact Constraint Graph for a tic-tac motion. The nodes represent the contacts* $\mathbf{c} \in \mathbf{C}$ *over the course of the motion, and the edges represent spatial constraints* $\mathbf{e}(.,.) \in \mathbf{E}$ *between contacts.*

4.2 Collision Representation

In order to accurately handle the complex interactions between characters and the environment, the system should satisfy not only contact constraints but also collision constraints. One approach is to precompute the sweeping volume of the character's mesh over the course of the motion, which can be then used for collision checks. However, this is very computationally expensive. To enable motion collision detection at interactive rates (useful for interactive authoring or games), we provide a coarse approximation of the character's collision model as follows. For each motion skill, we first extract the representative keyframes from trajectories of a subset of the joints (we used the head, root, feet, and hands). A compound collider using a combination of cylinders represents the collision volume of the character at each of these keyframes, and is used to check if a collision takes place. This coarse collision representation presents a tradeoff between computational performance and accuracy, and may produce false positives or negatives, but works well for the results described in the paper. Applications that have strict requirements on accuracy may use more complex collision representations [Hudson et al. 1997]. Fig. 6 illustrates detection of collisions for a tic-tac motion.

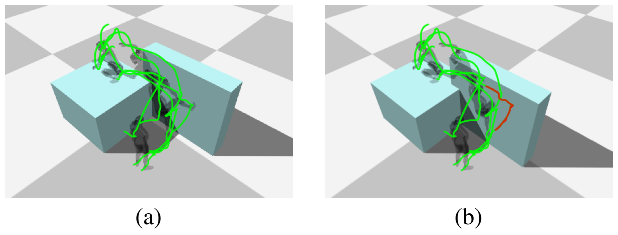


Figure 6: *Collision Check Precomputations. (a) A collision-free motion. (b) Collision representation collides with environment during the parts of the trajectory which are highlighted in red.*

5 Environment Analysis

Once the motions have been processed we analyze the environment to automatically detect locations where motion skills may occur.

5.1 Contact Surface Detection

The environment $\mathbf{W}$ is a collection of objects $\mathbf{o}$. The geometry of each object is represented as a set of discrete triangles generated from the environment modeling stage. We first cluster adjacent triangles into larger surfaces $\mathbf{s}$ that have the same surface normals $\mathbf{n_s}$. Starting from a triangle in the environment, we perform a breadth-first expansion to neighbouring triangles and cluster adjacent triangles that have similar facing normals. This clusters the environment triangulation into a set of contact surfaces $\mathbf{s} \in \mathbf{S}$, where $\mathbf{s}$ is a collection of adjacent triangles that face the same (or similar) direction. Fig. 7 illustrates the contact surface clustering process.

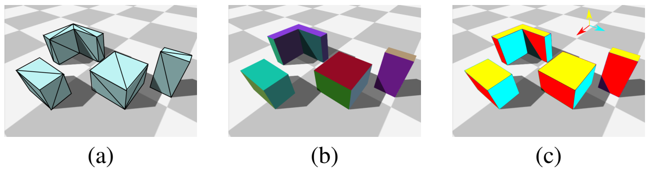


Figure 7: *Surface Clustering and Gathering. (a) Original Environment mesh. (b) Triangles with the same normals are clustered together to form surfaces. (c) Surfaces that share the same normal properties are grouped and mapped to corresponding contacts in the motion.*

5.2 Motion Detection

Given a set of surfaces $\mathbf{s} \in \mathbf{S}$ characterizing the environment, we seek to identify spatially co-located surfaces that satisfy the constraints on the motion signature of a specific motion skill, while avoiding collisions with any geometry, and to identify the valid subregions on each surface where each contact in the motion might take place. We provide an overview of the steps for detecting a valid transition in the environment for a motion skill $\mathbf{m}$. The complete algorithm is provided in Alg. 2.

Contact Rotation. The contacts $\mathbf{c} \in \mathbf{C}$ of a motion skill contain the relative positions of all contacts in the motion with respect to the initial position of the character's center of mass, assuming the character is initially facing along the positive X axis. To detect the execution of the motion skill in any direction, we first sample the set of valid start directions and transform the contact signature for the chosen direction. Note that the spatial constraints between contacts remain unaffected as a result of this rotation.

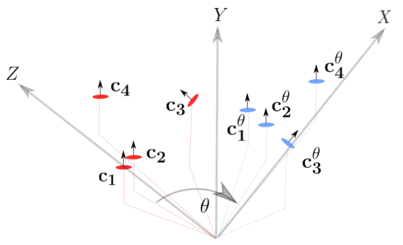


Figure 8: *Contact Rotation.*

Surface Gathering. Next, we identify all contact surfaces $\mathbf{s_i} \subseteq \mathbf{S}$ with a surface normal which is consistent with each contact $\mathbf{c_i} \in \mathbf{C}$ for the motion skill. Fig. 7(c) illustrates the surface gathering process.

Contact Constraint Satisfaction. For each pair of contact surfaces $(\mathbf{s_a}, \mathbf{s_b}) \in \mathbf{S_1} \times \mathbf{S_2}$, we find the subregions $(\mathbf{s_a^{'}}, \mathbf{s_b^{'}})$ on each surface that satisfy the spatial constraint $\mathbf{e}(1,2)$ between the first two contacts $\mathbf{c_1}$ and $\mathbf{c_2}$ in the motion. This is accomplished using the algorithm described in § 5.3. This produces a new set $\mathbf{R} \in \mathbf{S_1^{'}} \times \mathbf{S_2^{'}}$ comprising pairs of contact surfaces that satisfy the spatial constraint between the first two contacts. This process is iteratively repeated for each subsequent contact in $\mathbf{C}$ to produce $\mathbf{R} \in \mathbf{S_1^{'}} \times \cdots \mathbf{S_{|C|}^{'}}$ which contains a set of surface sequences that accommodate all the contacts in the motion, and each surface represents the set of all valid locations where the particular contact in the motion may take place.

Collision Constraint Satisfaction. Each sequence of candidate surfaces in $\mathbf{r} = \langle \mathbf{s_1}, \ldots \mathbf{s_{|C|}} \rangle \in \mathbf{R}$ must additionally be checked to see if the motion produces collisions with any aspect of the environment. This translates to finding all points on the start region $\mathbf{s_1} \in \mathbf{r}$ such that the motion can be successfully performed starting from that location. We adaptively subdivide $\mathbf{s_1}$ to find all subregions within the start region that are collision-free. The collision check is performed using the coarse collisions representation of the motion described in § 4.2. Note that this process may divide a start region into multiple disconnected regions from which the motion skill maybe performed.

5.3 Contact Constraint Satisfaction

We check the spatial relationship between two contact surfaces and return subsurfaces that satisfy the distance constraint imposed between them, as specified by the contact constraint graph. We consider a non-deformable motion skill (e.g., a single motion clip with no procedural modification), where the spatial constraint is a single discrete value. For motion skills that have deformable motion models (e.g., a blend tree of multiple similar motion clips and/or procedural deformation using inverse kinematics), this is a distance interval, which is not considered here.

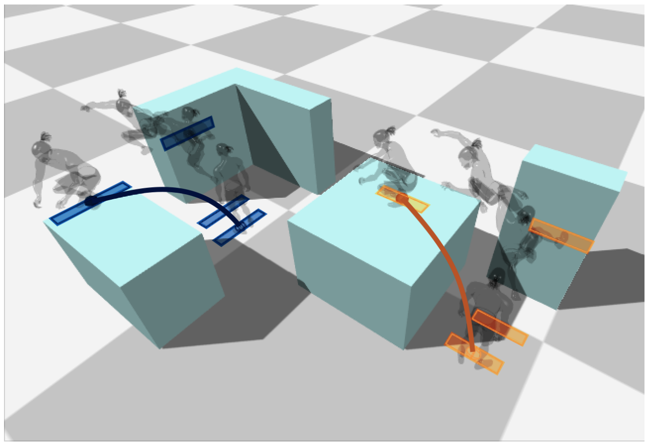


Figure 9: *Motion detection results.*

For a pair of contacts $(\mathbf{c_i}, \mathbf{c_j}) \in \mathbf{C}$, let $\hat{\mathbf{v}}_{\mathbf{i,j}}$ represent the unit vector along the direction of $\mathbf{p}_i - \mathbf{p}_j$. Let d denote the desired distance between the two contacts. Let $\mathbf{S_i}$, $\mathbf{S_j}$ be the set of contact surfaces that have the same normal direction as $\mathbf{n}_i$ and $\mathbf{n}_j$ respectively. Let $\mathbf{s_a}, \mathbf{s_b}$ be a pair of contact surfaces in $\mathbf{S_i} \times \mathbf{S_j}$. We translate the boundary points of $\mathbf{s_a}$ along $\hat{\mathbf{v}}_{\mathbf{a,b}} \cdot d$ and check if the translated surface overlaps with $\mathbf{s_b}$ to produce $\mathbf{s_a^{'}}$ (polygon intersection operation). This process is repeated for $\mathbf{s_b}$ to return the pair of subsurfaces $\langle \mathbf{s_a^{'}}, \mathbf{s_b^{'}} \rangle$ that satisfy the spatial constraint between the two contacts. If either of the translated surfaces do not overlap, this constraint is not satisfied between $\mathbf{s_a}$,$\mathbf{s_b}$. Fig. 10 illustrates the projection and intersection step, and the steps are outlined in Alg. 1.

Algorithm 1: ProjectIntersect ($\mathbf{s_a}$, $\mathbf{s_b}$, $\hat{\mathbf{v}}_{\mathbf{i,j}}$, d)

```
1 $\mathbf{s_a^*} \leftarrow \mathbf{s_a} + \hat{\mathbf{v}}_{\mathbf{i,j}} \cdot d$
2 $\mathbf{s_b^{'}} = \mathbf{s_a^*} \cap \mathbf{s_b}$
3 $\mathbf{s_b^*} = \mathbf{s_b} - \hat{\mathbf{v}}_{\mathbf{i,j}} \cdot d$
4 $\mathbf{s_a^{'}} = \mathbf{s_b^*} \cap \mathbf{s_a}$
5 return $\langle \mathbf{s_a^{'}}, \mathbf{s_b^{'}} \rangle$
```

6 Runtime Navigation and Motion Synthesis

6.1 Runtime Navigation

The motion detection algorithm described above produces a a set of surface sequences $\mathbf{R}$ that correspond to viable transitions in the environment using the specific motion skill. Each surface sequence $\mathbf{r} = \langle \mathbf{s_1}, \ldots \mathbf{s_{|C|}} \rangle$ contains information of the surfaces of the environment which can be used to support all the contacts $\mathbf{C}$ for a particular motion skill $\mathbf{m}$. The surface $\mathbf{s_1}$ represents the set of all possible locations from which the character can successfully start executing $\mathbf{m}$ without violating contact or collision constraints. $\mathbf{s_{|C|}}$ represents the surface where the character ends the motion. This translates to a motion transition $\mathbf{t} : \mathbf{s_1} \rightarrow \mathbf{s_{|C|}}$ which uses the motion skill, $\mathbf{m}$. $\mathbf{T}$ is the set of all motion transitions for all elements

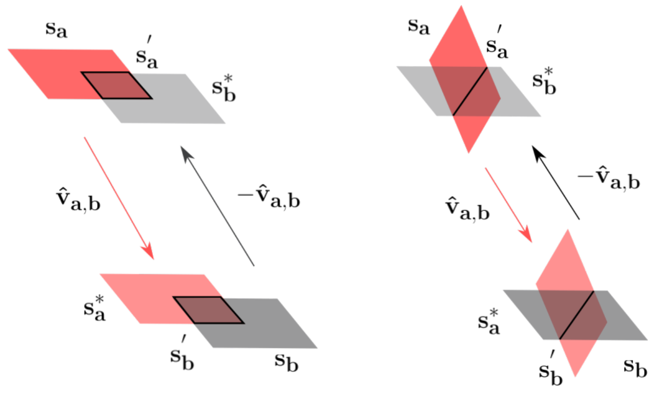


Figure 10: ProjectIntersect *for a motion skill. (a) Two contact surfaces are parallel to each other, the overlapping region will either be a 2D polygon or null. (b) Two contact surfaces are in arbitrary positions relative to each other, the overlapping region will either be a line segment or null.*

Algorithm 2: DetectMotions $(\mathbf{M}, \mathbf{S})$

```
foreach $\mathbf{m} \in \mathbf{M}$ do
    foreach $\theta \in (0, 2\pi)$ do
        foreach $i \in (1, |\mathbf{C_m}|)$ do
            $\mathbf{c_i}^{\theta} = \mathbf{Rotate}(\mathbf{c_i}, \theta)$
            foreach $\mathbf{s} \in \mathbf{S}$ do
                if $\mathbf{n_s} \cdot \mathbf{n_{c_i^{\theta}}} \le \epsilon$ then
                    $\mathbf{S_i} \leftarrow \mathbf{S_i} \cup \mathbf{s}$
        $\mathbf{R} \leftarrow \mathbf{S_1}$
        foreach $i \in (2, |\mathbf{C_m}|)$ do
            $\mathbf{R}_t \leftarrow \emptyset$
            foreach $(\mathbf{s_a}, \mathbf{s_b}) \in \mathbf{R} \times \mathbf{S_i}$ do
                $\hat{\mathbf{v}}_{\mathbf{i-1,i}} \leftarrow \frac{\mathbf{p}_{i-1} - \mathbf{p}_i}{||\mathbf{p}_{i-1} - \mathbf{p}_i||}$
                $\langle \mathbf{s_a}', \mathbf{s_b}' \rangle \leftarrow$ ProjectIntersect$(\mathbf{s_a}, \mathbf{s_b}, \hat{\mathbf{v}}_{\mathbf{i-1,i}}, d)$
                $\mathbf{R}_t \leftarrow \mathbf{R}_t \cup \langle \mathbf{s_a}', \mathbf{s_b}' \rangle$
            $\mathbf{R} \leftarrow \mathbf{R}_t$
        $\mathbf{R} \leftarrow$ CollisionCheck$(\mathbf{R}, \mathbf{V_m})$;
        return $\mathbf{R}$
```

in $\mathbf{R}$.

Let $\mathbf{\Sigma} = \langle \mathbf{N}, \mathbf{A} \rangle$ be a navigation graph generated using standard approaches such as navigation meshes [Memononen 2014], where $\mathbf{N}$ are walkable regions in the environment (e.g., polygonal surfaces) and $\mathbf{A}$ represent valid transitions between elements in $\mathbf{N}$ using simple locomotion behaviors (e.g., walking or running). We generate an extended navigation graph $\mathbf{\Sigma}^+ = \langle \mathbf{N} \cup \mathbf{S}, \mathbf{A} \cup \mathbf{R} \rangle$ that includes the transitions between previously disconnected environment surfaces using the extended set of motion skills available to the character. A navigation mesh extended with motion transitions is illustrated in Fig. 11.

Path Computation and Steering. Given the current and desired location of the character $\mathbf{s}_0$, $\mathbf{s}_g$ respectively, we use standard discrete search methods such as A* [Hart et al. 1968] to generate a plan $\pi = \mathbf{Plan}(\mathbf{s}_0, \mathbf{s}_g, \mathbf{\Sigma}^+)$ which can include one or more motion transitions $\mathbf{t} \in \mathbf{T}$. A steering system [Singh et al. 2011] is used to move the root of the character along the computed path while avoiding other characters. Note that the steering system is only used while moving (walking or running) along standard transitions $\mathbf{A}$ in the navigation graph. The motion synthesis system described below superimposes an animated character to follow the

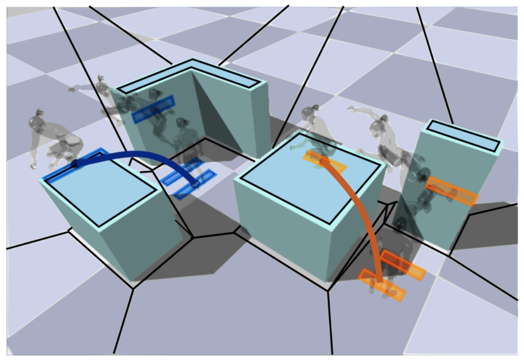
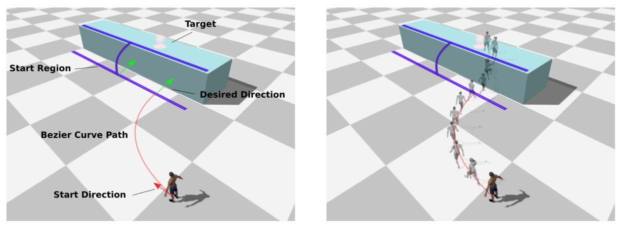


Figure 11: *Integration of motion transitions into a navigation mesh.*



(a) (b)

Figure 12: *Generation of smooth motion path to enter a motion transition while satisfying position, orientation, and speed constraints.*

trajectory generated by steering, and handles the motion of the character while traversing motion transitions $\mathbf{t} \in \mathbf{T}$.

6.2 Motion Synthesis

Once the path π is determined, the animation system will synthesize a continuous motion to move the character from its current position to the target. We use a standard move graph approach to construct a set of animation states, corresponding to the various movement capabilities of the character. For example, there is a "locomotion" state that handles the basic movement such as walking or running. For more details on the implementations of the basic locomotion system, please refer to [Johansen 2009]. There is one animation state for each new motion skill which can be easily added into an existing animation state machine by defining the appropriate transitions between motion states. We manually construct this state machine, but automated solutions can also be used [Zhao et al. 2009].

Each motion transition specifies which skill to use, the facing direction of the character, the admissible start region, and desired entry speed (e.g., a tic tac motion skill may require the character to be running prior to entry). This imposes position, orientation, and speed constraints that the character must satisfy. A naive *stop and turn* strategy is to allow the character to simply move to the desired location and then turn in place. However, this introduces undesirable motion discontinuities and violates speed constraints for more dynamic behaviors.

The second strategy which works well for our application is to generate a cubic Bézier curve using the current position and orientation of the character to select the initial control points, and adaptively sampling a target location in the start region while meeting the desired orientation constraint, to produce a collison-free continuous path. The character linearly interpolates from its current speed to the desired speed while traveling along the curve. This process is illustrated in Fig. 12. In the remote case that no collision-free curve is found, the character continues to move along a straight line towards the centre of the start region, and repeats the process.

7 Results

We evaluate the performance of our approach in its ability to detect a variety of motions in complex 3D environments. We used a total of 16 motion skills including climbing, squat and roll, double hand vaults, precision jumps etc. All results were generated using a single-threaded application on a Macbook Pro, 2.4 Ghz Intel Core i7 processor, with 8GB 1600 Mhz DDR3 RAM.

As shown in Fig. 14, our system seamlessly integrates into existing navigation and animation pipelines to produce virtual characters that can autonomously reason about their environments while harnessing an increased repertoire of motion skills to navigate in complex virtual worlds. Our approach scales to handle multiple characters while still maintaing real-time rates.

Dynamic Game Worlds. The computational efficiency of our approach also allows motion transitions to be repaired at interactive rates, enabling new applications including dynamic game worlds. We demonstrate two example game demos in the supplementary video. In the first example, a player interactively manipulates objects in the game world while an autonomous character navigates in the environment. The second example shows many characters chasing a player-controlled character in the game.

Interactive Authoring of Game Worlds. We have developed a simple authoring tool that allows game level designers to interactively author game worlds while harnessing the power of our approach. Using our tool, an author simply drags and drops geometry into the scene. Our system works behind the scenes to detect and annotate potential ways in which a character may interact in the environment by highlighting the different possible motion transitions between objects. At any time, the author can see a real-time animation of the character navigating in the current environment to get instant visual feedback, which can be used to iteratively refine the initial design. The supplementary video shows the interactive authoring session to create the complex scene shown in Fig. 14.

Computational Efficiency. Fig. 13 analyses the computational efficiency of our approach. For complex environments with greater than 3000 surface elements (more than 20,000 triangles), nearly 100 motion transitions for 10 motion skills are detected within 5 seconds. For local repair of motion transitions for dynamic and interactive use, the operation is instantaneous.

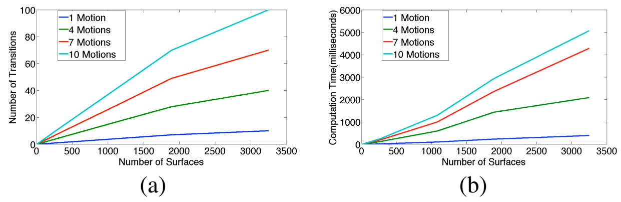


Figure 13: *Computational Efficiency Analysis.*

8 Conclusion

This paper presents an automated system for identifying and annotating the different ways in which an environment can afford a virtual character to move by using its surfaces. Detailed hand-surface contact constraints are correctly taken into account. Our method only receives as input the raw contact-rich motion data and the environment geometry. We provide efficient algorithms for precomputing and annotating motion semantics, and integrating them into standard navigation and animation pipelines, to greatly enhance the movement capabilities of virtual characters. Our tool can be used in a variety of ways. For example, virtual world designers can get instant feedback of how characters may potentially interact in their current designed environments. Next-generation games can support truly dynamic game worlds where players may dramatically manipulate the environment geometry which is automatically reflected in the movement strategies of autonomous virtual characters.

Our work is complementary to the rich body of work in character animation, navigation, and local collision-avoidance, and can benefit from the recent advances in these areas. Improved representations of the collision volumes [Hudson et al. 1997] of a motion can address fine collision checks without imposing a considerable computational overhead. Our algorithms for motion detection can be extended to support deformable motion models [Choi et al. 2011], more complex motion databases [Kapadia et al. 2013], and complex multi-character interactions that have narrative significance [Shoulson et al. 2013; Shoulson et al. 2014]. An interesting avenue for future exploration is to investigate the use of physically-based controllers that rely on environment semantics to guide their control policies. Finally, our work has potential applications beyond character animation to assist in the computational design of environments [Berseth et al. 2015].

References

Al-Asqhar, R. A., Komura, T., and Choi, M. G. 2013. Relationship descriptors for interactive motion adaptation. In *ACM SIGGRAPH/EG SCA*, ACM, 45–53.

Arikan, O., and Forsyth, D. A. 2002. Interactive motion generation from examples. *ACM Trans. Graph. 21*, 3 (July), 483–490.

Berseth, G., Usman, M., Haworth, B., Kapadia, M., and Faloutsos, P. 2015. Environment optimization for crowd evacuation. *CAVW 26*, 3-4, 377–386.

Choi, M. G., Lee, J., and Shin, S. Y. 2003. Planning biped locomotion using motion capture data and probabilistic roadmaps. *ACM Trans. Graph. 22*, 2 (Apr.), 182–203.

Choi, M. G., Kim, M., Hyun, K., and Lee, J. 2011. Deformable Motion: Squeezing into Cluttered Environments. *Comput. Graph. Forum 30*, 2, 445–453.

Coros, S., Beaudoin, P., and van de Panne, M. 2009. Robust task-based control policies for physics-based characters. *ACM Trans. Graph. 28*, 5 (Dec.), 170:1–170:9.

Fang, A. C., and Pollard, N. S. 2003. Efficient synthesis of physically valid human motion. In *ACM SIGGRAPH*, ACM, 417–426.

Hart, P., Nilsson, N., and Raphael, B. 1968. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on 4*, 2 (July), 100–107.

Hauser, K., Bretl, T., and Latombe, J.-C. 2005. Non-gaited humanoid locomotion planning. In *Humanoid Robots, 2005 5th IEEE-RAS International Conference on*, 7–12.

Heck, R., and Gleicher, M. 2007. Parametric motion graphs. In *ACM SIGGRAPH I3D*, 129–136.

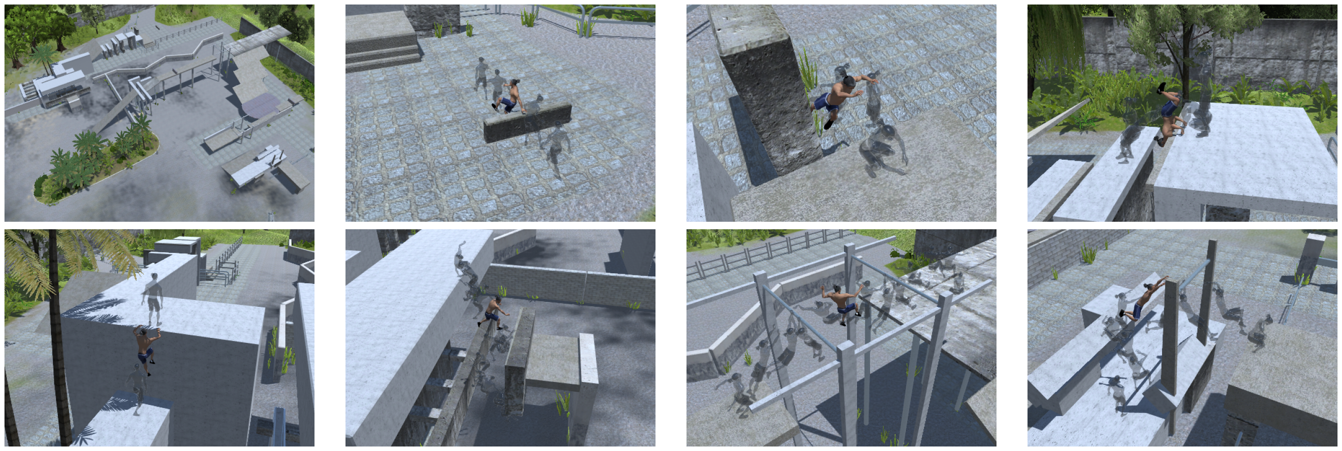


Figure 14: *Navigation in complex static environment.*

HO, E. S. L., AND KOMURA, T. 2009. Character motion synthesis by topology coordinates. In *CGF*, vol. 28.

HUDSON, T. C., LIN, M. C., COHEN, J., GOTTSCHALK, S., AND MANOCHA, D. 1997. V-collide: Accelerated collision detection for vrml. In *VRML*, ACM, 117–ff.

JOHANSEN, R. 2009. *Automated semi-procedural animation for character locomotion*. PhD thesis.

KALISIAK, M., AND VAN DE PANNE, M. 2001. A grasp-based motion planning algorithm for character animation. *Journal of Visualization and Computer Animation 12*, 3, 117–129.

KALLMANN, M., AND KAPADIA, M. 2014. Navigation meshes and real-time dynamic planning for virtual worlds. In *ACM SIGGRAPH 2014 Courses*, ACM, New York, NY, USA, SIGGRAPH '14, 3:1–3:81.

KANG, C., AND LEE, S.-H. 2014. Environment-adaptive contact poses for virtual characters. *Computer Graphics Forum 33*, 7, 1–10.

KAPADIA, M., CHIANG, I.-K., THOMAS, T., BADLER, N. I., AND KIDER, JR., J. T. 2013. Efficient motion retrieval in large motion databases. In *ACM SIGGRAPH I3D*, ACM, 19–28.

KIM, M., HWANG, Y., HYUN, K., AND LEE, J. 2012. Tiling motion patches. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, SCA '12, 117–126.

KIM, V. G., CHAUDHURI, S., GUIBAS, L., AND FUNKHOUSER, T. 2014. Shape2pose: Human-centric shape analysis. *ACM Trans. Graph. 33*, 4 (July), 120:1–120:12.

KOVAR, L., GLEICHER, M., AND PIGHIN, F. 2002. Motion graphs. In *ACM SIGGRAPH*, ACM, 473–482.

LAU, M., AND KUFFNER, J. J. 2005. Behavior planning for character animation. In *ACM SIGGRAPH/EG SCA*, ACM, 271–280.

LAU, M., AND KUFFNER, J. J. 2006. Precomputed search trees: Planning for interactive goal-driven animation. In *ACM SIGGRAPH/Eurographics SCA*, 299–308.

LAU, M., AND KUFFNER, J. 2010. Scalable precomputed search trees. In *MIG*, 70–81.

LEE, J., AND LEE, K. H. 2004. Precomputing avatar behavior from human motion data. In *ACM SIGGRAPH/EG SCA*, Eurographics Association, 79–87.

LEE, J., CHAI, J., REITSMA, P. S. A., HODGINS, J. K., AND POLLARD, N. S. 2002. Interactive control of avatars animated with human motion data. In *ACM SIGGRAPH*, ACM, 491–500.

LEE, Y., LEE, S. J., AND POPOVIĆ, Z. 2009. Compact character controllers. *ACM Trans. Graph. 28*, 5 (Dec.), 169:1–169:8.

LEVINE, S., LEE, Y., KOLTUN, V., AND POPOVIĆ, Z. 2011. Space-time planning with parameterized locomotion controllers. *ACM Trans. Graph. 30*, 3 (May), 23:1–23:11.

LIU, L., YIN, K., VAN DE PANNE, M., SHAO, T., AND XU, W. 2010. Sampling-based contact-rich motion control. *ACM Trans. Graph. 29*, 4 (July), 128:1–128:10.

LIU, L., YIN, K., VAN DE PANNE, M., AND GUO, B. 2012. Terrain runner: control, parameterization, composition, and planning for highly dynamic motions. *ACM Trans. Graph. 31*, 6 (Nov.), 154:1–154:10.

LO, W.-Y., AND ZWICKER, M. 2008. Real-time planning for parameterized human motion. In *ACM SIGGRAPH/Eurographics SCA*, 29–38.

MCCANN, J., AND POLLARD, N. 2007. Responsive characters from motion fragments. *ACM Trans. Graph. 26*, 3 (July).

MEMONONEN, M. 2014. Recast: Navigation-mesh toolset for games.

MIN, J., AND CHAI, J. 2012. Motion graphs++: A compact generative model for semantic motion analysis and synthesis. *ACM Trans. Graph. 31*, 6 (Nov.), 153:1–153:12.

MORDATCH, I., TODOROV, E., AND POPOVIĆ, Z. 2012. Discovery of complex behaviors through contact-invariant optimization. *ACM Trans. Graph. 31*, 4 (July), 43:1–43:8.

SAFONOVA, A., AND HODGINS, J. K. 2007. Construction and optimal search of interpolated motion graphs. In *ACM SIGGRAPH*.

SHOULSON, A., MARSHAK, N., KAPADIA, M., AND BADLER, N. I. 2013. ADAPT: the agent development and prototyping testbed. In *ACM SIGGRAPH I3D*, ACM, 9–18.

SHOULSON, A., MARSHAK, N., KAPADIA, M., AND BADLER, N. I. 2014. Adapt: The agent developmentand prototyping testbed. *IEEE TVCG 20*, 7 (July), 1035–1047.

SINGH, S., KAPADIA, M., REINMAN, G., AND FALOUTSOS, P. 2011. Footstep navigation for dynamic crowds. *CAVW 22*, 2-3, 151–158.

TONNEAU, S., PETTRÉ, J., AND MULTON, F. 2014. Task efficient contact configurations for arbitrary virtual creatures. In *Graphics Interface*, 9–16.

UBISOFT. 2014. Assassins creed.

XIAO, J., ZHUANG, Y., YANG, T., AND WU, F. 2006. An efficient keyframe extraction from motion capture data. In *Advances in Computer Graphics*, vol. 4035 of *LNCS*. Springer Berlin Heidelberg, 494–501.

ZHAO, L., NORMOYLE, A., KHANNA, S., AND SAFONOVA, A. 2009. Automatic construction of a minimum size motion graph. In *ACM SIGGRAPH/EG SCA*, ACM, 27–35.