Multi-Contact Locomotion Using a Contact Graph with Feasibility Predictors

CHANGGU KANG
Gwangju Institute of Science and Technology (GIST)
and
SUNG-HEE LEE
Korea Advanced Institute of Science and Technology (KAIST)

Multi-contact locomotion that uses both the hands and feet in a complex environment remains a challenging problem in computer animation. To address this problem, we present a contact graph, which is a motion graph augmented by learned feasibility predictors, namely contact spaces and an occupancy estimator, for a motion clip in each graph node. By estimating the feasibilities of candidate contact points that can be reached by modifying a motion clip, the predictors allow us to find contact points that are likely to be valid and natural before attempting to generate the actual motion for the contact points. The contact graph thus enables the efficient generation of multi-contact motion in two steps: planning contact points to the goal and then generating the whole-body motion. We demonstrate the effectiveness of our method by creating several climbing motions in complex and cluttered environments by using only a small number of motion samples.

Categories and Subject Descriptors: I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism—*Animation*

General Terms: Algorithms, Design

Additional Key Words and Phrases: Character animation, multi-contact motion, motion planning, motion graph, motion deformation

ACM Reference Format:

Changgu Kang and Sung-Hee Lee. 2017. Multi-contact locomotion using a contact graph with feasibility predictors. ACM Trans. Graph. 36, 2, Article 22 (April 2017), 14 pages.
DOI: http://dx.doi.org/10.1145/2983619

1. INTRODUCTION

Enabling virtual characters to locomote using both the hands and feet in a complex and cluttered environment, such as climbing up cliffs and passing through holes, remains a challenging problem in computer animation research. Since one needs to deal with the contact and penetration against the environment induced by the character's motion, many existing animation techniques assuming a collision-free environment cannot be easily applied to this problem.

In general, such a multi-contact locomotion problem has been addressed by randomized search-based approaches that adopted two-step planning: First, candidate contact points are sampled on the surface of the environment and then *local planning* is performed, in which the quality of the contact points is examined by generating a motion that reaches the contact points. While the quality of contact points and motions depends on many factors, they should at least satisfy the following conditions:

(1) *Kinematic feasibility*: The character should be able to reach the contact points while satisfying joint limits.
(2) *Physical feasibility*: The motion should be balanced and not require excessive force.
(3) *Geometric feasibility*: The character's body should not penetrate the environment.

In addition, the motion should look natural and appropriate for locomotion. Since the local planning step consumes a considerable amount of computation to test these feasibilities, the motion generation process slows down significantly if many candidate contact points examined turn out to be infeasible. Therefore, it is critical to increase the hit rate of sampled candidate contact points for the efficient generation of multi-contact motions.

Sampling around the captured motion data is an effective way to increase the chances to find feasible and natural motions. However, multi-contact motion data are not widely available, because it is difficult to capture motions that interact with the environment, for example, due to occlusion. In fact, the sheer diversity of the environment shapes makes it nearly impossible to prepare enough motion data that cover a wide range of environments. Thus, it is important to be able to modify the captured motion data and make the data applicable to different environments, but the increased difference between the original and modified motions may degrade the feasibility and naturalness.

In this article, we present a method for creating multi-contact locomotion based on a motion graph constructed by a small number of motion capture sequences. To support multi-contact planning, our motion graph is constructed in the manner that a node of a graph represents a unitary change in contact configuration. Thus, we call our motion graph the *"contact graph."*

The key contribution of our method is the precomputation of range of feasible motions that can be obtained by modifying a motion clip stored in a node in the contact graph. In particular, the feasible range is stored in forms that can efficiently test candidate contact points. One is the *contact space*, which represents the space

This work was supported by the Global Frontier R&D Program funded by NRF, MSIP, Korea (2015M3A6A3073743).

Authors' addresses: C. Kang and S.-H. Lee, GSCT, KAIST, 291 Daehak-ro, Yuseong-gu, Daejeon Korea; emails: ckang@gist.ac.kr, sunghee.lee@kaist.ac.kr.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© 2017 ACM 0730-0301/2017/04-ART22 $15.00
DOI: http://dx.doi.org/10.1145/2983619

of feasible contact points that can be obtained from a motion clip. The points included in the contact space satisfy the kinematic and physical feasibilities. In addition, a contact space is modeled with a probability distribution function that reflects the naturalness of the contact points. The other form is the *occupancy estimator*, which estimates the space occupied by a motion to make candidate contact points and thus allows us to estimate the geometric feasibility of contact points. To summarize, each node of a contact graph possesses a motion clip, as well as corresponding contact spaces and an occupancy estimator that enable the examination of the feasibility and naturalness of contact points before generating the actual motions for the contact points.

In order to use a motion clip for different environments and contact points, we need to modify it such that it reaches target contact points while preserving the original motion style. Researchers have developed many approaches for motion modification in differing qualities and complexities, from conventional inverse kinematics to dynamic space-time optimization. Among these approaches, a motion deformation method based on geometric mesh deformation techniques has the advantages of smooth change of motion, preserving local styles, and fast computation by solving a linear system as opposed to solving iterative optimization carried out by inverse kinematics. We follow the Laplacian motion deformation method [Choi et al. 2011] in which a mesh representing a motion is generated and deformed using the Laplacian mesh editing technique [Lipman et al. 2004]. As noted by Choi et al. [2011], applying the Laplacian mesh editing to a three-dimensional (3D) motion mesh brings about undesirable results, because it modifies the mesh by solving a linear system.

An additional contribution of our work is a new variant of the Laplacian motion deformation method. Our proposed method achieves reasonable deformation of motion in two steps: It first finds the end effector paths by using Laplacian curve editing and then finds paths for the whole joints by using Laplacian mesh editing with the end effector paths as boundary constraints.

We assessed the effectiveness of our contributions by creating multi-contact locomotions in a number of test environments by using only a few motion clips for climbing. While the key components of our contributions can be naturally applied to any motion planning frameworks based on a randomized search, we implemented the greedy best first search method [Russell et al. 2003] that explores the environment in a greedy manner. In particular, we first perform contact planning that finds a contact sequence for reaching the target, and then carry out the local planning that creates the actual movement of the character.

The remaining part of the article proceeds as follows. After reviewing related researches to our work in Section 2, we first present a method to modify a motion clip in Section 3. Section 5 details the contact graph and the procedure to create it, and Section 6 shows how the contact graph is used to generate multi-contact locomotion. We investigate the effectiveness of the presented method through a number of experiments in Section 7 and discuss its properties, limitations, and future work in Section 8.

2. RELATED WORK

Generating a virtual character's motions to perform given tasks by interacting with an environment is a central problem in computer animation research. Thus, numerous approaches have been developed in this regard. In this section, we review approaches closely related to our proposed method for creating whole-body multi-contact locomotion.

Motion Graph: By defining valid transitions between motion clips acquired by motion capture devices, motion graph-based methods efficiently sequence motion clips to create realistic motions that accomplish given tasks while satisfying constraints such as collision avoidance [Kovar et al. 2002; Lee et al. 2002]. Including more motion clips and increasing their connectivity in the graph expands the range of producible motions but also increases memory usage and search time. Thus, researchers have developed techniques to increase motion generation capability while suppressing the complexity of the motion graph. An effective method is to construct a motion graph to support the interpolation of motion clips [Safonova and Hodgins 2007]. Parameterizing motion clips with respect to their spatial features (e.g., location of foothold) allows for the efficient creation of motions by blending motion clips per new input parameter [Wiley and Hahn 1997; Kovar and Gleicher 2004; Shin and Oh 2006]. A randomized search around a motion clip can adapt to new constraints while preserving the original style [Shapiro et al. 2007]. More complex mobile-manipulation behaviors can be generated by combining heterogeneous motion-generation methods such as motion graph-based and algorithmic approaches [Mahmudi and Kallmann 2015].

We also extend the range of reproducible motions by randomly sampling target contact points and modifying the motion clip. A novel aspect of our method is that we pre-compute the range of possible motions obtained from a motion clip by representing them with the probability distribution of the contact points. By sampling points with high probability, our method increases the chance to create natural motions.

Complex Environment: A cluttered or dynamic environment adds to the difficulty in motion generation as one needs to deal with collisions with the environment or moving obstacles. In this case, extracting contact-related information from the motion and the environment helps to find motions applicable to the given environment [Kapadia et al. 2016]. If a suitable motion sample is given for an environment, then motion editing techniques can generate new motions of a similar style for a different body scale or a modified environment shape [Ho et al. 2010; Lee et al. 2006].

Randomized search algorithms such as a probabilistic roadmap (PRM) planner or a rapidly exploring random tree (RRT) have shown good performance for the problem of planning motions of high-degrees-of-freedom robots [Latombe 1991; Choset et al. 2005]. Combined with motion samples, randomized search algorithms can produce realistic character locomotion [Choi et al. 2003] as well as hand manipulation [Koga et al. 1994; Yamane et al. 2004; Ye and Liu 2012]. Real-time locomotion against the dynamic obstacles has been achieved by constructing and searching for high-level behavior graphs [Lau and Kuffner 2005] or by training controllers with reinforcement learning [Levine et al. 2011].

For a severely cluttered environment, deforming motion samples may be necessary on top of path planning. By extending the motion editing technique in Kim et al. [2009], Choi et al. [2011] developed a 3D Laplacian motion editing method to modify a motion clip to eliminate penetration while reaching target contact points. We adopt their Laplacian motion editing framework but develop a new variation of 3D Laplacian motion deformation. For the case of a large deformation of environment, Tonneau et al. [2016] developed a method to reposition original contact points to preserve physical relationship with the environment.

Multi-Contact Motion: While many motion planning methods concern the avoidance of collision with complex environments, the problem of multi-contact motion generation confronts the challenging task of generating motions that can make contact with arbitrary points on the surface of the environment with both the hands and feet. Researchers have developed randomized search-based methods for this problem [Escande et al. 2013; Bouyarmane and Kheddar 2011]. These methods generate sample candidate contact points on the environment and then validate the points through local planning. If the contact points and motions are sampled at random, then the resulting motion, although feasible, often looks unnatural. Hauser et al. [2008] achieved a more realistic motion and faster performance by sampling contact points and motions around motion primitives. Lengagne et al. [2014] solved semi-infinite programming to obtain optimal multi-contact motions.

We also follow the two-step *contact-before-motion* approach. The distinctive feature of our method lies in the contact planning step. By modeling the probability of contact points and estimating occupancy due to the contact points, our contact planner finds the contact points that are likely to be achieved by a character, even before going through local planning.

Multi-contact motions have also been studied in physics-based animation research. Liu et al. [2010] achieved multi-contact character motion such as rolling by sampling parameters for the controller. Mordatch et al. [2012] showed that discrete motions such as multi-contact locomotion can be created with space-time optimization by involving continuous variables that indicate contact with a particular end effector and the environment. Jain et al. [2009] proposed an optimization-based framework that can generate various interacting movements responsive to dynamically varying environment such as climbing, swinging, and leaning.

3. MOTION DEFORMATION

Since the motion deformation method is a basic tool used in our multi-contact locomotion generation framework, we discuss it first in this section.

The Laplacian mesh editing method has proven to be effective in modifying motions as well [Kim et al. 2009], but applying the Laplacian mesh editing to a 3D motion mesh may result in undesirable results due to its linear transformation nature. For 3D Laplacian motion editing, Choi et al. [2011] proposed a method that applies 2D Laplacian editing separately in horizontal and vertical planes. This method is effective for cases where the complete poses at the initial and final frames serve as boundary constraints. However, in our case, the constraints need to be given only for the end effector positions, making it necessary to develop a different method. Our approach for this is to first find the end effector paths by using Laplacian curve editing and then to find paths for the whole joints by using Laplacian mesh editing with the end effector paths as boundary constraints.

Figure 1 compares our method with a motion mesh obtained by applying the Laplacian editing to the whole 3D mesh (dubbed the general 3D Laplacian motion editing in this article). When the target final position of the right hand is changed (green arrow) for an original motion clip (Figure 1(b)), the general 3D Laplacian motion editing method (Figure 1(a)) fails to retain the rigidly-deforming nature of Laplacian editing and generates a steep arm rotation near the end frame (red line). The non-uniform edge lengths of the hand trajectory shown in the rectangle mean the general 3D Laplacian motion editing may create a large variation in motion speed. We suspect that this is largely due to the structure of the motion mesh that is different from the typical 2D manifold mesh for geometric modeling. In contrast, our method better preserves the style of the original motion as shown in Figure 1(c). Figure 2 shows more examples obtained by the proposed motion deformation method.

Note that this two-step approach may bring additional benefits for the randomized motion search approach: Since the end effectors have a higher chance to penetrate into the environment than other body parts, it should be computationally more efficient to search for collision-free end effector paths first before dealing with the entire joints.

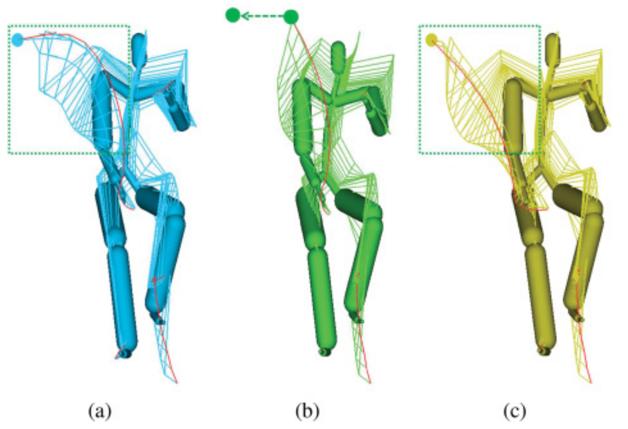

Fig. 1. (a) Laplacian deformation with point constraints. (b) Original motion. (c) Two-step Laplacian deformation (our method). Solid circles represent the new target contact points for the right hand.

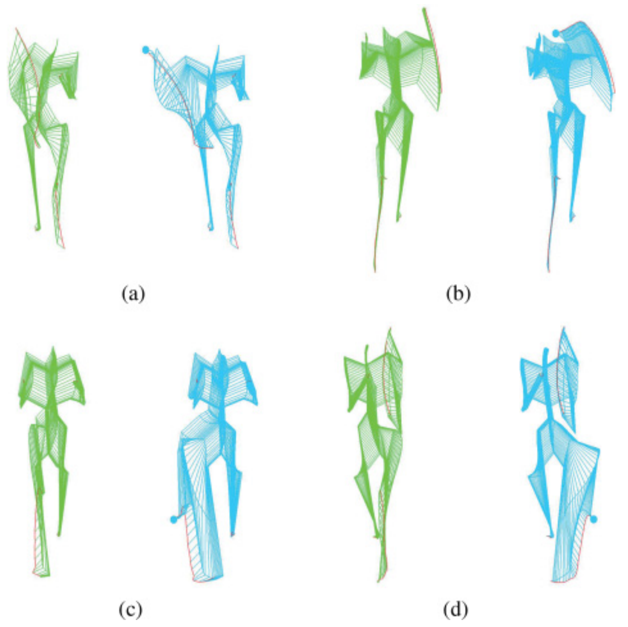

Fig. 2. More examples of our motion deformation method. The original mesh and its deformed mesh are shown in green and blue, respectively. Blue dots indicate the new target contact points.

3.1 Laplacian Motion Deformation

We first construct a mesh M such that vertices are positioned at the joint locations at every frame, and the edges are created between temporally and hierarchically adjacent vertices. For instance, $v_i(t)$, a vertex for the joint i at time t, is connected to the vertices $v_i(t-1)$, $v_i(t+1)$, $v_{p(i)}(t)$, and $v_{c(i)}(t)$ with $p(i)$ and $c(i)$ indicating parent and child joints, respectively. Figure 1(b) shows an example mesh drawn in a wire frame.

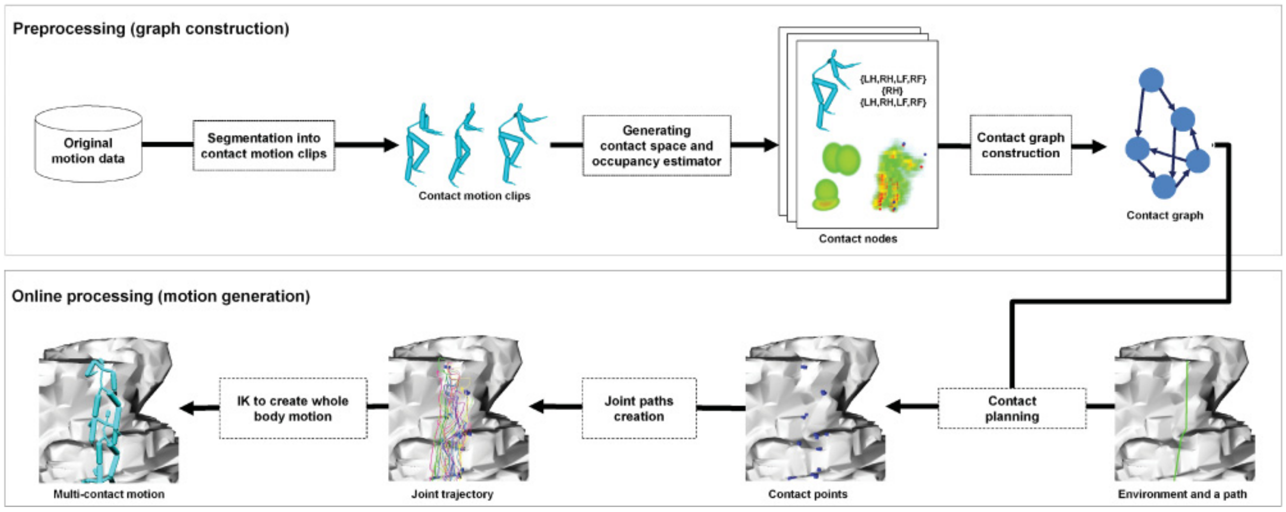
Preprocessing (graph construction)
Original motion data
Segmentation into contact motion clips
Contact motion clips
Generating contact space and occupancy estimator
{LH,RH,LF,RF}
{RH}
{LH,RH,LF,RF}
Contact nodes
Contact graph construction
Contact graph
Online processing (motion generation)
Multi-contact motion
IK to create whole body motion
Joint trajectory
Joint paths creation
Contact points
Contact planning
Environment and a path

Fig. 3. Overall flow of our proposed method.

The idea of Laplacian mesh deformation is to find the displacement of unconstrained vertices v_i such that their differential coordinates are preserved after deformation [Lipman et al. 2004]:

$$\delta v_i - \frac{1}{\deg(v_i)} \sum_{(i,j)\in M.\text{edge}} \delta v_j = 0, \tag{1}$$

where δv_i denotes the difference between the original and new positions of v_i. We can construct a linear system of Laplacian matrix by stacking Equation (1) for all free vertices v_i, and solve for v_i given the displacement of the constrained vertices.

End Effector Paths. Laplacian curve editing is performed to generate a new end effector path to reach the new contact point. Specifically, we set the boundary constraints specifying the displacement of the end-effector positions at the initial and final frames to satisfy the new contact points. We then find the positions of the vertices in the end effector paths such that they satisfy

$$\delta v_i(t) - \frac{1}{2}(\delta v_i(t-1) + \delta v_i(t+1)) = 0 \tag{2}$$

by solving a corresponding linear system of Laplacian matrix.

Internal Joint Paths. Next, paths of the whole-body joints are determined by deforming mesh M with the Laplacian mesh deformation method (Equation (1)) under the constraints specified by the end effector paths obtained from Equation (2). Examples of deformed meshes are shown in Figures 1(c) and 2.

3.2 Inverse Kinematics

Although the Laplacian motion editing method effectively solves for the joint paths satisfying the new contact points while preserving the original motion style, it does not take into account the bone length constraints. Therefore, we find the feasible motion by solving the inverse kinematics that best follows the joint paths. The Inverse Kinematics (IK) is performed at every frame by solving for joint coordinates q that minimize the following cost function:

$$\sum_i \alpha_i ||v_i^M(t) - v_i(q(t))||^2 + \lambda ||q(t)||^2, \tag{3}$$

where $v_i^M(t)$ denotes the desired position of the joint i at t obtained by Laplacian motion editing, and $v_i(q(t))$ is its position due to the joint coordinates $q(t)$. Larger values are given to α for the end effectors than to internal joints to give higher priority to the end effectors, and λ is the weight for the regularizing term. The Levenberg-Marquardt algorithm found the optimal joint coordinates $q(t)$ within 10ms per frame on average (Table II).

4. SYSTEM OVERVIEW

Our method consists of a preprocessing stage that constructs the contact graph from a set of multi-contact motion data and an online stage that generates multi-contact motion by following a given path in the complex environment. Figure 3 shows the overall flow of the proposed method.

In the preprocessing stage, we only deal with motion data without any information on the environment. In particular, the preprocessing stage segments the motion data into a set of motion clips based on contact configurations (Section 5.1). Each motion clip constitutes a node of the contact graph. We construct contact spaces that represent the contact points a motion clip can make (Section 5.2), and train the occupancy estimator to predict the space consumed by the character (Section 5.3).

In the online stage, when an environment and target path are given as input, a sequence of contact points is first planned such that it minimizes the cost of several criteria including contact feasibility, collision avoidance, and path following (Section 6.2). Next, whole-body motion is created as output by first planning joint paths by using Laplacian motion deformation and then performing inverse kinematics (Section 6.3).

5. CONTACT GRAPH

A contact graph $\mathbf{G} = (\mathbf{N}, \mathbf{E})$ is a directed graph with a set of nodes $\mathbf{N}$ and edges $\mathbf{E}$. We will call a node $n \in \mathbf{N}$ a *contact node* and it has the following attributes:

- Contact motion clip
- Contact configurations
- Contact space
- Occupancy estimator

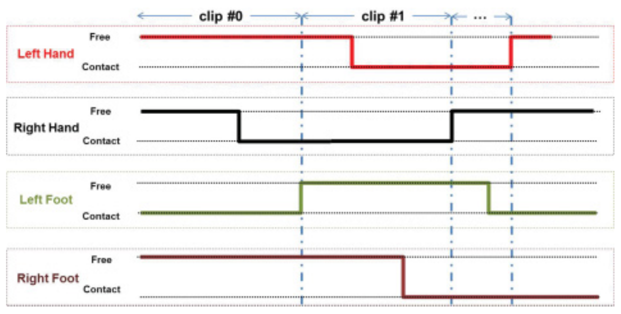

Fig. 4. Contact motion clips are segmented at the frame when the contact state of any one of four end effectors switch from *contact* to *free*. Clips #0 and #1 are divided when the character disengages contact at the left foot.

We will explain the properties of each attribute and the process to obtain them in the following subsections.

5.1 Contact Motion Clip

A contact motion clip is a motion clip acquired by segmenting a motion sequence according to the contact state of a character. The contact motion clip also serves as a unit to construct a node of the contact graph. Assuming that a character contacts the environment with only the hands and feet, we collect contact motion clips by using the following steps.

For every frame of input motion data, we determine the state of each end effector as either *contact* or *free* based on its velocity through the method presented in the Appendix. Then, we cut the motion data right before the frame in which any one of the end effectors goes through contact release (i.e., state changed from *contact* to *free*). Thus, every contact motion clip starts with a contact release of at least one end effector. The rationale for this choice is that a contact release can be considered as a cue to move toward a new target point.

In our experiment, we used a total of 4 motion samples for climbing motions collected from CMU (mocap.cs.cmu.edu) and Mixamo (www.mixamo.com) motion data libraries. The motion samples were segmented into 30 contact motion clips, among which we picked 18 clips that contained multi-contact motions. The lengths of motion clips ranged from 47 to 166 (104 on average).

Subsequently, each motion clip is automatically annotated according to its contact configurations. The annotation includes the contact configurations at the initial and final frames as well as the contact transition. The contact configuration includes a list of contacting end effectors at the initial and final frames, and the contact transition attribute stores the list of end effectors that have moved and contacted new points during the motion clip. Thus, the initial states of end effectors in contact transition can be either *free* or *contact*, but their final states must be *contact*. For instance, the clips #0 and #1 in Figure 4 will be annotated as follows:

- Clip #0
  —Initial contact configuration: {LF[1]}
  —Contact transition: {RH}
  —Final contact configuration: {RH, LF}
- Clip #1
  —Initial contact configuration: {RH, LF}
  —Contact transition: {LH, RF}
  —Final contact configuration: {LH, RH, RF}

[1] 1R: Right, L: Left, F: Foot, H: Hand.

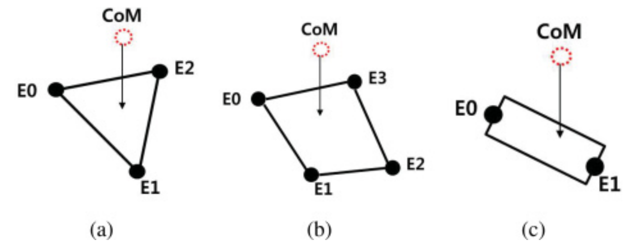

Fig. 5. For the stability check, contact points and center of mass are projected vertically to the level plane (ground) and verified if the projected CoM is included in the convex hull made by the projected contact points. Suitable width (length of the foot) is given to two-point contact (c). This stability check is valid regardless of the actual geometry of the environment except that friction is ignored.

From the annotation for the Clip #0, we can understand that LF is kept fixed at a point throughout the motion clip, RH is initially *free* but makes contact during the motion clip, and all other end effectors maintain a *free* state. For Clip #1, RH is kept fixed at a point, LF leaves a contact point, and both LH and RF make new contacts. Note that the final contact configuration of the previous motion clip is always the same as the initial contact configuration of the next motion clip.

We use the annotation not only to construct a contact graph, but also to modify the motion clip as explained later.

5.2 Contact Space

The contact space of a motion clip is the space of contact points that end effectors can make by modifying the motion clip. Of course, the size of a contact space increases as we allow for more modifications of a motion clip, but this may hurt the style of the motion clip. Thus, we create a contact space by sampling valid contact points near the contact point of the original motion clip. Specifically, we perform the following procedure repeatedly: We pick new candidate contact points for the end effectors included in the initial or final contact configurations by randomly sampling points within 50cm from the original points. Next, we perform motion deformation and verify whether the resulting motion is feasible by checking its kinematic and physical feasibilities. Balance check is performed by examining whether the ground projection of the center of mass (CoM) is included in the convex hull made by the ground projection of the contact points (see Figure 5). Joint torque limit is tested by computing the inverse dynamics against the static poses in the same way as in Kang and Lee [2014]. In principle, the feasibility check should be performed to every frame of the motion, but checking with only the initial and final frames was enough to sift out unnatural motions in our experiment.

We collect 300 motion samples for each motion clip, and construct contact spaces for each contacting end effector with the contact points obtained from the motion samples.[2] All these contact points are feasible, but we give higher priority to the points in the middle, because they require less deformation from the motion clip. Therefore, we model a contact space with a 3D Gaussian distribution

[2] Note that this is somewhat relaxed approach for collecting contact points. Exact method would be performing exhaustive examination such that, for a motion clip, any combinations of contact points in contact spaces per contacting end effector are guaranteed to be feasible.

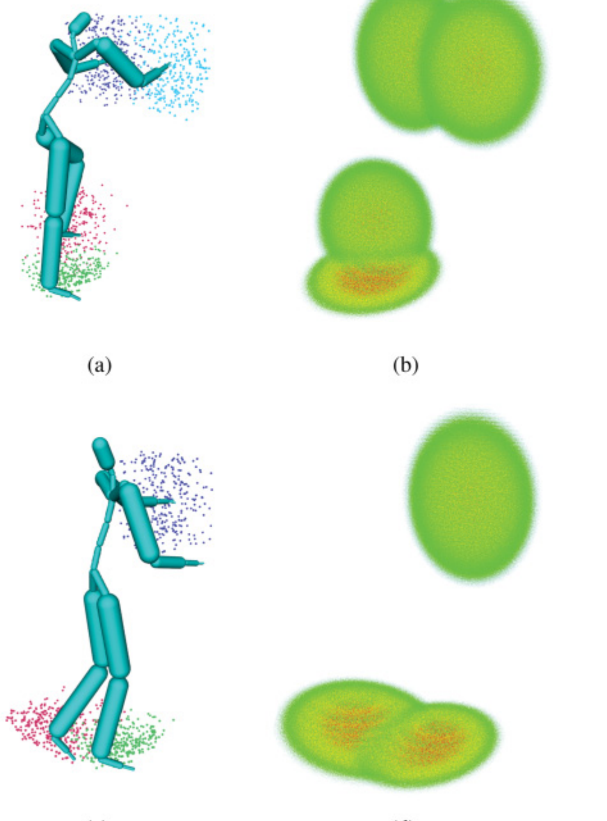

Fig. 6. Sampled contact points of a pose at a terminal frame (left) and the Gaussian distributions obtained from the points (right) for two example motion clips. Color visualizes probability density at the position: blue (0)→ red (1), with alpha value also set to the probability density.

$\mathcal{N}(\mu, \Sigma)$ in which the mean μ and the covariance Σ are computed from the sample points. Figure 6 shows the example distributions of the contact spaces. μ and Σ are expressed with respect to the reference frame of the motion clip.

5.3 Occupancy Estimator

An occupancy estimator estimates the space occupied by a character given target contact points. For this, we create an *occupancy grid* for each contact node such that the grid encloses every motion sample (Figure 7(a)). An occupancy estimator sets value $y_i \in [0, 1]$ to each voxel i as its spatial ratio occupied by a character,

$$y = f(x), \tag{4}$$

where the vector x represents the contact points of the initial and final configurations. The dimension of y is the number of voxels. For instance, $y_i = 0.3$ means that 30% of voxel i is swept by a character's body (Figure 7(b)). The occupancy value for a voxel is measured by subdividing the voxel into 10×10 sub-voxels and counting the number of sub-voxels that collide with any body part in any frame.

We use machine-learning algorithms to train an occupancy estimator. Once trained, the occupancy estimator will estimate the likelihood of collision with contact points at a certain location in

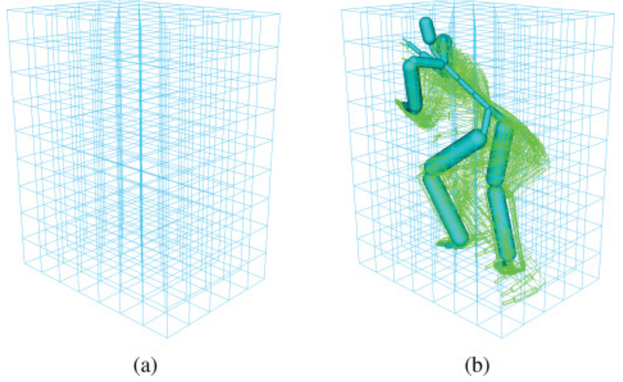

Fig. 7. (a) Occupancy grid that encloses all motion samples is created for each contact node. (b) Occupancy is measured as a swept volume by each motion sample.

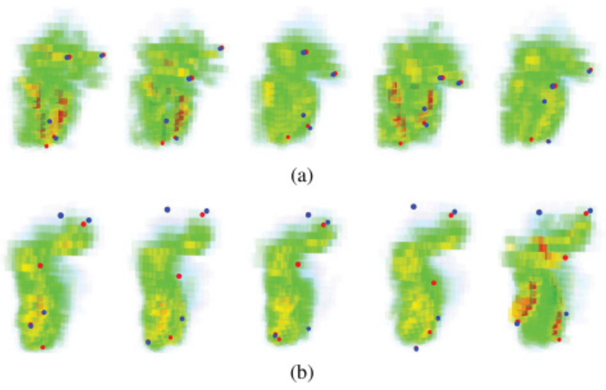

Fig. 8. Visualization of estimated occupancy for test contact points. Red and blue dots denote contact points at initial and final frames, respectively. (a) A motion clip with the right foot transition. (b) A motion clip with the left foot and right hand transitions.

the environment much faster than the local planning process. We train the occupancy estimator for each node of a contact graph. The dimension of y is normally several thousands, but voxel values are highly correlated with those of neighboring voxels. We reduce its dimension by using principal component analysis (PCA) and modeling the occupancy as follows:

$$y = P \cdot g(x) + \bar{y}, \tag{5}$$

where $\bar{y}$ denotes the mean of y, a matrix P includes the principal components of occupancy vectors, and $g(x)$ is the weight for every principal component. In our experiment, the number of principal components was set to 50.

We model $g(x)$ with a Gaussian process [Rasmussen 2004] but other function approximators, such as a radial basis function network, would work well. Specifically, we employ a noise-added squared exponential function as the kernel function of the Gaussian process. We use 100 sample motions collected for creating contact spaces to model $g(x)$ and the remaining 200 samples for validation. Figure 8 shows examples of occupancy values for the given contact points.

Table I shows the error rates of the motion clips measured as Equation (6). The second column shows the average error rate for all voxels, and the third column shows the average error rate for the

Table I. Error Rate of Learning Occupancy Space

Contact motion clip	All voxels	Non-zero voxels only (PCA error)
average	0.0170	0.1145 (0.0568)
std. dev.	0.0023	0.0243 (0.0140)

voxels with non-zero values only. The numbers in parenthesis are the average reconstruction errors of the 100 sample motions due to the dimensionality reduction using PCA. The third row shows the standard deviation of the errors. Table I shows that our method estimates the occupancy values with reasonably low error rates, 0.017 on average.

$$Error\ rate = \frac{\sum_{i=1}^{N} |y_i - y_i^{true}|}{N} \tag{6}$$

Memory consumption: For the contact spaces, we store only the mean and covariance matrix for each contact point in each node; thus, the memory consumed by the contact spaces is negligible. For the occupancy estimator, P and $\bar{y}$ for PCA, and the training data and hyper parameters for the Gaussian process are stored. $P \in \mathbb{R}^{S\times 50}$ and $\bar{y} \in \mathbb{R}^{S}$, where S is the grid size ranging from 4,000 to 5,700 in our experiment. For storing the training data, we store (x_i, g_i) $(i = 1 \ldots 100)$, where the dimension of x_i is 3×(# of contact points) and $g_i \in \mathbb{R}^{50}$. Therefore, roughly 256K numbers are stored in each node for the occupancy estimator.

5.4 Graph Construction

So far, we have discussed the attributes of a node constituting a contact graph. We construct a contact graph as a directed graph by creating an edge $n_i \rightarrow n_j$ if the following requirements are satisfied:

(1) Final contact configuration of n_i is the same as the initial contact configuration of n_j.
(2) The contact transitions of n_i and n_j are disjoint.

The second condition prevents the repetitive transition of the same body part. For instance, if the right hand has changed the contact point, then we do not allow it to take another contact point in the right next step. Note that when we create edges, we only consider the contact configurations without dealing with pose similarities between the motion clips. This was done to increase the connectivity of the graph as the number of nodes in our graph is rather small. A larger contact graph would benefit from considering the pose similarity for edge construction.

6. ONLINE MOTION GENERATION

Our method for generating multi-contact motion consists of two stages. Given the environment and a path to follow, contact planning is first performed to find the sequence of contact nodes and their contact points to reach the goal. Contact space and the occupancy estimator are used in this stage to find good contact points with high probability and less chance of penetration. Next, whole-body motion is generated to realize the planned contact points.

Detailed explanation of the method will be given after presenting how we process the environment data for efficient motion planning.

6.1 Environment Processing

In order to test collision and measure penetration efficiently between the character and the environment, we make it possible to compute the signed distance of a point to the nearest surface by creating an *environment grid*, a 3D grid enclosing the environment. Given the surface meshes for the environment, we densely sample points on the surfaces and find *surface voxels* that include the points. Each surface voxel is given a normal direction estimated from the mesh.

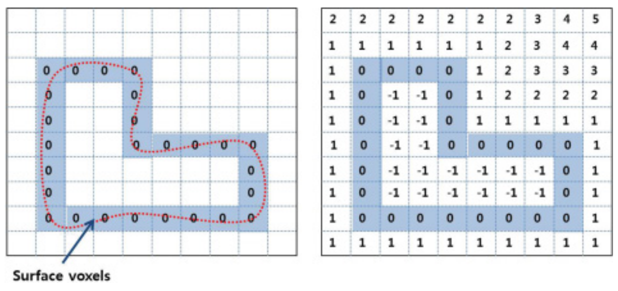

Fig. 9. Environment grid and signed distance field.

For each voxel x_i we compute its signed distance $sd(x_i)$. For this, we first set the signed distance of surface voxels to zero and iteratively fill the signed distances of adjacent voxels by incrementing (decrementing for interior voxels) its distance by 1. Figure 9 shows an example of the environment grid and the signed distance field.

6.2 Contact Planning

Our motion generation starts with contact planning, in which we find the optimal sequence of contact nodes and contact points to follow the target path. The essential element for motion planning is an appropriately defined energy (or cost) function of a motion unit with respect to a particular position in the environment. Using the energy function, motion planners find an optimal motion sequence that minimizes the total energy. For a contact node n with the transformation matrix $T \in SE(3)$ and a set of contact points $C = \{c_1, \ldots, c_N\}$ $(c_i \in \mathbb{R}^3)$, we define its energy as follows:

$$E(T, C) = w_1 f_n + w_2 f_c + w_3 f_p, \tag{7}$$

$$f_n = 1.0 - \frac{1}{N}\sum_{i=1}^{N} \mathcal{N}(T^{-1}c_i|\mu_i, \Sigma_i), \tag{8}$$

$$f_c = \sum_{i \text{ for } sd(x_i)<0} |sd(x_i) \cdot y(T^{-1}x_i)|, \tag{9}$$

$$f_p = dis_p + dis_d, \tag{10}$$

where w_i are the weights and N is the number of contacts. $y(T^{-1}x_i)$ denotes the occupancy value that corresponds to the position x_i of the environment voxel i.

f_n contributes to the increase in the probability of contact points. f_c measures the severity of penetration of the motion for the contact points. For this, we first compute the occupancy for the input contact points by using Equation (5) and find the occupancy voxels that collide with the interior voxels of the environment grid. f_c is measured as the sum of these occupancy values $y(T^{-1}x_i)$ weighted by their penetration depths $sd(x_i)$. f_p measures how well a contact node follows the target path. For this, we identify a local target point as a path point 1m from the current position. dis_p is the distance from the center of contact points to the local target point, and dis_d is the angle between the direction to the local target point and the moving direction of the character, which is estimated by the center of the contact points. We empirically set $w_1 = 0.25$, $w_2 = 0.5$, and $w_3 = 0.25$.

6.2.1 *Optimal Contact Points.* As Equation (7) is generally not a continuous function for a non-continuous environment, we use Covariance Matrix Adaptation Evolution Strategy (CMA-ES), a derivative-free optimization algorithm, for finding an optimal T and C that minimize Equation (7). More specifically, as the optimal C largely depends on T, we perform two-step optimization: The CMA-ES algorithm performs sampling on T, and for each sample T, we determine the optimal C by using Algorithm 1. The parameters of the CMA-ES algorithm are set as (population size = 100, standard deviation = 1.0, maximum iterations = 100, stop fitness = 0.2).

ALGORITHM 1: Find Optimal Contact Points and Energy of a Node

```
Input: Transformation matrix $T$
Output: $C$, $E$
 1: $C \leftarrow \varnothing$
 2: Transform Contact Spaces according to T
 3: for all contact space $\mathcal{N}(\mu_i, \Sigma_i)$ do
 4:     for all surface voxels $x_j$ near $\mu_i$ do
 5:         if IsValid($x_j$) = true then
 6:             Compute $f_{n,j} \leftarrow \mathcal{N}(x_j|\mu_i, \Sigma_i)$
 7:             if $f_{n,j} > f_{n,i}^{max}$ then
 8:                 $f_{n,i}^{max} \leftarrow f_{n,j}$
 9:                 $c_i \leftarrow x_j$
10:     $C \leftarrow C \cup \{c_i\}$
11: Compute $f_c$ and $f_p$ for $C$
12: $E \leftarrow w_1 f_n + w_2 f_c + w_3 f_p$
```

In our experiment, surface voxels within 80cm from the mean of each contact space are considered (line 4 of Algorithm 1). A surface voxel x_j is validated (line 5) if its surface normal is within 60° from the up-vector and is visible from the character. The visibility is examined by the intersection test between the environment and a line segment connecting the voxel and the head position at the initial frame of the motion clip.

In principle optimal C should be determined by considering all energies f_n, f_c, and f_p, but we determine the optimal C only with respect to the contact probability f_n and ignore the effects of f_c, f_p. This choice was made for computational performance, because repetitive call of occupancy generator may slow down the contact planner significantly. Therefore, the time complexity of finding the optimal contact points is proportional to the number of tested voxels and number of contacting end effectors.

6.2.2 *Contact Planning Algorithm.* We perform contact planning by using the energy function determined by Algorithm 1 to obtain an optimal contact sequence. An element Q of the contact sequence has the following attributes: $Q = \{n, T, C\}$, where n is the contact node, T is transformation matrix, C is the contact points ($C = C_{init} \cup C_{fin}$, where C_{init} denotes the contact points for the initial contact configurations and C_{fin} final configurations). While a globally optimal path could be obtained by well-known algorithms such as A* search algorithm, we verified the effectiveness of our technique by implementing the greedy best first search (GBFS) algorithm [Russell et al. 2003], which finds a locally optimal path. Note that the contact spaces and occupancy estimators are independent of search algorithms. They can also be used for the shortest path algorithms such as A* or Dijkstra's algorithms.

Algorithm 2 shows our pseudocode for the contact planning with GBFS, which finds a node's successor node in the increasing order of the energy in a recursive way. To find the very first node at the starting point of the path, we evaluate every node in the contact graph and find optimal T, C_{init}, and C_{tran} (contacts in contact transition). Except for the starting node for which both C_{init} and C_{tran} are unknown, we find only C_{tran} and set C_{init} to be C_{fin} of the previous node. The initial guess for the optimal T (line 7) is determined such that the distance from the mean of each contact space to the contact points obtained from the previous node is minimized. In our experiment, the threshold (line 9) was empirically set to 2.0.

ALGORITHM 2: NextBestContact

```
Input: Contact Graph $G$, User given path $P$, Current node $n$, Initial
    contact points $C_{init}$
Input: $Q$: current element from which to find remaining contacts
 1: if $Q.C_{fin}$ reached the goal then
 2:     $ContactSequence$.push_front($Q$)
 3:     return true
 4: $NodeList \leftarrow \varnothing$ {storage for the list of node}
 5: for all $n_i \in \mathbf{N}$ adjacent to $Q.n$ do
 6:     $Q_i.n \leftarrow n_i$
 7:     Find initial $Q_i.T$ using $Q.C_{fin}$
 8:     Run CMA-ES to find optimal $Q_i.T$, $Q_i.C_{tran}$, $Q_i.E$
 9:     if $Q_i.E <$ threshold then
10:         $NodeList$.push_back($Q_i$)
11: Sort $NodeList$ in increasing order of $E$
12: while $NodeList$ is not empty do
13:     $Q_c \leftarrow NodeList$.pop_front()
14:     Result = NextBestContact($Q_c$)
15:     if Result = true then
16:         $ContactSequence$.push_front($Q$)
17:         return true
18: return false
```

6.3 Contact Motion Generation

We have obtained a sequence of contact nodes, their transformation matrices, and contact points from the contact planning stage. Next, the motion generation stage computes the whole-body motion of a character that follows the contact sequence. The motion deformation process in Section 3 is used again for this purpose, with some augmentations to the Laplacian motion deformation as explained now. Recall that in the preprocessing stage Laplacian deformation is performed under constraints specified by contact points. In the online stage, to ensure the smooth transition of joint paths across contact changes, we impose additional constraints such that the joint positions at the initial frame of a contact node is the same as their final positions in the previous contact node.

Note that the contact planning stage selects the contact node that is less likely to penetrate into the environment, but still it is not guaranteed. Therefore, we examine whether the joint paths penetrate the environment after Laplacian motion deformation, and if so, we add a step to resolve the penetration. Choi et al. [2011] presented an effective solution that iterates Laplacian motion editing by adding constraints for penetration elimination. Similarly, we take an iterative collision resolving approach. At each iteration, we find the deepest penetrating joint path and obtain a modified collision-free path that is close to the original path (Figure 10). For the path modification, we identify two control points that equally divide the path into three segments and solve the Laplacian path editing under additional position constraints specified by the new positions of the control points. We employ the CMA-ES algorithm for finding the optimal positions of the control points, with parameters set as (population size = 11, standard deviation = 5.0, maximum iterations = 100, stop fitness = 1.0). Once the collision-free path is

Table II. Compute Time for Each Experiment

	# environment voxels	voxel size (cm)	# contact nodes	# frames	contact planning/ node (s)	joint path/node (s)	IK/frame (ms)	total time (s)
Cliff	233K	11.2	7	911	2.7	0.01	8.4	26.6
Tree	2231K	8.4	16	1989	4.7	0.1	9.2	93.0
Rings	813K	3.3	10	1114	20.8	3.2	9.8	250.9
Tunnel	126K	11.2	17	1884	3.4	7.4	7.6	198.2

Contact planning/node: average time for finding optimal contact points per node, measured for the nodes in the found optimal path. Joint path/node: average time for generating joint paths given the contact points, including the collision resolving step. IK/frame: average time for solving inverse kinematics per frame.

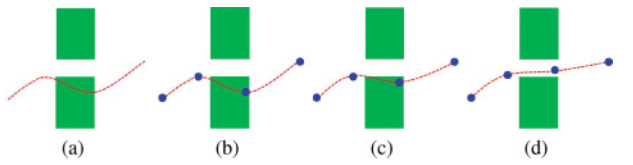

Fig. 10. Our collision resolution scheme. If a joint path (red) penetrates into an environment (a), then two control points (blue dots in the middle) are selected for the Laplacian path editing. Initial and final points are fixed and given as boundary constraints (b). The CMA-ES algorithm finds optimal positions for the control points that make a collision-free joint path as close as possible to the original path ((c) and (d)).

obtained, full-joint paths are recomputed with this resolved path fixed, and collision check is performed again to find next penetrating joint path. Finally, the inverse kinematics in Section 3.2 is performed to create the actual movement of the character.

7. EXPERIMENTS

We assessed the effectiveness of our method for generating multi-contact motions in several environments that have complex surfaces or narrow passages. The total DoFs of the character is 96 with each joint modeled as a ball joint.

Complex Environment Surface

Figure 11 shows each step of online motion planning. Given the environment and a target path (Figure 11(a)), the motion generator searches for a sequence of contact nodes to follow the path. Figure 11(b) shows the contact spaces of the contact sequence in different color per end effectors. Red circles denote the positions of the means of Gaussian distributions. Figure 11(c) shows the occupancy values at each voxel estimated by the occupancy estimator, given the contact points shown in Figure 11(d). Figures 11(e) and (f) show the joint paths obtained by the motion deformation process and the final motion after performing inverse kinematics. The appropriateness of the contact points and the naturalness of the resulting motions are better evaluated by watching the accompanying video.

Figure 12 shows another example of complex environmental surfaces. The input path is quite challenging to follow closely using the limited range of motion clips in our experiments. Our method finds a motion that successfully climbs up a tree while allowing some distance from the input path.

Since our motion graph contains only moderately climbing motions and the motion planner gives high preference to the contact points near the mean of the contact spaces, making extreme motions such as fully extending the limbs would be difficult to achieve, which is often necessary to climb a challenging cliff. A straightforward way to generate such motions in our framework would be to expand the motion graph to contain far-reaching motion clips.

Narrow Passage

Figure 13 shows an example that initial joint paths obtained from Laplacian motion deformation penetrated into the environment (Figure 13(b)) while the character passes through a hole. When penetration is detected, the joint paths are further adjusted to remove the collision as shown in Figure 13(c).

To investigate the extent to which the initial paths can be modified by our collision handling method, a climbing motion was created in a tapered cave as shown in Figure 14. The height of the cave could be reduced to approximately 30% of that required by initial paths given by Laplacian motion deformation.

Figure 15 shows a motion generated in an environment with non-continuous environment surfaces and a narrow passage. Recall that we checked for visibility in the validity test for a contact point (Algorithm 1, line 5), which was critical for an environment like this where the contact points on unreachable surface of the environment (e.g., outer surface of the rings) could otherwise be confirmed valid.

Table II shows the complexity and the computation time for each test shown in the accompanying video. As can be seen from the table, the motion planning time is affected by the voxel size, collision objects near the given path, and the path length. We performed the experiments on a desktop computer with an Intel Core i7 at 3.50GHz CPU. We implemented the method in a single-threaded program, and there is much room for speed improvement by parallelizing the program.

Note that the time for contact planning in a "rings" environment (Figure 15) is significantly larger than other test cases because its voxel size is smaller, and thus, more voxels are checked by the contact planner. Increasing the voxel size improves contact planning time yet decreases the accuracy of the collision test.

Comparisons

In order to validate the usefulness of the contact spaces and occupancy estimators, we compared four scenarios in an environment with obstacles as shown in Figure 16.

Figure 16(a) shows the joint paths and the character poses created by our method that uses both the contact spaces and occupancy estimators. Given the collision-inducing input path, our method generates a motion that keeps enough distance from the obstacles while retaining the naturalness of the original motion clips. In the second scenario that removes the effect of the contact spaces, we randomly select contact points instead of running Algorithm 1, which finds contact points near the mean of contact spaces. A candidate pair of T and C are evaluated by Equation (7) with f_n term removed. Random selection of contact points decreases the planning time, but the resulting motion shows large deviations from the original motion clips as shown in Figure 16(b). Figure 16(c) shows the case that only the contact spaces are used but the occupancy estimators and collision resolution are omitted. The resulting motion inevitably collides with the obstacles. Figure 16(d) is the case that collision check is performed for each sample contact points instead of using

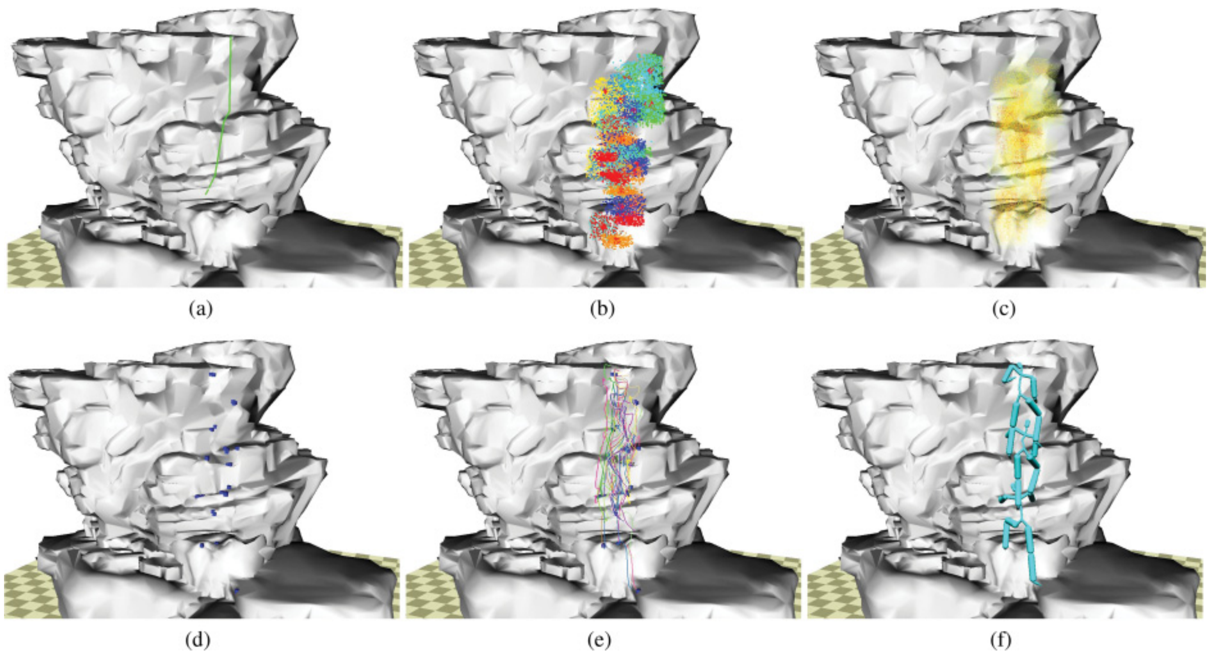

Fig. 11. Climbing up a cliff. (a) Environment and input path. (b) Contact spaces of selected nodes differently colored per end effector. Red circles denote the mean of contact spaces. (c) Occupancy values. (d) Contact points. (e) Joint paths. (f) Whole-body motion after inverse kinematics.

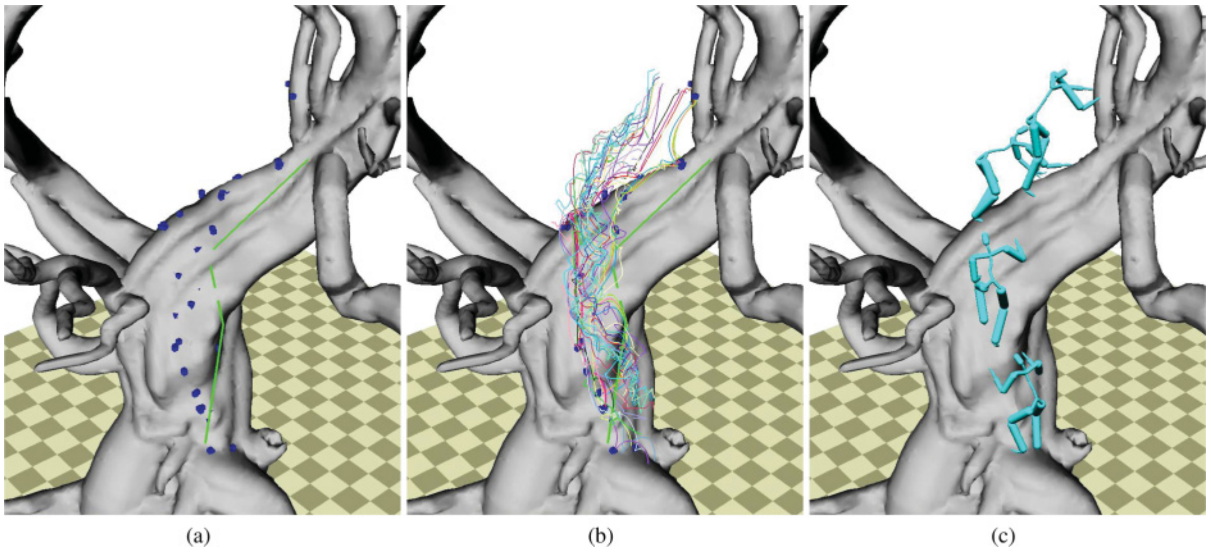

Fig. 12. Tree climbing example. (a) Contact points found for the input path (green). (c) Joint paths. (d) Final whole-body motion.

the occupancy estimator. For randomly selected contact points, we generate joint paths to reach the contact points using the Laplacian motion deformation and then check collision between the joint paths and the environment. The resulting motion shows smaller deviation from the input path while avoiding collisions with the obstacles. However, this is achieved at much higher computational cost than our method. Table III reports compute times for each method. The second column shows the average time taken for contact planning per node, of which the time consumed for collision checking is shown in the third column.

Effect of the Voxel Size

The voxel size largely depends on the geometric details of the environment. In our experiment, the voxel size in each experiment was determined by the fine details of each environment, for example, pillars in Figure 16, branches in Figure 12, and the gap between the rings in Figure 15.

To investigate the effect of the voxel size on the compute time and accuracy of occupancy estimators, we varied the voxel size for the tunnel environment (Figure 13) and measured them for a single

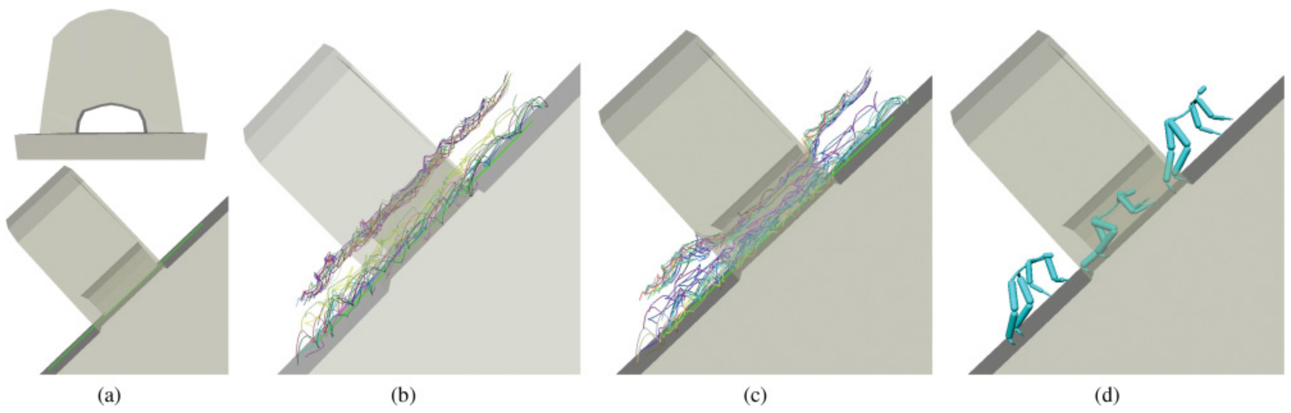

Fig. 13. Passing through a hole. (a) Side and top views of target environment. (b) Joint paths before collision resolution. (c) Joint paths after collision resolution. (d) Final motion.

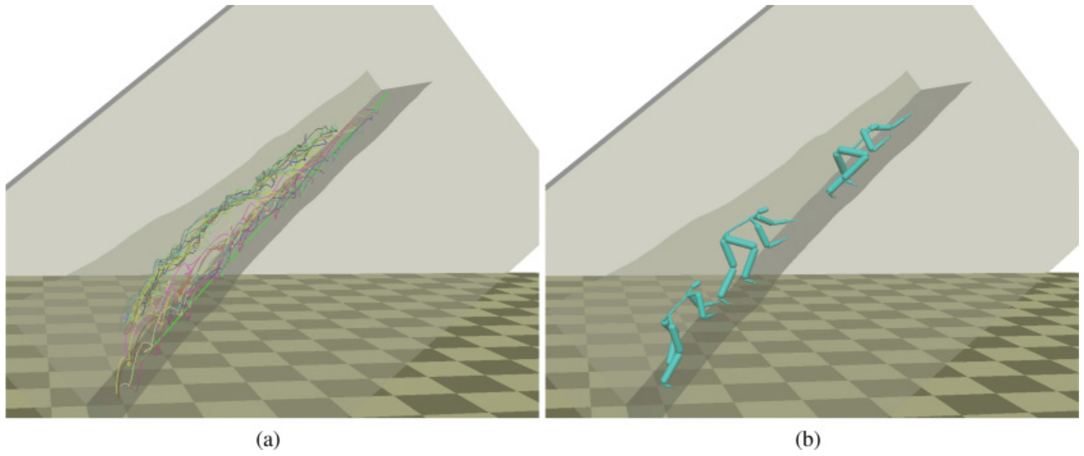

Fig. 14. Our method could find joint paths for a tapered cave of which the final height is only about 30% of the height needed by the original motion clip.

Table III. Compute Times of Compared Methods in Figure 16

	Number of contact nodes	Time for contact planning/node	Time for collision check/node
(a)	13	3.7s (CS)	0.3s (OE)
(b)	16	0.5s (RS)	0.3s (OE)
(c)	9	3.3s (CS)	—
(d)	15	104.3s (CS)	100.9s (CC)

CS: contact spaces, RS: random sampling, OE: occupancy estimators, CC: per-sample collision check.

node (motion segment) in the middle of the tunnel. The error rate of the occupancy estimator is measured by the average occupancy value of those occupancy grid cells that are incorrectly identified as penetrating the environment. Figure 17 shows the test result. Increasing the voxel size decreases the contact planning time, since fewer surface voxels are tested. For a flat surface, halving the voxel size increases the number of surface voxels by a factor of four. The error rate of occupancy estimator increases with the voxel size because a larger surface voxel includes more empty space, making more occupancy grid cells in the air be diagnosed as colliding with the environment.

8. DISCUSSION AND FUTURE WORK

We introduced a method for generating the multi-contact motion of a character in a complex environment. The key elements of the method are the contact space and occupancy estimator that allow for the estimation of the quality of a contact node and its contact points at a particular location in the environment before performing local planning. Our approach can be understood as an effort to enable the contact planner to know whether a candidate contact point is feasible or not without actually trying to reach the points. The effectiveness of this approach has been demonstrated by the experiments. Here we discuss the limitations of our proposed method and the possible directions for improvement.

For the experiments, we constructed a small contact graph that included mostly climbing motions because there are not many multi-contact motions available. The proposed method is, of course, not restricted to climbing motions, and we plan to expand the contact graph and generate other kinds of multi-contact locomotions.

We implemented the two-step motion generation in which the contact sequence to reach the goal is first planned and the whole-body motion is generated next. However, the contact sequence can

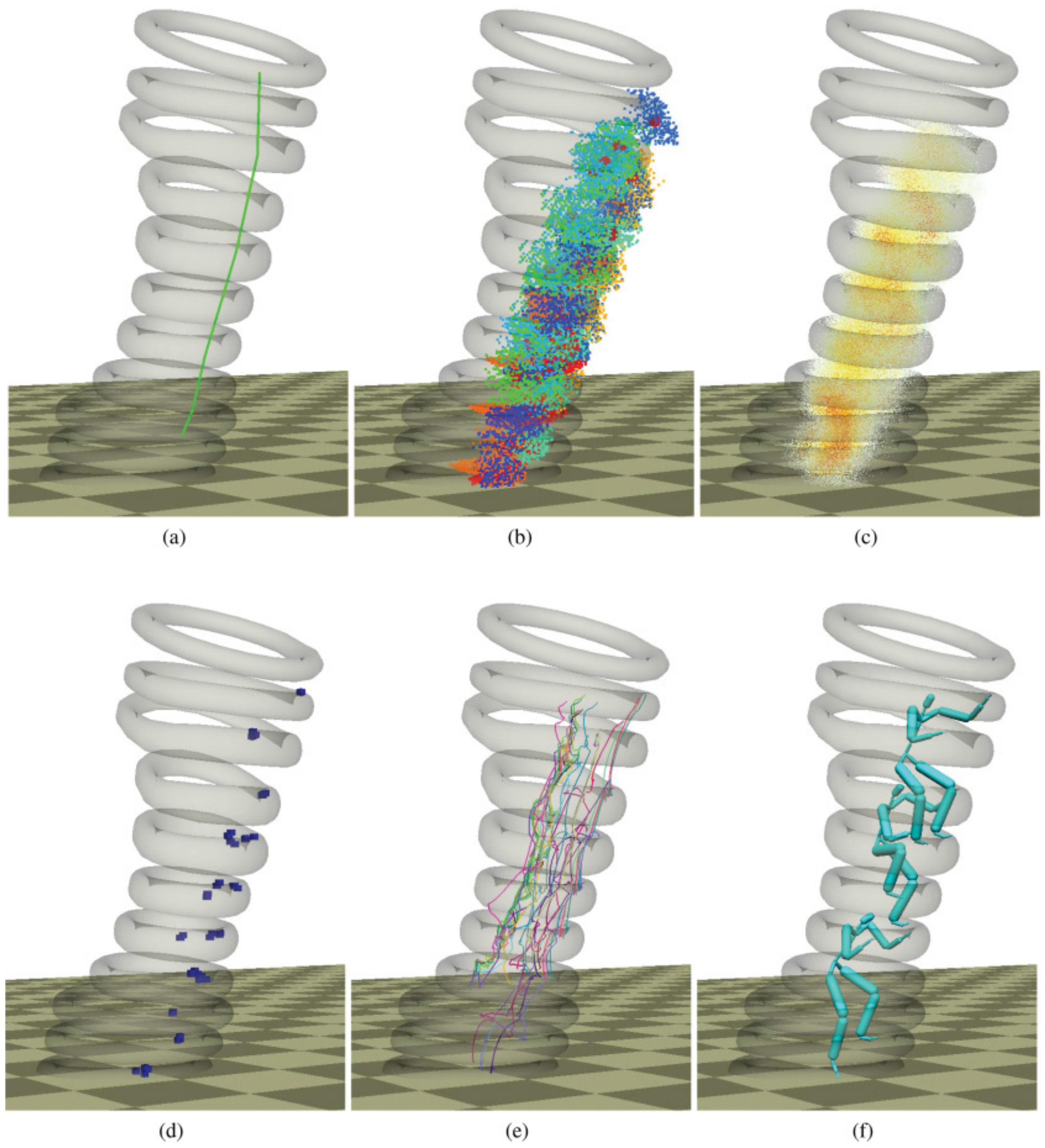

Fig. 15. An example of narrow passage. (a) Environment and input path. (b) Contact spaces of selected nodes differently colored per end effector. Red circles denote the mean of contact spaces. (c) Occupancy values. (d) Contact points. (e) Generated joint paths. (f) Whole-body motion after inverse kinematics.

turn out to be infeasible during the motion generation step, in which case we need to return to the contact planning step and find the next best contact sequence. We have not implemented this as it is not our major concern, but it can be done without difficulties by modifying the presented algorithms. In addition, dynamic feasibility and quality of the final motion will be improved if a dynamic optimization is included for the final motion generation stage.

Although we examined the physical feasibility of contact points when training contact spaces and occupancy estimators, the resulting motion obtained from the online motion generation is not guaranteed to be physically feasible. Recall that our method for balance check is static and does not take friction into account. Therefore, our method would have a high rate of failure for vigorously dynamic movements or for the environment with low friction such as on ice. For the applications where accuracy is important (e.g., motion planning of physical robots), the contact space needs to be improved to be able to consider dynamic balance and variable friction states.

We did not consider grasping as a means of locomotion. Grasping will enable a virtual character to move in a challenging environment with small footholds. To include grasping for the planning, the contact space needs to be extended to distinguish contact points for pushing and those for grasping. Balance check should also be adequately changed to take grasping into account. In addition, we

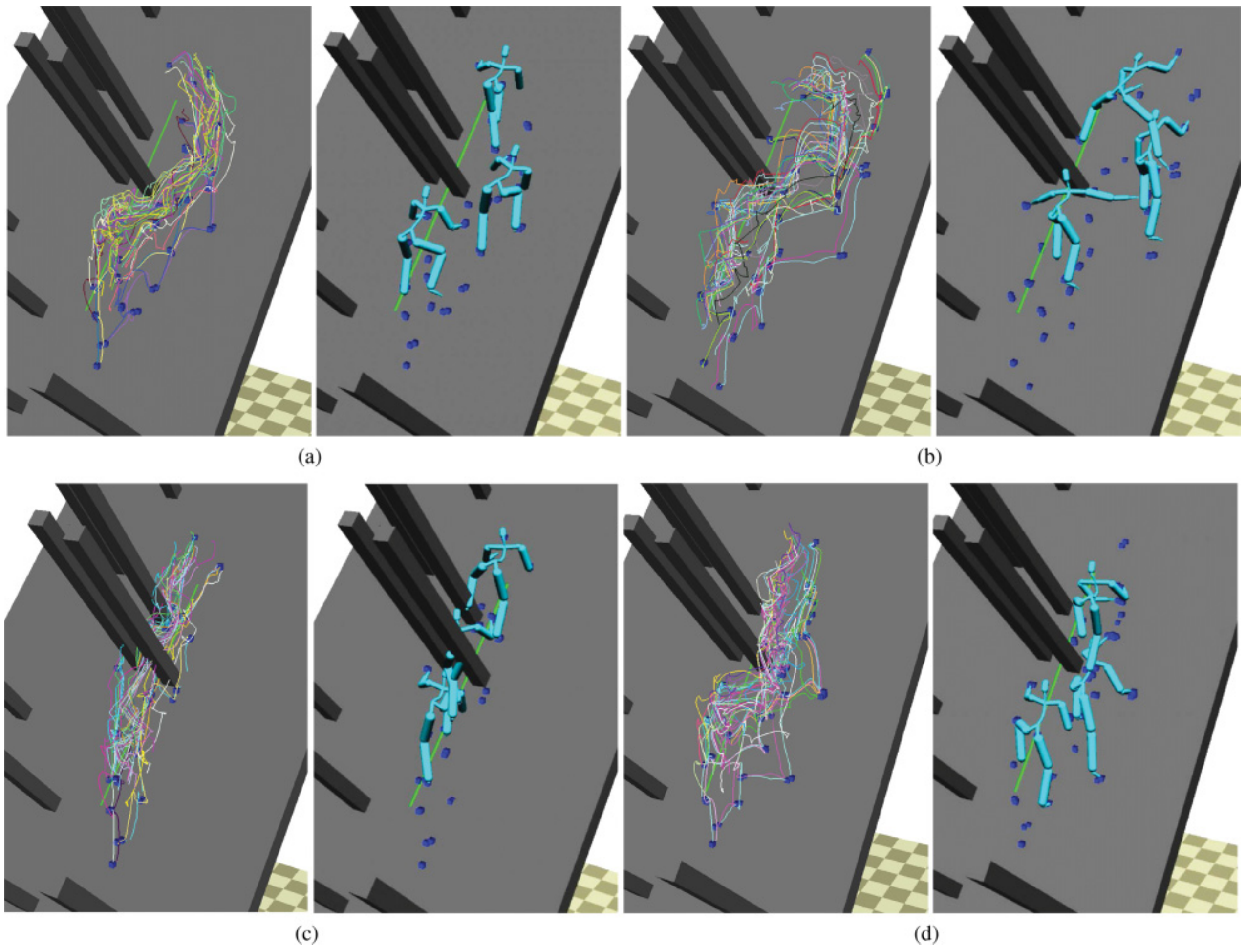

Fig. 16. Four compared cases. (a) Both contact spaces and occupancy estimators are used (our method). (b) Contact points are selected randomly and occupancy estimators are used. (c) Only contact spaces are used. (d) Contact spaces are used and collision check is performed for each sample instead of using occupancy estimators.

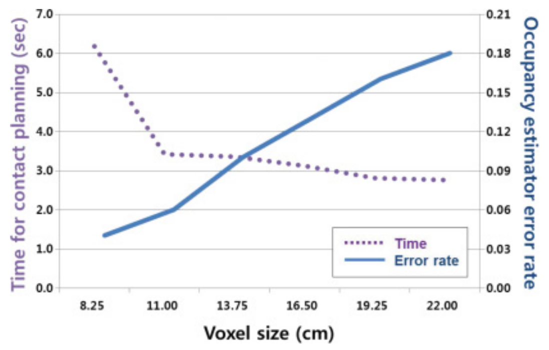

Fig. 17. Time for contact planning and error rate for occupancy estimator according to the voxel size.

assumed that only the end effectors make contact with the environment. Allowing for more body parts to contact with the environment will generate a larger repertoire of motions such as rolling and crawling with the elbows and knees.

Since the occupancy estimator outputs the occupancy values of a discretized space, its accuracy will degrade if an environment contains structures that are much thinner than the grid size of the occupancy estimator (e.g., jungle gyms). A solution to this problem would be to develop a continuous version of an occupancy estimator, which predicts the occupancy *density* at any 3D position against input contact points.

APPENDIX

A. EXTRACTING CONTACT MOTION CLIPS FROM MOTION DATA

In order to segment motion data into contact motion clips, as shown in Section 5.1, we need to determine the state of each end effector to be either *contact* or *free*. Estimating the contact state is not easy without any environment information. Thus, we make the task a little easier by assuming that the motion data of interest contains continuously moving motion and an end effector with the smallest speed is always in contact with the environment. State assessment can now be done by the speed of the end effectors, but this process is still generally not straightforward due to the high level of noise in the data. After several trials, we adopted the following procedure, which worked reasonably well in our experiment, but still requires some manual fixing.

First, we compute the speeds of the end effectors and denoise them with a low-pass filter. Then, at each frame, we divide the four end effectors into three groups with respect to the speed. The average speed of the slowest group sets the base speed of the contacting end effectors. Then, we compare with the average speed of the other groups. If it is less than twice of the base speed, then the end effectors in the group are also determined to be in a contact state. Otherwise, they are in a free state.

REFERENCES

Karim Bouyarmane and Abderrahmane Kheddar. 2011. Using a multi-objective controller to synthesize simulated humanoid robot motion with changing contact configurations. In *IEEE/RSJ Int'l Conf. Intelligent Robots and Systems (IROS)*. 4414–4419.

Myung Geol Choi, Manmyung Kim, Kyung Lyul Hyun, and Jehee Lee. 2011. Deformable motion: Squeezing into cluttered environments. *Comput. Graph. Forum* 30, 2 (Nov. 2011), 445–453.

Min Gyu Choi, Jehee Lee, and Sung Yong Shin. 2003. Planning biped locomotion using motion capture data and probabilistic roadmaps. *ACM Trans. Graph.* 22, 2 (Apr. 2003), 182–203.

Howie Choset, Kevin M. Lynch, Seth Hutchinson, George A. Kantor, Wolfram Burgard, Lydia E. Kavraki, and Sebastian Thrun. 2005. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press.

Adrien Escande, Abderrahmane Kheddar, and Sylvain Miossec. 2013. Planning contact points for humanoid robots. *Robot. Auton. Syst.* 61, 5 (2013), 428–442.

Kris Hauser, Timothy Bretl, Jean-Claude Latombe, Kensuke Harada, and Brian Wilcox. 2008. Motion planning for legged robots on varied terrain. *Int. J. Robot. Res.* 27, 11–12 (2008), 1325–1349.

Edmond S. L. Ho, Taku Komura, and Chiew-Lan Tai. 2010. Spatial relationship preserving character motion adaptation. *ACM Trans. Graph.* 29, 4 (July 2010), 33:1–33:8.

Sumit Jain, Yuting Ye, and C Karen Liu. 2009. Optimization-based interactive motion synthesis. *ACM Transactions on Graphics (TOG)* 28, 1 (2009), 10.

Changgu Kang and Sung-Hee Lee. 2014. Environment-adaptive contact poses for virtual characters. *Comput. Graph. Forum* 33, 7 (2014), 1–10.

Mubbasir Kapadia, Xu Xianghao, Maurizio Nitti, Marcelo Kallmann, Stelian Coros, Robert W. Sumner, and Markus Gross. 2016. Precision: Precomputing environment semantics for contact-rich character animation. In *Proceedings of the 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. ACM, 29–37.

Manmyung Kim, Kyunglyul Hyun, Jongmin Kim, and Jehee Lee. 2009. Synchronized multi-character motion editing. *ACM Trans. Graph.* 28, 3 (Jul. 2009), Article 79, 9 pages.

Yoshihito Koga, Koichi Kondo, James Kuffner, and Jean-Claude Latombe. 1994. Planning motions with intentions. In *SIGGRAPH'94*. 395–408.

Lucas Kovar and Michael Gleicher. 2004. Automated extraction and parameterization of motions in large data sets. *ACM Trans. Graph.* 23, 3 (Aug. 2004), 559–568.

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

J.-C. Latombe. 1991. *Robot Motion Planning*. Kluwer Academic Publishers.

Manfred Lau and James J. Kuffner. 2005. Behavior planning for character animation. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA'05)*. ACM, New York, NY, 271–280.

Jehee Lee, Jinxiang Chai, Paul Reitsma, Jessica Hodgins, and Nancy Pollard. 2002. Interactive control of avatars animated with human motion data. *ACM Trans. Graph.* 21, 3 (July 2002), 491–500.

Kang Hoon Lee, Myung Geol Choi, and Jehee Lee. 2006. Motion patches: Building blocks for virtual environments annotated with motion data. *ACM Trans. Graph.* 25, 3 (July 2006), 898–906.

Sbastien Lengagne, Joris Vaillant, Eiichi Yoshida, and Abderrahmane Kheddar. 2014. Motion planning with sequential convex optimization and convex collision checking. *Int. J. Robot. Res.* 33 (2014), 1251–1270.

Sergey Levine, Yongjoon Lee, Vladlen Koltun, and Zoran Popović. 2011. Space-time planning with parameterized locomotion controllers. *ACM Trans. Graph.* 30, 3 (May 2011), Article 23, 11 pages.

Y. Lipman, O. Sorkine, D. Cohen-Or, D. Levin, C. Rossi, and H.-P. Seidel. 2004. Differential coordinates for interactive mesh editing. In *Proceedings of the Shape Modeling Applications, 2004*. 181–190.

Libin Liu, KangKang Yin, Michiel van de Panne, Tianjia Shao, and Weiwei Xu. 2010. Sampling-based contact-rich motion control. *ACM Trans. Graph.* 29, 4 (July 2010), Article 128, 10 pages.

Mentar Mahmudi and Marcelo Kallmann. 2015. Multi-modal data-driven motion planning and synthesis. In *Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games*. ACM, 119–124.

Igor Mordatch, Emanuel Todorov, and Zoran Popović. 2012. Discovery of complex behaviors through contact-invariant optimization. *ACM Trans. Graph.* 31, 4 (July 2012), Article 43, 8 pages.

Carl Edward Rasmussen. 2004. Gaussian processes in machine learning. In *Advanced Lectures on Machine Learning*. Springer, 63–71.

Stuart Jonathan Russell, Peter Norvig, John F. Canny, Jitendra M. Malik, and Douglas D. Edwards. 2003. *Artificial Intelligence: A Modern Approach*. Vol. 2. Prentice Hall, Upper Saddle River, NJ.

Alla Safonova and Jessica K. Hodgins. 2007. Construction and optimal search of interpolated motion graphs. *ACM Trans. Graph.* 26, 3 (July 2007), Article 106.

Ari Shapiro, Marcelo Kallmann, and Petros Faloutsos. 2007. Interactive motion correction and object manipulation. In *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games (I3D'07)*. ACM, New York, NY, 137–144.

Hyun Joon Shin and Hyun Seok Oh. 2006. Fat graphs: Constructing an interactive character with continuous controls. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Eurographics Association, 291–298.

Steve Tonneau, Rami Ali Al-Ashqar, Julien Pettré, Taku Komura, and Nicolas Mansard. 2016. Character contact re-positioning under large environment deformation. In *Computer Graphics Forum*, Vol. 35. Wiley Online Library, 127–138.

Douglas J. Wiley and James K. Hahn. 1997. Interpolation synthesis of articulated figure motion. *IEEE Comput. Graph. Appl.* 17, 6 (Nov. 1997), 39–45.

Katsu Yamane, James J. Kuffner, and Jessica K. Hodgins. 2004. Synthesizing animations of human manipulation tasks. *ACM Trans. Graph.* 23, 3 (Aug. 2004), 532–539.

Yuting Ye and C. Karen Liu. 2012. Synthesis of detailed hand manipulations using contact sampling. *ACM Trans. Graph.* 31, 4 (July 2012), Article 41, 10 pages.

Received January 2016; revised December 2016; accepted January 2017