Pacific Graphics 2014
J. Keyser, Y. J. Kim, and P. Wonka
(Guest Editors)

Volume 33 (2014), Number 7

Environment-Adaptive Contact Poses for Virtual Characters

Changgu Kang[1] and Sung-Hee Lee[2†]

[1]Gwangju Institute of Science and Technology (GIST) [2] Korea Advanced Institute of Science and Technology (KAIST)

Abstract

We present a novel method to generate a virtual character's multi-contact poses adaptive to the various shapes of the environment. Given the user-specified center of mass (CoM) position and direction as inputs, our method finds the potential contacts for the character in the surrounding geometry of the environment and generates a set of stable poses that are contact-rich. Major contributions of the work are in efficiently finding admissible support points for the target environment by precomputing candidate support points from a human pose database, and in automatically generating interactive poses that can maintain stable equilibrium. We develop the concept of support complexity to scale the set of precomputed support points by the geometric complexity of the environment. We demonstrate the effectiveness of our method by creating contact poses for various test cases of environments.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

1. Introduction

Despite steadily advancing computer animation technologies, existing methods do not provide an efficient means for creating contact-rich whole-body poses of a virtual character interacting with complex environments, for example when it makes contact poses adaptive to various types of furniture. Thus, animators should devote a significant amount of effort to creating key-frames that deal with complicated contact configurations between a character and its environment.

The plausibility of a character's pose in an environment depends on many aspects. The character should not penetrate into the environment. A static pose must be balanced so as to be maintained for particular duration. Additionally, a proper pose should accomplish the user-specified goals, such as reaching a target point with a certain style. Since characters interact with the environment mostly through contact, it is critical to choose the correct body part to make contact with a particular location within the given environment. In particular, *support contacts* that bear a character's weight play a significant role in maintaining the balance, determining the overall pose of the character.

In this paper, we present a novel method for generating contact poses that are adaptive to the arbitrary shapes of the environment. Given CoM position and facing direction of the whole body as user inputs, our method computes the potential contact points in the surrounding geometry of the environment, and then determines stably balanced contact poses. Developing an algorithm that generates a single, optimal pose satisfying user's need, which is sometimes difficult to quantify, would be ideal but is very demanding. Instead, our method takes a practical approach that generates multiple candidate poses, from which a user may conveniently select the preferred pose. Thus, our method can be straightforwardly adopted to create key-frame poses of virtual characters.

We generate environment-adaptive poses using the sample poses in the database. Due to the sheer diversity of environments, the size of motion database should be extremely large to provide a motion clip that can match an arbitrary target environment. A feasible alternative is capturing both the motion and the environment and then adjusting the motion to a modified environment, but this approach would easily face scalability issues. Rather, because there can be various shapes of environments (or contact configurations) compatible with a single pose, we should be able to associate many different environments with a pose. To this end, our novel approach is to precompute in an offline process a number of possible contact configurations, represented with *support point sets*, from the sample poses in a database. The support point set is a set of contact points through which a character can maintain stable balance against the environment. After that, when a target environment is given, our method selects the admissible support point sets from the precomputed data,

† Corresponding author. E-mail: sunghee.lee@kaist.ac.kr

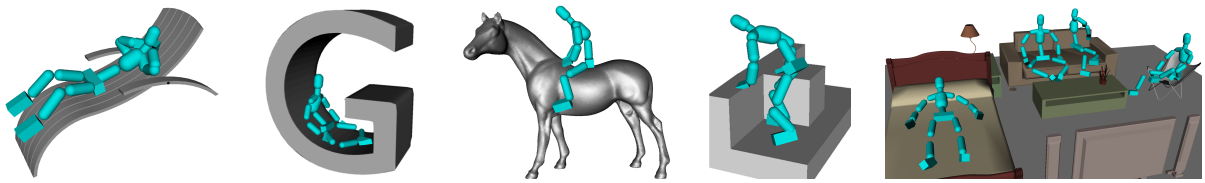


Figure 1: *The proposed method can generate various contact-rich poses that are adaptive to the geometry of the environments.*

and subsequently generates contact poses that match the target environment.

The main idea of our method concerns finding suitable support point sets from a pose. Let us assume that a number of sample points are distributed over the surface of a character and we will find support point sets from the combinations of the sample points. A set of sample points is regarded as a valid support point set if a character can maintain a static equilibrium against the support points and can exert enough muscle forces to keep the pose. Modeling contacts between a character and an environment as point contacts with friction, the number of support points to maintain the balance (or static equilibrium) should be three or more. Even for a moderate number of sample points, the total number of support point sets associated with a particular pose can be very high, making it necessary to develop a reasonable means to control the size of support point sets. To this end, we define the *support complexity*, by which we measure how complex the corresponding environment should be to contain a given support point set, and use this to scale the precomputed support point sets.

The main contribution of this work is in a novel framework of computing a human character's contact poses based on the precomputed support point sets, and the framework provides the following advantages:

- By collecting a multitude of support point sets from a pose, our method finds suitable contact poses for a wide variety of environment shapes.
- Our method runs much faster than when directly matching the candidate poses to the environment thanks to the offline process of precomputing the support point sets.

We demonstrate the effectiveness of our method by creating contact poses for various shapes of environments. Fig. 1 shows some of the example poses created by our method.

2. Related Work

Generating a virtual character's pose or motion is a central problem in computer animation, and researchers have developed various methods in the key frame-based, data-driven, and physics-based approaches. In this paper, we mainly discuss previous work that deals with creating a character's pose or motion that involve multiple contacts with the environment or objects therein.

A large body of research has concentrated on creating motions to achieve particular goals while avoiding collision with the environment, e.g., [KNK*01, LCR*02]. Recently, researchers in computer animation and robotics have been studying motion generation problems that involve complex contacts. Based on the randomized sampling method, Liu et al. [LYvdP*10] developed a physics-based animation technique that generates motions involving complex contacts with an environment, such as rolling. Bouyarmane and Kheddar [BK11] developed a motion planning method for a humanoid robot to compute the optimal motion interpolating several key poses that may have multiple contacts with the environment. Mordatch et al. [MTP12] proposed a method to optimize the motion and the contact configurations simultaneously. In doing so, they showed impressive results in which virtual characters could make complex, contact-rich transitions that could not easily be created by conventional footstep planning-based approaches. Our objective is to create a set of plausible poses for the environment. Thus, our method can be utilized for motion planning techniques to enrich the possible pose candidates in a complex environment.

Generating a whole body motion that interacts with a complex environment is quite a challenging task, requiring a sophisticated treatment of the relationship between the shape of the environment and the pose of a character. A practical approach for this is to construct a database that stores example character-environment interactions and apply data-driven techniques such as motion editing, e.g., [LCL06] and [HKT10]. While these methods create highly realistic character animation that is adaptive to the environment, the number of the required motion examples increases with the increasing number of example environments. To improve this possible limitation, we generate contact-rich poses from a generic pose database, rather than capturing poses for particular environments. Thus, our method does not suffer from the scalability issue induced by diverse environments.

In computer vision, researchers have recently studied the human-object interactions for object recognition, scene understanding, and finding environment-adaptive human poses. Grabner et al. [GGG11] and Gupta et al. [GSEH11] examined the affordance of objects and scenes by locating a few human pose templates to the target environment and evaluating the fitness of the two based on the penetration and predefined target support points. We present a more advanced technique that generates new poses compatible to target environments in arbitrary shapes using the sample poses

in the database and inverse kinematics, as well as equilibrium analysis.

Lin et al. [LIM*12] share our goal of creating contact-rich human poses efficiently. They developed a novel framework to create human sitting poses on chairs using a sketching interface in which a user draws a target pose as a two-dimensional stick figure. In contrast, we take a different approach that automatically finds various candidate poses suitable to the target environment and lets a user select the preferred pose. In addition, our method is not targeted specifically for making sitting poses, rather it allows for creating various contact poses including lying and standing poses.

Hand manipulation and grasping problems have been researched more extensively than the whole body contact problems [KKKL94, Liu09, YL12]. While most studies have treated these problems separately from whole body problems, some deal with whole body posture planning for manipulation [YKH04]. One notable study has been undertaken by Li et al. [LFP07], who solved the grasping problem from the perspective of shape matching. After defining a global feature for grasping, they randomly generated the features on the target objects, and verified the containment of particular features related to a candidate hand pose. Taking a similar approach, we define specific features for finding suitable contact poses, and use the features to measure the matching quality of a pose with respect to the environment.

3. Background: Static Equilibrium Of a Character

The primary condition of a suitable pose in a target environment is that the pose satisfies the static equilibrium. Treating the character as a single rigid body under the assumption that the character can generate enough joint torques, statically stable equilibrium is achieved if the sum of wrenches generated by all external forces is zero. Expressing the relation with respect to a reference frame located at a character's center of mass (CoM), we get

$$\sum_i \mathbf{G}_i \mathbf{f}_i = -\begin{bmatrix} \mathbf{0} \\ m\mathbf{g} \end{bmatrix} - \sigma, \ \mathbf{f}_i \in F_i, \tag{1}$$

where $\mathbf{G}_i \in \mathbb{R}^{6\times 3}$ is a matrix transforming a contact force $\mathbf{f}_i$ of a support point to a wrench with respect to the reference frame, m is the total mass of a character, $\mathbf{g} = (0, g, 0)^T$ is the gravitational vector (the Y-axis is the up-vector), $\sigma \in \mathbb{R}^6$ is the wrench due to external perturbation. We model the contact as a point contact with friction, and $\mathbf{f}_i$ must lie inside the friction cone F_i.

If we assume σ to be an arbitrary vector in $\mathbb{R}^6$, (1) is equivalent to the force closure condition in grasp analysis; that is, $\sum_i \mathbf{G}_i \mathbf{f}_i$ must span $\mathbb{R}^6$. However, for stable equilibrium, we can reasonably assume that the perturbation force in upward direction cannot exceed the gravitational force (i.e., perturbation force is not strong enough to lift a character), and then the condition for stable equilibrium can be written as follows:

$$\text{span}\big(\sum_i \mathbf{G}_i \mathbf{f}_i\big) \supseteq \mathbb{R}^5 \times \mathbb{R}^+, \ \mathbf{f}_i \in F_i, \tag{2}$$

where $\mathbb{R}^+$ refers to the positive half-space defined by XZ-plane and $\mathbb{R}^5$ corresponds to the remaining dimensions. Obviously, if an object is in force closure, it can be stably in static equilibrium, but its converse is not necessarily true. Convex analysis confirms that at least three support points are necessary to satisfy (2) [MLS94].

In order to achieve the static equilibrium, a character must be able to exert necessary contact forces. The joint space equations of motion of a character in steady state (i.e., joint velocities and accelerations are zero) are as follows:

$$\tau_g(\mathbf{q}) - \sum_i \mathbf{J}_i^T \mathbf{G}_i \mathbf{f}_i - \mathbf{J}_\sigma^T \sigma = \tau, \ \tau_l \le \tau \le \tau_u, \ \mathbf{f}_i \in F_i, \tag{3}$$

where $\mathbf{q} \in \mathbb{R}^n$ denotes the generalized coordinates of n degrees of freedom, $\tau_g(\mathbf{q})$ is the joint space gravity term, $\mathbf{J}_i$ and $\mathbf{J}_\sigma$ transform the wrenches to the joint space, and $\tau \in \mathbb{R}^n$ is the generalized forces, bounded by τ_l and τ_u. Modeling the character's root as a floating link, the first six elements of τ must be zero. Then the equilibrium is achieved if there exist such τ and $\mathbf{f}_i$ that satisfy (3) under some perturbation σ.

In fact, (3) is a sufficient condition of (2), and hence one can verify static equilibrium only by examining (3). However, since solving (2) is orders of magnitude faster than solving (3), we reduce overall compute time by first sifting out candidate sample points satisfying (2) before further testing with (3).

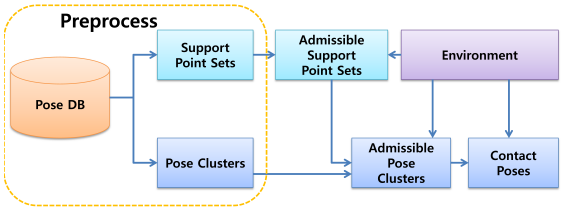
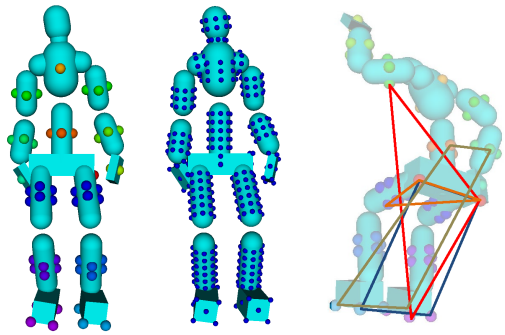


Figure 2: *The overall framework of the pose generation process.*

4. Overview

We assume that a human pose database is available and the geometry information of the environment is given as point cloud data. Then, our method consists of a preprocess to collect the contact-related features from a pose database and an online process to create suitable contact poses for a specific environment. Fig. 2 shows the overall framework of our proposed method.

As an offline preprocess, we construct the *pose clusters* and the *support point sets* from a human pose database. First, by examining every pose in the database, we collect a large collection of support point sets (Sec. 5.1), which will be used



(a) (b) (c)

Figure 3: *(a) Sample points on a character for finding support point sets. (b) Points for penetration test (Sec. 6.4). (c) A subset of support point sets found from a sitting pose. Only 4 sets are drawn from a total of 2200 support point sets found from this pose. Line segments connect points in the same set for visualization purposes.*

to find candidate support points from the environment. Subsequently, the poses in the database are clustered into a set of pose clusters that serve as units for matching poses to the environment. Pose clustering reduces the number of pose-environment matching tests and the number of poses presented to the user.

In the online process, when a user specifies CoM position and the facing direction of a character in the environment, a set of plausible poses is generated. First, the admissible support point sets that match the target environment are collected and then used to find admissible pose clusters that can maintain balance (Sec. 6.2). Finally, the user can browse the pose clusters and apply inverse kinematics to generate preferred poses (Sec. 6.4).

5. Preprocess

In general, surface contact is a dominant type of contact in character-environment interaction. We simplify this by considering contacts for a number of sample points on the surface of the character's body. Fig. 3(a) shows the sample points $s_i (i = 1 \dots 70)$ we assigned to the character model. More sample points are attached to the body parts (e.g., the pelvis and thighs) that are considered more important with respect to the character-environment contact in order to distinguish sub-part contacts.

We ensure that all the poses in the database are aligned to face forward. To this end, we calculate a pose's *direction vector*, defined as the vector normal both to the up-vector and to the vector passing through the two points of the shoulder joints. Alignment is performed by rotating the pose about the vertical axis so that the direction vector points to the frontal direction.

5.1. Collecting Support Point Sets

A pose can be maintained in an environment if there is at least one support point set and no penetration exists between the pose and the environment. While the penetration can be examined only when both the pose and the target environment are given, support point sets can be found solely from a pose. Therefore, our strategy is to extract candidate support point sets from the pose database offline. Later, when a target environment is given, we will find admissible support point sets for the environment, and will then find the poses corresponding to the admissible support point sets. Thereby, we can efficiently acquire balanced poses that are adaptive to the environment.

Representation. We represent a support point set with the positions and the indices of the corresponding sample points; that is, $t = ((s_1, \mathbf{v}_1), (s_2, \mathbf{v}_2), \dots, (s_k, \mathbf{v}_k))$ where $\mathbf{v}_i \in \mathbb{R}^3$ is the position of the sample point s_i and $k (\geq 3)$ refers to the cardinality of the set. $\mathbf{v}_i$ is expressed as the relative coordinates with respect to the CoM of a character so that they can easily be located in the environment when a user specifies the target location of the CoM.

Algorithm. The precomputed support sets consist of the support point sets of every pose in the database. For each pose, we collect the support point sets such that they are mutually exclusive, i.e., a support point set is not a subset of any other set. The mutual exclusiveness ensures that every single point in a support point set is necessary to satisfy the static equilibrium condition, and is important to keep the size of support point sets as compact as possible. To this end, we find the support point sets in the increasing order of cardinality to ensure that a candidate point set does not include a valid support point set of a lower cardinality.

Let S_k denote the support point sets of a pose with the maximum cardinality of each set being k. The pseudocode for collecting S_k is provided in Algorithm 1, of which key subroutines are the one checking the complexity of support and the other verifying the static equilibrium, both of which are described in detail now.

Algorithm 1 Collecting support point sets of k-cardinality

Require: S_{k-1}
1: $S \Leftarrow \emptyset$
2: **for all** $P_i = k$-combination of sample points **do**
3: **if** Any subset of P_i is included in S_{k-1} **then**
4: continue
5: **if** SupportComplexity(P_i) > threshold **then**
6: continue
7: **if** CheckStaticEquilibrium(P_i) = true **then**
8: $S = S \cup P_i$
9: $S_k = S_{k-1} \cup S$
10: **return** S_k

5.1.1. Verifying Static Equilibrium

Given a k-combination of sample points, we verify whether the points satisfy the equilibrium conditions (2) and (3). Modeling a friction cone at the surface of a body part as a friction pyramid using four basis vectors $\mathbf{b}_{1,\ldots,4}$, a contact force is expressed as

$$\mathbf{f}_i = \sum_{j=1}^{4} \mathbf{b}_{i,j} x_{i,j} = \mathbf{B}_i \mathbf{x}_i, \tag{4}$$

where $\mathbf{B}_i = \left[\mathbf{b}_{i,1}, \ldots, \mathbf{b}_{i,4}\right] \in \mathbb{R}^{3\times 4}$ and $\mathbf{x}_i = \left[x_{i,1}, \ldots, x_{i,4}\right]^T \in \mathbb{R}^4$. Then, one can write (2) as

$$\text{span}(\mathbf{Ax}) \supseteq \mathbb{R}^5 \times \mathbb{R}^+, \ \mathbf{x} \geq \mathbf{0}, \tag{5}$$

where $\mathbf{A} = [\mathbf{G}_1\mathbf{B}_1, \ldots, \mathbf{G}_k\mathbf{B}_k]$, $\mathbf{x}^T = [\mathbf{x}_1^T, \ldots, \mathbf{x}_k^T]$. Non-negativity of $\mathbf{x}$ is due to the unilateral property of contact force. We modify (5) to make a simpler form: By appending a column $\mathbf{y}^- = [0,0,0,0,-1,0]^T$ to $\mathbf{A}$, we can transform (5) to an equivalent problem

$$\text{span}\left(\left[\mathbf{A}\ \mathbf{y}^-\right]\begin{bmatrix}\mathbf{x}\\ \lambda\end{bmatrix}\right) = \mathbb{R}^6, \ \mathbf{x} \geq \mathbf{0}, \ \lambda \geq \mathbf{0} \tag{6}$$

for some dummy variable λ. Equation (6) is satisfied if $\left[\mathbf{A}\ \mathbf{y}^-\right]$ positively spans $\mathbb{R}^6$, which can be verified by a number of methods (e.g., [MLS94]). Following [PL12], we first transform $\left[\mathbf{A}\ \mathbf{y}^-\right]$ into reduced row echelon form $[\mathbf{I}\ \mathbf{S}]$, where $\mathbf{I}$ is the 6×6 identity matrix and $\mathbf{S} \in \mathbb{R}^{6\times(4k-5)}$, and examine whether $\mathbf{Sy}$ can make a vector of which entries are all strictly below zero for some $\mathbf{y} \geq \mathbf{0}$ by solving a linear programming problem (See [PL12] for details).

Condition (2) only checks whether a support point set can maintain stable equilibrium provided that arbitrary contact forces can be applied within friction cones, and thus the point sets satisfying (2) may still be infeasible if a character cannot exert necessary contact forces, which is verified by the second condition (3). Although we need to know the perturbation force σ to examine (3), we simply assume that the condition (3) is satisfied if the joint torque solution τ for (3) lies sufficiently inside the bound $[\tau_l, \tau_u]$ when no perturbation is applied.

We verify condition (3) by solving the bounded-variable least-squares problem [SP95]

$$\underset{\tau,\mathbf{x}}{\text{argmin}} \left\| \begin{bmatrix}\mathbf{I} & \mathbf{C}\\ \varepsilon_\tau\mathbf{I} & \mathbf{0}\\ \mathbf{0} & \varepsilon_\mathbf{x}\mathbf{I}\end{bmatrix}\begin{bmatrix}\tau\\ \mathbf{x}\end{bmatrix} - \begin{bmatrix}\tau_g\\ \mathbf{0}\\ \mathbf{0}\end{bmatrix} \right\|^2, \ \alpha\tau_l \leq \tau \leq \alpha\tau_u, \ \mathbf{x} \geq \mathbf{0}, \tag{7}$$

where $\mathbf{C} = \left[\mathbf{J}_1^T\mathbf{G}_1\mathbf{B}_1, \ldots, \mathbf{J}_k^T\mathbf{G}_k\mathbf{B}_k\right]$, and check whether the solution is feasible. The second and third rows in the matrix drive τ and $\mathbf{x}$ toward zero, and ε_τ and $\varepsilon_\mathbf{x}$ are small positive constants. α $(0 < \alpha < 1)$ controls the margin of torque bounds. Instead of solving (7), one may formulate (3) in a linear programming problem.

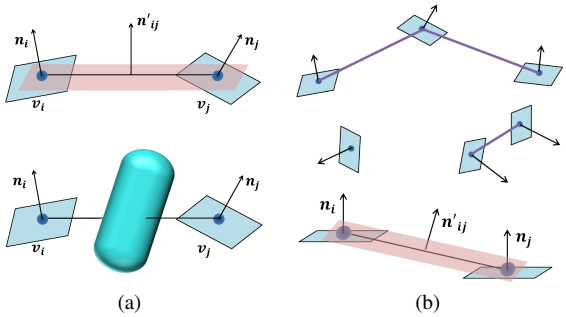


Figure 4: *Computing the support complexity.*

5.1.2. Support Complexity

Since the number of valid support point sets increases rapidly with the cardinality, collecting all the support point sets satisfying the static equilibrium condition, although seeming ideal at first thought, is not pragmatic. Rather it would be more memory- and computation-efficient to collect only the support point sets that are likely to appear in the environments in the online pose generation session. To this end, we define the complexity of a support point set, dubbed the *support complexity*, and collect only those support point sets with complexity below some threshold.

The support complexity should not be measured trivially by the number of support points because, for example, a lying pose on a flat ground may contain many support points but it should be regarded as a simple support. Instead, we define the support complexity as the minimum number of planes that can include the given support points.

Figure 4 shows how the support complexity is measured. A basic element for this is to measure how likely two support points lie in the same plane, which is computed as follows: each support point $\mathbf{v}_i$ and its normal $\mathbf{n}_i$ define a surface patch p_i. Given two surface patches p_i and p_j corresponding to two support points, we compute a *mean plane* intersecting $\mathbf{v}_i$ and $\mathbf{v}_j$, with its normal $\mathbf{n}'_{ij}$ determined to be closest to $\mathbf{n}_i$ and $\mathbf{n}_j$ in the least-squares sense (a pink plane in Fig. 4(a)). Then the planar distance $planalDist(p_i, p_j)$ is defined in terms of the discrepancies of normals:

$$planalDist(p_i, p_j) = \sum_{k=i,j} |\cos^{-1}(\mathbf{n}_k^T\mathbf{n}'_{ij})|. \tag{8}$$

A planar distance is zero when two patches lie perfectly in the mean plane and gets higher as the patches are pivoted from the mean plane. Figure 4(b) (bottom) shows a case of non-zero planar distance between two patches having the same normals. Additionally, we examine whether a line segment between the two support points intersects the character's body, in which case the two points are considered to be in separate planes and thus the planar distance is set to infinity (Fig. 4(a), bottom).

The minimum number of planes is calculated based on

Cardinality	No complexity check	Threshold = 4	Threshold = 3
3	363	363	363
4	1003	1003	308
5	1987	770	71
6	2667	376	32

(a) A standing pose.

Cardinality	No complexity check	Threshold = 4	Threshold = 3
3	343	343	343
4	499	499	157
5	1886	448	67
6	4038	149	4

(b) A lying pose.

Figure 5: *The number of valid support points sets of each cardinality of a standing (top) and a lying (bottom) poses when the support complexity check is not performed, and when the threshold is set to three or four.*

graph algorithms. We first construct a graph that has the support points as vertices but has no edges. Then for each pair of vertices, an edge is created if the planar distance (8) between the two points is less than a threshold (5 degrees in our experiment). The number of planes is set to the number of connected components of the graph. In the example shown in Fig. 4(b), the support complexity is three.

Note that we store the correspondences between a pose and the associated support point sets so that we can easily find a corresponding pose from a support point set and vice versa in the online pose generation process.

5.2. Pose Clustering

The high number of poses in the database brings benefits of finding many support point sets that could exist in the environments. As the pose database contains many poses that look very similar, structurizing the pose data is important for the online pose generation process to ensure that the user is given only perceivably different options for choosing the preferred poses.

To this end, we divide the poses into a set of pose clusters. We employ a standard k-mean clustering algorithm in which the coordinates of the sample points relative to the root position are used as features. Additionally, the jump method [SJ03] is used for determining the number of clusters in a pose database. The simple k-mean clustering works well for our purpose while more advanced algorithms could increase the quality of pose clusters. Note that we store the mean pose of each cluster as a representative pose for the cluster.

Experiment. We constructed a pose database by collecting all the static and distinct poses in the CMU motion capture database (http://mocap.cs.cmu.edu/). We determine a pose is static if the speeds of all end effectors are below 0.07m/sec

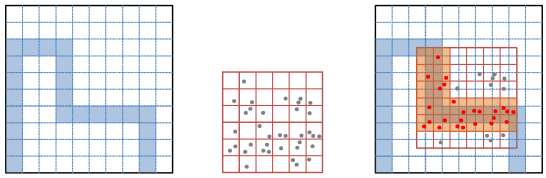


Figure 6: *Right: Surface cells (blue) in an environment grid. Middle: A pose grid. Left: The pose grid is located inside the environment grid. Admissible support points (red) are included in the pose grid cells overlapping with surface cells (orange).*

assuming that the hip does not move fast in that case. A total of 6,500 distinct poses are gathered in the pose database and divided to 400 clusters. Figure 5 shows the number of valid support point sets for sample poses. While the number of valid sets increase rapidly with the cardinality, enforcing the support complexity limit efficiently constrains the number of support point sets to be stored. From the pose database, we collected approximately 5,482,000 support point sets by setting the support complexity threshold to 4.

6. Online Pose Generation

When a user sets the desired position and direction for a character on the surface of the environment, our online algorithm creates suitable poses that are adaptive to the environment.

6.1. Environment Processing

We assume that the geometry of the environment is given as a point cloud because it is a common data format used by 3D scanners and other formats, e.g. mesh data, can be easily converted to the point cloud data by point sampling. In order to perform collision and interior tests efficiently, we structuralize the point cloud through a regular 3-D grid, dubbed the *environment grid*, and compute the signed distance of each grid cell's center to the surface of the environment. The distance of the *surface cells* that contain cloud points inside is set to 0, and the interior cells are given a negative distance to the nearest surface cell while exterior cells are given positive distance values. We performed the interior check for a cell simply by using the odd-parity rule and manual fix-up if necessary. There exist many advanced methods for finding the signed distance function from point cloud data, e.g., [CL96]. If the target environment is static and known beforehand, the environment processing can be performed in the preprocessing stage. Using the signed distance of the grid, we implement a function $d_E(\mathbf{v})$ that returns the signed distance of a point $\mathbf{v} \in \mathbb{R}^3$ to the nearest surface cell, calculated by interpolating the distance values of neighboring cells.

6.2. Admissible Support Point Sets and Pose Clusters

The user-specified location of the CoM and the facing direction determine the absolute coordinates of the points in the support point sets, from which we can find the admissible support point sets that match the target environment. A support point set is classified as admissible if all of its points lie in the surface cells. Note that we do not consider normal directions at contact points on the surface for evaluating the validity of support point. This was a choice motivated by the facts that computing normal directions requires significant amount of time.

As there are millions of points in the precomputed support point sets, testing individual point for environment matching requires prohibitive computational cost for interactive applications. Instead, we significantly reduce the compute time by constructing another 3-D regular grid centered around the CoM, dubbed the *pose grid*, which includes the support points inside the cells, and relaxing the point-grid matching task to a grid-grid matching task (Fig. 6). A pose grid is fixed to the reference frame at CoM and thus located inside the environment grid according to the CoM position and facing direction set by the user. In our experiment, we compute the pose grid in the beginning of the online pose generation process and adjust its cell lengths to be the same as those of the environment grid. Instead, one could generate a pose grid in the offline process.

Now the problem is reduced to finding pose grid cells that overlap with surface cells of the environment. Note that a pose grid can rotate about the Y-axis due to the user input and thus its X-and Z-axes are generally not aligned with those of the environment grid. For the collision check, we simply find pose grid cells that intersect with the eight corners and a center point of surface cells.

From a set of pose clusters, we find the admissible pose clusters that can match the target environment in the sense that they are associated with admissible support point sets and do not severely penetrate the environment. Penetration test is performed against the mean pose of the pose cluster by calculating the penetration cost as the sum of penetration depths of all the sample points of the pose using the function $d_E(\mathbf{v})$. We discard a pose cluster of the penetration cost higher than some threshold. A pose cluster containing a pose associated with at least one admissible support point set should be a candidate for admissible pose cluster, but we found that plausible results are created by those pose clusters with multiple admissible support point sets. Having only a small number of admissible support point sets roughly implies that a pose matching the environment would be far from the mean of the cluster, in which case our IK process in Sec. 6.4 may not perform well. Thus, we select those pose clusters that are associated with at least five admissible support point sets. The pseudocode for finding admissible pose clusters is shown in Algorithm 2.

Algorithm 2 Finding admissible pose clusters

Require: All points are marked "off". All pose clusters are marked "inadmissible" and their admissibilityCount's are set to zero.

```
for all E_i = a surface cell do
    Find the pose grid cells G_j's overlapping with E_i.
    for all G_j do
        Mark all points inside G_j "on"
for all P_i = a support point set do
    if all points are marked "on" then
        C_j ⇐ the pose cluster of P_i.
        Increase C_j.admissibilityCount by one.
for all C_i = a pose cluster do
    if C_i.admissibilityCount ≥ 5 then
        if C_i passes the penetration test then
            Set C_i as "admissible".
```

6.3. Selecting Pose Clusters

All of the admissible pose clusters are valid for the target environment, and it depends on the particular need as to which poses are selected. Using the information on the admissible support point sets and pose clusters, one can present all the admissible pose clusters to a user, or develop algorithms for finding poses relevant to additional queries, such as "poses that contact with particular body parts" and "poses that contact at a particular location in the environment". One can also sort the admissible pose clusters by some metric, such as by the number of contacting body parts or by the magnitude of required joint torques. Such selection tools can be implemented using the functions provided by our method in a straightforward way. Fig. 8 shows the poses in the order of increasing CoM heights and in different directions.

6.4. Inverse Kinematics

For each pose cluster selected by a user, we perform inverse kinematics to generate a pose for the target environment. Inverse kinematics is carried out by minimizing the following cost function f:

$$f = w_{sup} f_{sup} + w_{pen} f_{pen} + w_{mean} f_{mean} \tag{9}$$

$$f_{sup} = \sum_{i,j} ||s_i(\mathbf{q}) - \mathbf{c}_{i,j}||^2 \tag{10}$$

$$f_{pen} = \sum_{i} \min[0, d_E(z_i(\mathbf{q}))]^2 \tag{11}$$

$$f_{mean} = ||\theta - \theta^*||^2 \tag{12}$$

where $\mathbf{q} = (\mathbf{T}, \theta)$ denotes the generalized coordinates of a character. $\mathbf{T} \in \mathrm{SE}(3)$ is the transformation matrix of the root link, and θ is the joint angle vector. f_{sup} refers to the cost due to the distance between the sample points and $\mathbf{c}_{i,j} \in \mathbb{R}^3$, the support points corresponding to the sample point s_i. $\mathbf{c}_{i,j}$ are collected from the admissible support point sets associated with the target pose cluster. f_{pen} penalizes the penetration of the character into the environment. To this end, we attach a set of sample points z_i densely to the surface of a body for

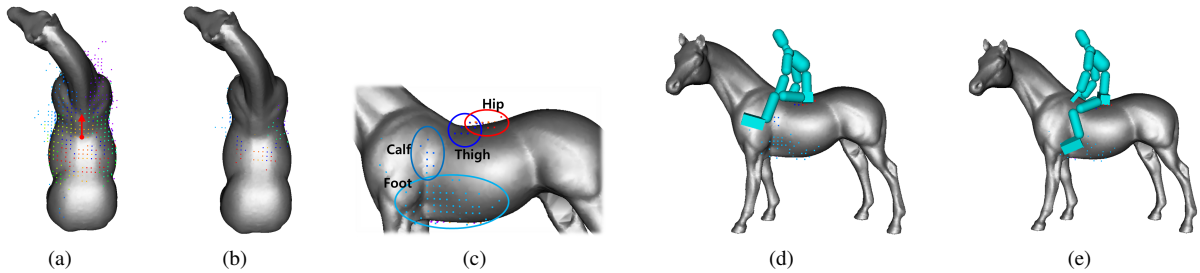


Figure 7: *(a) Points from all admissible support point set, color-coded by the corresponding body parts. Red, blue, and green points correspond to the hip, legs, and hands, respectively. User-specified CoM position and direction are marked with a red arrow. (b-c) Points from the admissible support point sets associated with a selected pose cluster. (d) The mean pose of the selected pose cluster, located at the user-specified CoM position. (e) Resulting pose after performing inverse kinematics. Note that the overall pose has become closer to the surface of the horse model.*

collision detection, as shown in Fig. 3(b), and measure their signed distances in the environment grid. f_{mean} encourages the joint angles θ to be close to those of the mean pose of the pose cluster, denoted by θ^*. The weights w_i control the importance of each cost. We adopt the SNOPT package, an SQP-based solver, for the optimization.

7. Experiments

We tested our method for a variety of environments represented as point cloud data, acquired by sampling points from synthetic mesh models. Figure 7 shows the procedure to create a pose with respect to a horse model. When the user specifies CoM position and direction for a character, our method generates admissible support point sets for the target environment (Fig. 7(a)), and subsequently the admissible pose clusters. When the user selects a pose cluster, the admissible support point sets corresponding to the selected pose cluster provide support points for inverse kinematics. The support points belonging to the same body part tend to form a cluster as shown in Figs. 7(b) and 7(c), a natural consequence of how point sets have been constructed. Figure 7(d) shows the mean pose of the selected pose cluster, located at the used-defined CoM position. Figure 7(e) shows the final pose created after the inverse kinematics operation. One can see that the resulting pose engages multiple contacts with the target environment while preserving the style of the selected pose cluster. Figure 8 shows various poses for a horse model by different CoM heights and facing directions. The example shows that our method can create a number of suitable poses for curved surfaces, such as the back of a horse. Our algorithm does not consider coherence between consecutive user inputs, but similar poses tend to be included in the respective sets of resulting poses per similar inputs, as demonstrated in the accompanying video.

Figure 9 demonstrates that our method can create a broad range of contact poses including those which have unusual contact configurations such as being supported by the arms. Since the environment grid describes only the occupancy of a space by objects in the environment, our method performs regardless of the number or attributes of the objects. Thus, a virtual character interacts with multiple objects in the environment in the same manner as it interacts with a single object, as demonstrated by a character making poses on multiple bases of different heights shown in Fig. 10.

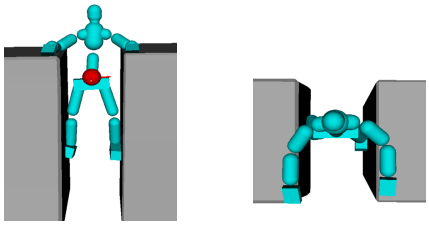


Figure 9: *A pose supported by the arms for a deep hole, seen from front and top views. Red sphere marks the user-specified CoM position.*

Figure 11 shows a number of poses created for various environments. Our method creates a variety of poses including lying, sitting, and standing poses. Figures 11(a)-11(c) show poses that are partly supported by the chest and hands. Figures 11(d)-11(f) show various poses that can be made on a sittable environment. The character can lean on a wall with the back or even with the hip. Figures 11(g)-11(i) show poses generated for furniture and complex environments.

It takes approximately 1.0 seconds for finding pose grid cells intersecting with surface cells, 0.6 seconds for finding admissible pose clusters, and 0.2 seconds for inverse kinematics using a single core of a 3.4GHz Intel i7 CPU PC. The size of grid cells (40 mm in our experiment) affects the number of valid poses and compute time. Generally, a larger grid cell admits more contact point sets, which increases the number of admissible pose clusters at the cost of increased compute time. Our method can be parallelized with ease, which can enhance the performance significantly.

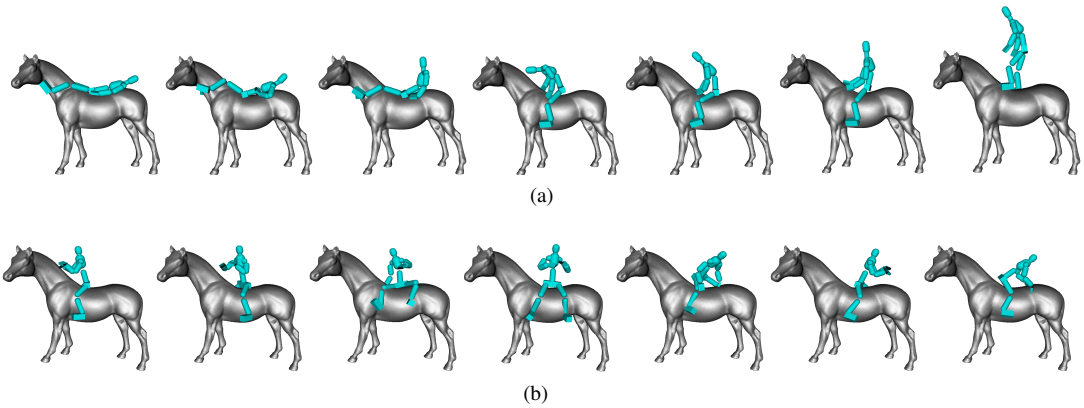


Figure 8: *Poses created for different CoM heights and facing directions.*

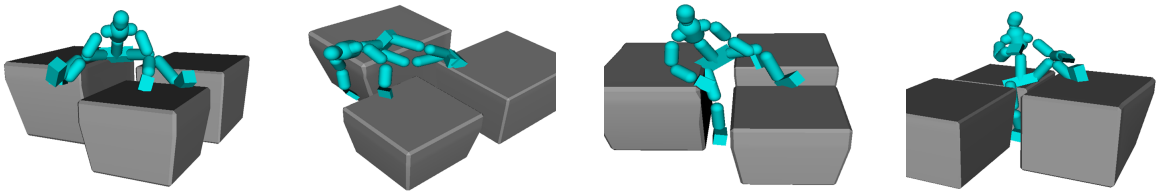


Figure 10: *Poses created for multiple bases of different heights, separated with large gaps.*

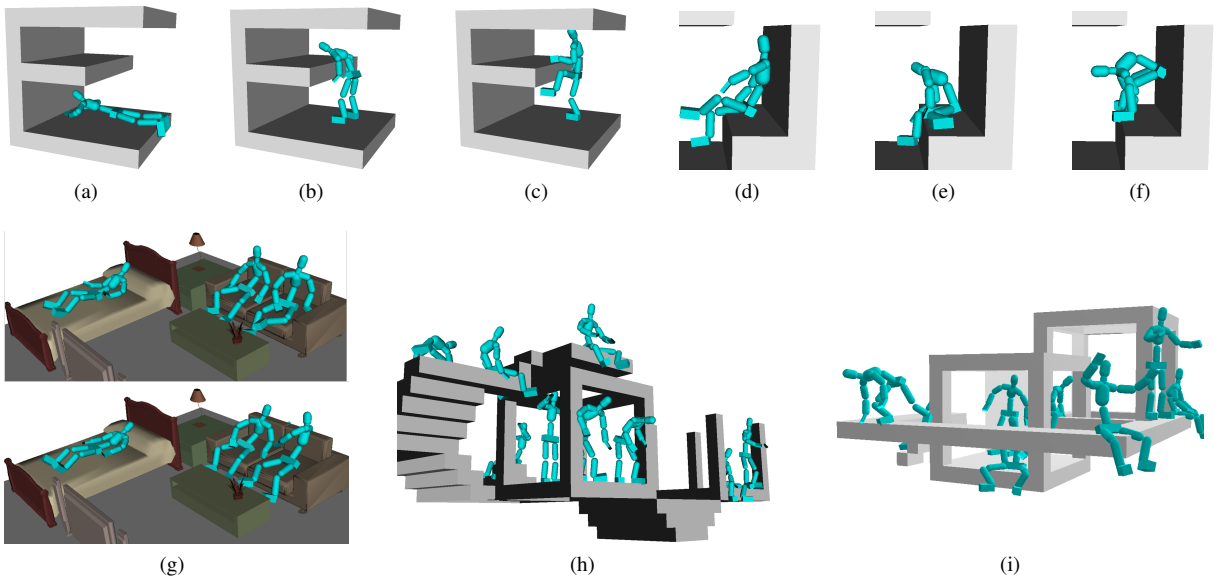


Figure 11: *Poses created for various environments.*

8. Conclusion and Future Work

We introduced a novel method to generate environment-adaptive character poses. In the preprocess, by analyzing the human pose database, we extract the pose clusters as well as the support point sets. Given a user-specified position and direction of a character in the target environment, we find the admissible support point sets for the environment, and then collect valid pose clusters using the admissible support point sets and the penetration test. A user can efficiently generate the desired pose by performing inverse kinematics for the selected pose cluster.

Our method has limitations in the following aspects. Since we assume a friction coefficient in the preprocess (4), a resulting pose may fail to satisfy the static equilibrium if the friction coefficients of the target environment are substantially smaller. This problem can be partially remedied by adding the equilibrium constraint in the inverse kinematics step. Our discrete and soft penetration-avoidance scheme (11) may not completely remove interpenetration between the character and the environment. Additionally, since support point sets depend on the body dimensions of the character, another drawback is that the preprocess should be carried out for each character.

There are many interesting venues for the future work. Improving the framework so as to take normal directions of the environment into account would enhance the physical plausibility of the resulting poses. Currently, we assume that a user specifies both the position and direction of a virtual character, but a more general method would receive as input only the position or even the region for a character to interact with an environment. In this regard, a technique to find suitable interaction directions and positions from the environment would be very useful.

Our method creates geometrically (penetration-safe) and physically (balanced) correct poses for a given environment, and the poses are close to the captured human poses. Nevertheless, the resulting poses may not harmonize with the environment because higher-level criteria such as comfort and affordance have not been considered, which is an important future research issue to create more natural and intelligent environment-adaptive poses.

Another important future effort would be to develop motion planning algorithms for transitions between the poses generated by the proposed method. Transition motions that require minimal contacts (e.g., walking) are not difficult to create, but it remains a significant challenge to plan contact-rich transitions such as crawling and rolling [LYvdP*10]. We expect that the proposed method can be further developed for planning contact-rich transitions efficiently.

Acknowledgements

This work was supported by the Global Frontier R&D Program funded by NRF, MSIP, Korea (2012M3A6A3055690), and by the Basic Science Research Program of NRF, Korea (2010-0025725).

References

[BK11] BOUYARMANE K., KHEDDAR A.: 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)* (2011), pp. 4414–4419. 2

[CL96] CURLESS B., LEVOY M.: A volumetric method for building complex models from range images. In *SIGGRAPH '96* (1996), pp. 303–312. 6

[GGG11] GRABNER H., GALL J., GOOL L. V.: What makes a chair a chair? In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)* (2011), pp. 1529–1536. 2

[GSEH11] GUPTA A., SATKIN S., EFROS A. A., HEBERT M.: From 3D scene geometry to human workspace. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)* (2011). 2

[HKT10] HO E. S. L., KOMURA T., TAI C.-L.: Spatial relationship preserving character motion adaptation. *ACM Trans. Graph. 29*, 4 (July 2010), 33:1–33:8. 2

[KKKL94] KOGA Y., KONDO K., KUFFNER J., LATOMBE J.-C.: Planning motions with intentions. In *SIGGRAPH '94* (1994), pp. 395–408. 3

[KNK*01] KUFFNER J., NISHIWAKI K., KAGAMI S., INABA M., INOUE H.: Motion planning for humanoid robots under obstacle and dynamic balance constraints. In *IEEE Int'l Conf. Robotics and Automation (ICRA)* (2001), pp. 692–698. 2

[LCL06] LEE K. H., CHOI M. G., LEE J.: Motion patches: building blocks for virtual environments annotated with motion data. *ACM Trans. Graph. 25*, 3 (July 2006), 898–906. 2

[LCR*02] LEE J., CHAI J., REITSMA P., HODGINS J., POLLARD N.: Interactive control of avatars animated with human motion data. *ACM Trans. Graph. 21*, 3 (July 2002), 491–500. 2

[LFP07] LI Y., FU J. L., POLLARD N. S.: Data driven grasp synthesis using shape matching and task-based pruning. *IEEE Trans. Vis. Comp. Graph. 13*, 4 (2007), 732–747. 3

[LIM*12] LIN J., IGARASHI T., MITANI J., LIAO M., HE Y.: A sketching interface for sitting pose design in the virtual environment. *IEEE Trans. Vis. Comp. Graph. 18*, 11 (2012), 1979–1991. 3

[Liu09] LIU C. K.: Dextrous manipulation from a grasping pose. *ACM Trans. Graph. 28*, 3 (July 2009), 59:1–59:6. 3

[LYvdP*10] LIU L., YIN K., VAN DE PANNE M., SHAO T., XU W.: Sampling-based contact-rich motion control. *ACM Trans. Graph. 29*, 4 (July 2010), 128:1–128:10. 2, 10

[MLS94] MURRAY R. M., LI Z., SASTRY S. S.: *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994. 3, 5

[MTP12] MORDATCH I., TODOROV E., POPOVIĆ Z.: Discovery of complex behaviors through contact-invariant optimization. *ACM Trans. Graph. 31*, 4 (July 2012), 43:1–43:8. 2

[PL12] PARK F. C., LYNCH K.: *Introduction to Robotics: Mechanics, Planning, and Control*. http://robotics.snu.ac.kr/fcp/, 2012. 5

[SJ03] SUGAR C. A., JAMES G. M.: Finding the number of clusters in a dataset. *Journal of the American Statistical Association 98:463* (2003), 750–763. 6

[SP95] STARK P. B., PARKER R. L.: Bounded-variable least-squares: an algorithm and applications. *Computational Statistics 10* (1995), 129–141. 5

[YKH04] YAMANE K., KUFFNER J. J., HODGINS J. K.: Synthesizing animations of human manipulation tasks. *ACM Trans. Graph. 23*, 3 (Aug. 2004), 532–539. 3

[YL12] YE Y., LIU C. K.: Synthesis of detailed hand manipulations using contact sampling. *ACM Trans. Graph. 31*, 4 (July 2012), 41:1–41:10. 3