Computers & Graphics 45 (2014) 40–50

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Special Section on Graphics Interaction

Using task efficient contact configurations to animate creatures in arbitrary environments

Steve Tonneau [a,*], Julien Pettré [b,1], Franck Multon [c,2]

[a] IRISA Rennes, 263 Avenue Général Leclerc, 35000 Rennes, France
[b] Inria
[c] Université Rennes 2 - Inria

ARTICLE INFO

Article history:
Received 27 June 2014
Received in revised form
27 August 2014
Accepted 27 August 2014
Available online 16 September 2014

Keywords:
Autonomous virtual characters
Animation for games
Procedural Animation
Contact Before Motion
Range Of Motion
Force Transmission Ratio

ABSTRACT

A common issue in three-dimensional animation is the creation of contacts between a virtual creature and the environment. Contacts allow force exertion, which produces motion. This paper addresses the problem of computing contact configurations allowing to perform motion tasks such as getting up from a sofa, pushing an object or climbing. We propose a two-step method to generate contact configurations suitable for such tasks. The first step is an offline sampling of the range of motion (ROM) of a virtual creature. The ROM of the human arms and legs is precisely determined experimentally. The second step is a run time request confronting the samples with the current environment. The best contact configurations are then selected according to a heuristic for task efficiency. The heuristic is inspired by the force transmission ratio. Given a contact configuration, it measures the potential force that can be exerted in a given direction. The contact configurations are then used as inputs for an inverse kinematics solver that will compute the final animation. Our method is automatic and does not require examples or motion capture data. It is suitable for real time applications and applies to arbitrary creatures in arbitrary environments. Various scenarios (such as climbing, crawling, getting up, pushing or pulling objects) are used to demonstrate that our method enhances motion autonomy and interactivity in constrained environments.

Research in computer animation is motivated by the need to provide virtual creatures with an increased autonomy of motion in 3D environments. Such improvements allow to propose new forms of gameplay in video games, or to validate ergonomic designs.

In this work we are interested in the contacts created between a creature and the environment: contacts allow to efficiently exert the force necessary to perform motion tasks (such as getting up, climbing or pulling). For instance in Fig. 13, several contacts are created between the end-effectors of a virtual insect and the books composing the environment.

Motion capture methods are inherently limited in such a constrained context: addressing various tasks and environments for different creatures requires the creation of prohibitively large motion databases. Therefore, a common approach is the decomposition of the motion into a sequence of contact configurations between a virtual creature and the environment. The notion of configuration is central in motion planning [1]. Such planners often use randomly generated configurations [2], and select those preserving static stability [3]. However, they lack heuristics to determine if those configurations are suited for the task in terms of force exertion. In the rest of the paper such configurations are called *task efficient*. Dynamic simulations use predefined configurations as inputs to motion controllers, but show little adaptation to the environment [4].

Thus, motion planners and dynamic controllers could benefit from a method to generate appropriate contact configurations. This is our problem statement, formalized in Section 2.

The key idea: The environment as a mean to exert a force: Contacts allow force exertion, which in turn produces the motion. Therefore to select a contact configuration, it is important to make sure it will allow to perform the task. For this reason we need heuristics to measure the compatibility of a contact configuration with a translational motion task. Examples of such tasks are pushing, pulling, standing up, or climbing. This set of motions is commonly needed by interactive simulations (such as videogames). They could benefit from our method to introduce more variety in the environments and interactions they propose. Rotational tasks will be addressed in future works.

To measure the task efficiency, we propose a heuristic inspired by the force transmission ratio [5]. It defines the efficiency of a configuration as the potential force it allows to exert in the direction of a translational task, as detailed in Section 2.4. It is traditionally used to optimize the configuration of a robotic arm, but requires to

* Corresponding author. Tel.: 06 71 30 36 68; fax: 02 99 84 71 71.
E-mail addresses: stonneau@irisa.fr (S. Tonneau), jpettre@inria.fr (J. Pettré), fmulton@irisa.fr (F. Multon).
[1] Tel.: 02 99 84 22 36; fax: 02 99 84 71 71.
[2] Tel.: 02 99 84 74 23; fax 02 99 84 71 71.

http://dx.doi.org/10.1016/j.cag.2014.08.005
0097-8493/

know in advance the future position of the end-effector. To overcome this issue, we combine our heuristic with a random sampling approach, independent from the environment (Section 3).

The sampling of all the possible joint configurations is performed offline to ensure good performance during the online simulation. However simply sampling to joint angles in a random manner may lead to unrealistic poses. To overcome this we propose to limit the sampling to a subspace corresponding to the Range Of Motion for each joint. Classical approaches consider minimum and maximum joint angle values while actual joint limits are more complex to model, as there may be interactions between joint axis – Fig. 7. In this paper we wish to model these interactions between joint axis to limit the sampling to natural configurations (Section 4).

Then, the samples are filtered online to select configurations in contact with the environment, and free of collisions. The chosen configurations are finally used as inputs for an inverse kinematics solver that will compute the final animation.

Therefore the contribution of this paper is a method for the real time, automated computation of task efficient contact configurations for arbitrary creatures. As shown in Section 5, it can be applied to various motions tasks in arbitrary environments. We discuss the limitations of our method, potential applications and future works in Section 6.

1. Related work

The issue of creating contact configurations has been addressed in different ways: Example-based methods use motion clips as references for motion (Section 1.1); Biomechanical and robotical approaches define relevant contact configurations by quantifying them in terms of force exertion (Section 1.2); Motion planning and optimization methods focus on contact configurations that preserve balance (Section 1.3).

1.1. Example-based methods for constrained environments

To improve the natural aspect of an animation, a common method consists in using motion clips, either created by an artist or obtained through motion capture. Effective methods exist to adapt those clips to the constraints of the environment such as external force pressure [6] or locomotion on uneven terrain [7,8]. Similarly foot-step planning techniques proposed hybrid approaches to address this issue [9,10].

Motion graphs [11,12] or precomputed search trees [13] can be used for acyclic motions, and be adapted for contact interaction in constrained environments.

Other methods address acyclic motions such as reaching and manipulating tasks [14,15], or close contact interaction motions [16].

However, methods based on motion capture do not easily apply to arbitrary virtual creatures.

Another drawback is that although motion adaptation is possible (for instance through inverse kinematics), the adaptation of a motion clip is limited to a motion including the same end-effectors in contact. This is problematic when the environment differs too much from the one used in the reference motion. To provide such methods with rich contact interactions for complex environments would require to be able to produce the animations corresponding to each possible interaction and appropriately choose between them at run time.

Conversely the generality of our method covers a large set of tasks, applies to any kind of virtual creature and adapts to the environment.

1.2. Inverse kinematics and manipulability for virtual creatures

The issue of optimizing a contact configuration for a task has been widely studied. Inverse kinematics methods exploit the redundancy of kinematic trees to optimize secondary objectives [17]. Yoshikawa presented the manipulability measure for quantifying the ability of robotic mechanisms in positioning and orienting end-effectors [18]. Based on this work, Chiu proposed the force transmission ratio, another index for optimizing a manipulator pose relatively to a specific task [5]. Several manipulability-based methods have since been proposed to either optimize a configuration [19] or a trajectory [20,21]. Recent works in biomechanics tend to show the relevance of the manipulability measure for human beings [22].

Those methods require *a priori* knowledge of the target that an end-effector must reach. They only solve half of our problem because we need to know where a contact must be created to find a suitable configuration.

Conversely, our method extends the force transmission ratio and uses it along with a random sampling approach. This allows us to address simultaneously the issues of finding a contact position and a task efficient configuration.

1.3. Motion planning and optimization for constrained environments

The advantage of procedural methods over example-based ones is that they are not limited by a motion database. Recently Wampler et al. proposed a method to automatically synthesize gaited motion for arbitrary creatures [23].

In [24], Kallman and Mataric generate a roadmap of configurations independent from the environment. The roadmap is updated at runtime as the environment is modified: colliding nodes are removed and paths consequently updated. Our method is also based on environment independent sampling, but the objective is different: Kallman and Mataric build a roadmap to address the issue of computing a collision-free motion; we do not build a roadmap and use sampled configurations as candidates for task efficient contact configurations.

Contact interactions have been considered for grasping tasks [25,26], or for motion planning problems. Hauser et al. introduced the *Contact before motion* approach [27], used in several other contributions [28,3,29]. A common drawback of this approach is that it requires prior discretization of possible contact positions in the environment. Also, task efficiency is not always considered in the process of finding contact configurations.

In the continuity of those works, Mordatch et al. proposed the Contact-Invariant Optimization (CIO) term [30]: contact positions and trajectory are planned simultaneously in the same optimization loop. Along the process, an end-effector is guided towards the nearest surface satisfying dynamic constraints. Al Borno et al. proposed a full-body trajectory optimization method that does not require explicit contact definition, but still requires to specify with which obstacle an effector should be in contact [31].

However, to get up from a chair in the environment shown in Fig. 8, a human would more likely put his hand on the table than on the chair, even if the table is farther away. Those methods cannot achieve this without requiring the user to explicitly define the table as an input of the problem (Fig. 8). Another drawback of those approaches is that, as for other planning methods, the computation time is too long for interactive simulations.

Other contributions in robotics have considered the quality of the contact configurations in their approach [32]. In particular Bretl et al. proposed a heuristic similar to the manipulability measure as a criteria for contact creation [33].

Our method lies in the continuity of these procedural approaches. It does not address the planning issue, rather the

problems of automatically finding better task efficient contact positions and configurations, while offering more flexibility than example-based methods [34].

1.4. Previous work and content of the extended version

This paper is an extension of [35] which was based on the design of a new method to compute task-efficient contact configurations for arbitrary virtual creatures. In this paper we focus on human motion. More specifically we aim at designing realistic joint limits. To do so we rely on recent works in biomechanics [36]. As a result, the reduced sampling space generates more human-like configurations. The changes related to this extension are detailed in Section 4. New results are shown in Section 5, and additional discussion is provided in Section 6. Also, in Section 3.3 we present the inverse kinematics solver we use to synthesize the resulting animation.

2. Problem statement

In this section we give several mathematical definitions to formulate our issue: How to rapidly compute a limb contact configuration, efficient for performing a given task? Fig. 2 provides an illustration for such definitions.

2.1. Kinematic representation of a virtual creature

A virtual creature is described by a kinematic structure A, with m end-effectors. We decompose A to treat each limb separately.

Definition of a limb: A limb $L_j, 0 \leq j \leq m-1$ is a kinematic sub-chain of A, comprising n rotational joints, and exactlyone end-effector e_j (Fig. 2 –blue rectangle). R_j denotes the 4×4 transformation matrix attached to L_j's root joint.

Limb configuration: A configuration Θ_j is a set of n angle values for each joint of the limb L_j. $p(\Theta_j) = (x_j\ y_j\ z_j)^T$ gives the position of the end-effector e_j for the configuration Θ_j, in world coordinates.

Jacobian matrix of a configuration: $J(\Theta_j)$ is the $3*n$ Jacobian matrix of L_j in the configuration Θ_j. $J(\Theta_j)^T$ is its transposed matrix. If $\theta^i,\ i = 1 \ldots n$ are the joint values of the configuration Θ then the Jacobian is defined as follows (the $_j$ indices are removed for clarity):

$$J(\Theta) = \begin{pmatrix} \frac{\partial x}{\partial \theta^1} & \cdots & \frac{\partial x}{\partial \theta^n} \\ \frac{\partial y}{\partial \theta^1} & \cdots & \frac{\partial y}{\partial \theta^n} \\ \frac{\partial z}{\partial \theta^1} & \cdots & \frac{\partial z}{\partial \theta^n} \end{pmatrix} \quad (1)$$

$J(\Theta_j)$ is computed using the method given in [37]. We also define $J_p(\Theta_j) = J(\Theta_j) * J(\Theta_j)^T$ as the product of the jacobian by its transpose; We call sample a the triplet $\langle p(\Theta_j), \Theta_j, Jp(\Theta_j) \rangle$.

2.2. Environment and contact interactions

The virtual creature moves in an environment W. W is composed of obstacles with which the creature interacts.

The environment as a set of obstacles: An obstacle is a planar surface $O \in W$ defined in the 3-dimensional Euclidian space (orange surfaces in Fig. 2). This definition of an obstacle is not restrictive since any complex three-dimensional object can be decomposed into a set of obstacles. $\mathbf{n_O}$ is the obstacle normal unit vector (orange arrow in Fig. 2).

Contact between a limb and the environment: We say that a configuration Θ_j is in contact regarding an obstacle set $E \subset W$ if

$$\exists O \in E, D(x_O, p(\Theta_j)) < \epsilon$$

where: x_O is the orthogonal projection of $p(\Theta_j)$ onto the obstacle O; D returns the Euclidian distance between two points; $\epsilon \in \mathbb{R}$ is small (red cylinders in Fig. 2).

2.3. Objective formulation

The motion task is expressed as a **unit** vector $\mathbf{v_t} \in \mathbb{R}^3$, expressed in R_j coordinates (black arrow in Fig. 2) for a limb L_j. $\mathbf{v_t}$ expresses a translational motion task for the root of the creature. Rotational tasks are not discussed in this work.

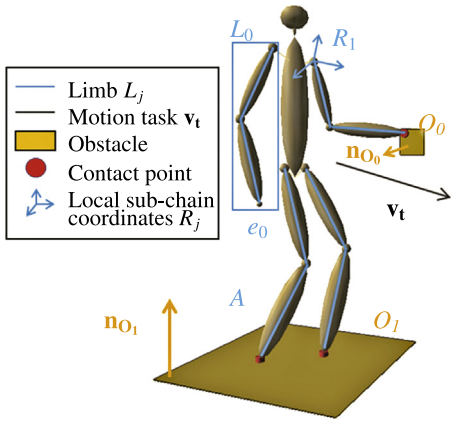

Fig. 2. Virtual human composed of 4 limbs. 3 end-effectors are in contact with 2 obstacles.

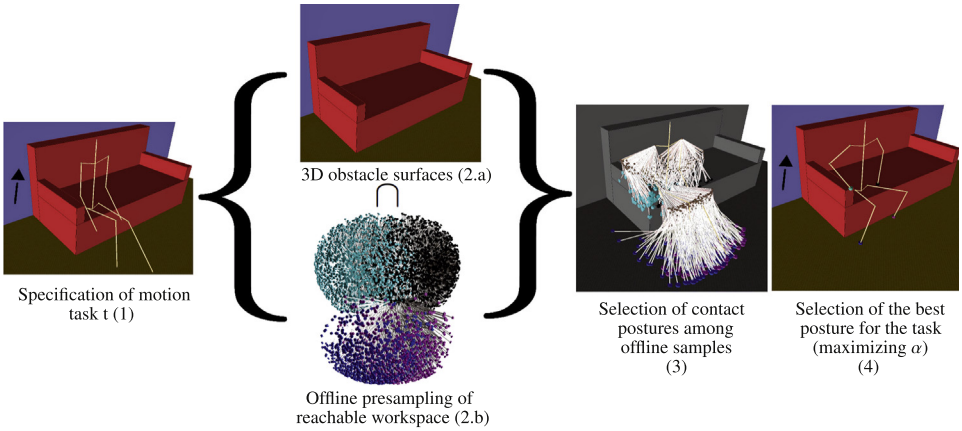

Fig. 1. Online step request. Given the task of getting up (1), We transpose the samples from our database into the local environment (2), and select the configurations in contact with the environment (3). Among these candidates, we select the collision-free configurations that maximize the heuristic α (4). For clarity the creature and samples are shown in a wireframe form.

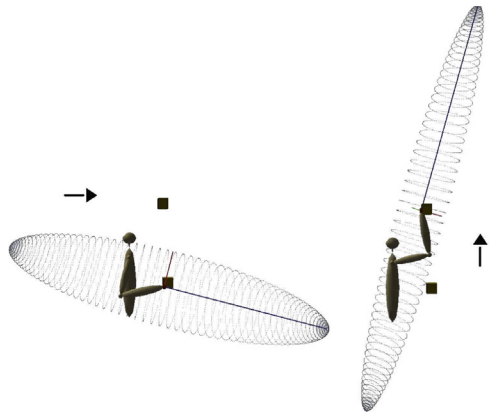

Fig. 3. The force manipulability ellipsoid for two different configurations. A longer axis means that a more important force can be exerted in the direction of the axis.

Task efficient configuration: We want to compute a configuration *Theta*$_j$ with the three following properties: Θ_j is in contact with an obstacle O_i of W; Θ_j is collision-free (no interpenetration) and does not violate eventual joint limits; Θ_j is suitable for the task $\mathbf{v_t}$, according to a heuristic α that must be provided. Computation time should be compatible with real time interactive simulations.

2.4. A heuristic for task efficient contact configurations

We want a heuristic α to answer the following question: How appropriate is a configuration Θ_j regarding $\mathbf{v_t}$? We make the hypothesis that for a subset of the possible motions, $\mathbf{v_t}$ will be satisfied more easily if the end-effector can exert a high force in the direction of $\mathbf{v_t}$. This makes sense for the tasks we are addressing, such as pushing a cupboard, which might require an important effort. We propose to use that potential force as a heuristic for task compatibility. Previous works in robotics [5] showed that this potential force can be quantified by computing, for a given configuration, the force transmission ratio f_T regarding $\mathbf{v_t}$:

$$f_T(\Theta_j, \mathbf{v_t}) = [\mathbf{v_t}^T (J(\Theta_j) J(\Theta_j)^T) \mathbf{v_t}]^{-1/2} \quad (2)$$

The force manipulability ellipsoid provides an intuitive representation of the different values taken by the force transmission ratio, as shown in Fig. 3 [18]. The value $f_T(\Theta_j, \mathbf{v_t})$ corresponds to the length of the ellipsoid in the direction $\mathbf{v_t}$. According to f_T in those two examples it appears that the lower obstacle is more suited for a horizontal task, when the upper obstacle is more suited for a vertical task.

We extend the force transmission ratio to use it as a heuristic for contact location. We consider that Θ_j is in contact with an obstacle O_i. We weigh f_T with the dot product between the task $\mathbf{v_t}$ and the normal $\mathbf{n_{Oi}}$ of O_i.

$$\alpha(\Theta_j, \mathbf{v_t}) = f_T(\Theta_j, \mathbf{v_t}) \mathbf{v_t} \cdot \mathbf{n_{Oi}} \quad (3)$$

If we maximize α, obstacles with normals collinear to the motion task will be advantaged for contact creation. This is coherent with our problem because it verifies that force exertion is actually applied against the obstacle, as shown in Fig. 8.

Also, we can see that α can take negative values. This is interesting especially for pushing and pulling tasks, as shown in Section 5.2.

In the results shown in this paper α is the only heuristic used. Its integration with other heuristics is discussed in Section 6.

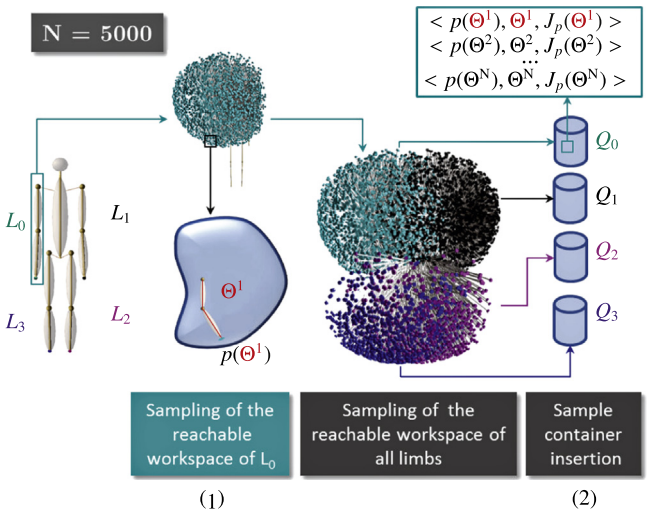

Fig. 4. Illustration of the environment-independent offline sampling for $N=5000$, for the right arm first, then for all the limbs. A sample container is created for each limb. An entry contains a configuration Θ, and its jacobian product J_p. Entries are indexed by the end-effector position $p(\Theta)$. For clarity the samples are shown in a wireframe form.

3. Computation of task efficient contact configurations

The definitions given in Section 2 allow us to describe our method in mathematical terms. Our contribution lies in the proposition of a method to generate and compare contact configurations according to a motion task. For efficiency reasons, the algorithm is decomposed in two steps:

Offline sampling step: The first step is independent from the environment. A large set of arbitrary configurations Q_j is randomly generated for each limb L_j –Fig. 4–. Precomputation is made for each configuration to accelerate run time performance.

Online request step: The second step consists in a request performed on the configuration set Q_j. The configurations that, in the current situation, are in contact with the environment W, will be selected as potential solutions. Among those we select the configuration for which our heuristic α gives the highest score – Fig. 1. In this paper this configuration is used as an input to an inverse kinematics solver in order to produce the final animation.

3.1. Offline generation of random limb configurations

This step is independent from the environment, and thus only has to be run once for each limb L_j composing our creature A. Fig. 4 illustrates the sample configuration generation process. As inputs, we take a number of samples N and a limb L_j. We fill a sample container Q_j with the sample configurations of L_j by repeating N times four steps:

Random generation of a configuration Θ_j –Fig. 4 (1)–: This is done by generating a random angle value for each joint of L_j. The value of the angle is restricted by a range of motion (ROM) to avoid obtaining unnatural poses. This generation step is further discussed in Section 4.

Computation of the jacobian product $Jp(\Theta_j)$: The jacobian matrix $J(\Theta_j)$ is computed and multiplied by its transpose $J(\Theta_j)^T$ to obtain the jacobian product $Jp(\Theta_j)$, which is needed for the run time computation of the extended force transmission ratio α. It can be computed directly from Θ but its computation is expensive. Storing it results in the additional storage of N matrices of size 3∗3 in memory. However it improves the online performance because it avoids computing it multiple times and reduces the computation of α to two simple matrix products.

Computation of $p(\Theta_j)$ –Fig. 4 (1): The end-effector position $p(\Theta_j)$ is expressed in the limb coordinates R_j. Storing $p(\Theta_j)$ allows the implementation of the sample container Q_j as a data structure efficient regarding proximity requests that will be performed at run time.

Insertion of the resulting sample in Q_j –Fig. 4 (2): We create the sample denoted by the triplet $\langle p(\Theta_j), \Theta_j, Jp(\Theta_j)\rangle$, and store it into the sample container Q_j.

As stated earlier, the generation of sample configurations has to be performed for every limb composing our creature A. For instance, for a virtual human, we would end up with four sample containers (one for each arm, and one for each leg). We cannot use a single tree for both arms because the joint limits differ symmetrically in our model.

The appropriate value for N is discussed Section 5.4. The sampling method is discussed in Section 6.

3.2. Online computation of task efficient contact configurations

We consider the motion task $\mathbf{v_t}$ –Fig. 1 (1): *the black arrow indicates the task of getting up*–. To find a contact configuration for a limb L_j that is efficient for $\mathbf{v_t}$, we proceed in four steps:

Identification of the reachable obstacles: We retrieve the obstacle set $E \subset W$ of obstacles potentially reachable by the limb L_j –Fig. 1 (2.a): the sofa, the ground *and the wall*–. E is the result of a collision detection query between the environment and a sphere S_j centered at the root of L_j. The radius of S_j is defined as the length of L_j.

Selection of the samples in contact: We request Q_j for all the samples that are in contact with an obstacle of E –Fig. 1 (2.b)–. The result of the query is a list of limb configurations $Q_j^{contact} \subset Q_j$ –Fig. 1 (3): *Selection of configurations in contact with the sofa and the ground*–.

Ordering of the candidate samples: We sort the samples of $Q_j^{contact}$ using our heuristic $\alpha(\Theta_j, \mathbf{v_t})$. This means that the first sample of $Q_j^{contact}$, that we call Θ_j^{max}, verifies:

$$\forall \Theta_j \in Q_j^{contact}, \alpha(\Theta_j, \mathbf{v_t}) <= \alpha(\Theta_j^{max}, \mathbf{v_t}).$$

This is the configuration that is the most appropriate for the task regarding the extended force transmission ratio.

Selection of the best collision-free sample. We perform a collision check between the environment and Θ_j^{max}. If Θ_j^{max} is free of collision, it is returned as the solution configuration. Otherwise, we keep iterating through the sorted configurations of $Q_j^{contact}$ until we find a configuration Θ_j^{target} free of collisions. If all the configurations of $Q_j^{contact}$ are colliding, no configuration is returned –Fig. 1 (4): *our method places the right hand on the armchair, both feet on the ground, close to the root, and the left hand on the sofa*–.

3.3. Computation of the resulting animation

The selected contact configuration Θ_j^{target} is used as a target for a simple animation system –Fig. 5 (1) and (2). To achieve the animation two steps are necessary:

First we compute a trajectory between the end-effector current and target with Bézier curves –Fig. 5 (3). Then, an inverse kinematics solver is used to guide the end-effector along the trajectory –Fig. 5 (4).

We use the method proposed in [17] which handles efficiently joint limits and makes it possible to specify secondary constraints minimized along the trajectory.

In order to reach not only the position, but also the configuration Θ_j^{target}, we use a secondary constraint h formulated as the euclidian distance between the current configuration and the target configuration:

$$h(\Theta_j^{current}) = \|(\Theta_j^{target}) - (\Theta_j^{current})\| \quad (4)$$

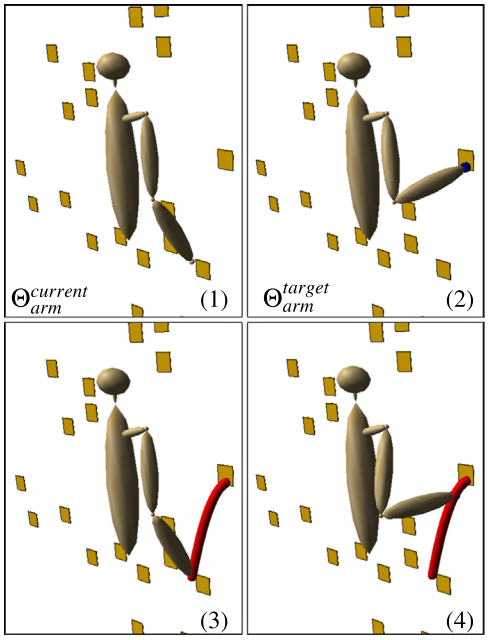

Fig. 5. Animation between the current configuration $\Theta_{arm}^{current}$ and the target configuration $\Theta_{arm}^{current}$. (3): A trajectory is computed between the end-effector positions of both configurations. (4): An inverse kinematics solver moves the end-effector along the trajectory.

With this animation system we show that our method is able to generate task efficient configurations within time limits acceptable for real time applications. The offline and online steps are automatic and do not require manual editing.

4. Representation and sampling of the range of motion (ROM)

In order to avoid generating unnatural results, the configurations of a limb are sampled inside a boundary called Range Of Motion (ROM). A ROM represents the possible angle values a degree of freedom can take in a given situation. In Section 4.1 we explain how we determine the ROM of a limb. In Section 4.2 we show how we generate limb configurations inside their corresponding ROM.

4.1. Determination of the range of motion (ROM)

Usually, the Range Of Motion (ROM) of an articulation is determined using joint limits [38]. When using joint limits, the angle value θ^i a degree of freedom can take is bounded to an interval $[\beta_i, \gamma_i]$. For the three degrees of freedom of the shoulder for instance, the resulting ROM has the shape of a parallepiped rectangle –Fig. 7 bottom row–. However the actual ROM of the shoulder is more complex and restrictive –Fig. 6–, because of the dependencies that exist between the degrees of freedom, as shown in [39]. Using joint limits can result in unnatural configurations, or, if the limits are too restrictive, in the rejection of natural configurations. For the case of a human, we address this issue by determining a more accurate ROM for the complex shoulder and hip articulations. To do so we use the protocol established by Haering et al. [36]. Using a motion capture system, we record hip and shoulder motions of maximal amplitude. From the obtained data, we reconstruct the 3D joint kinematics, and determine the angular configurations of the studied articulation at

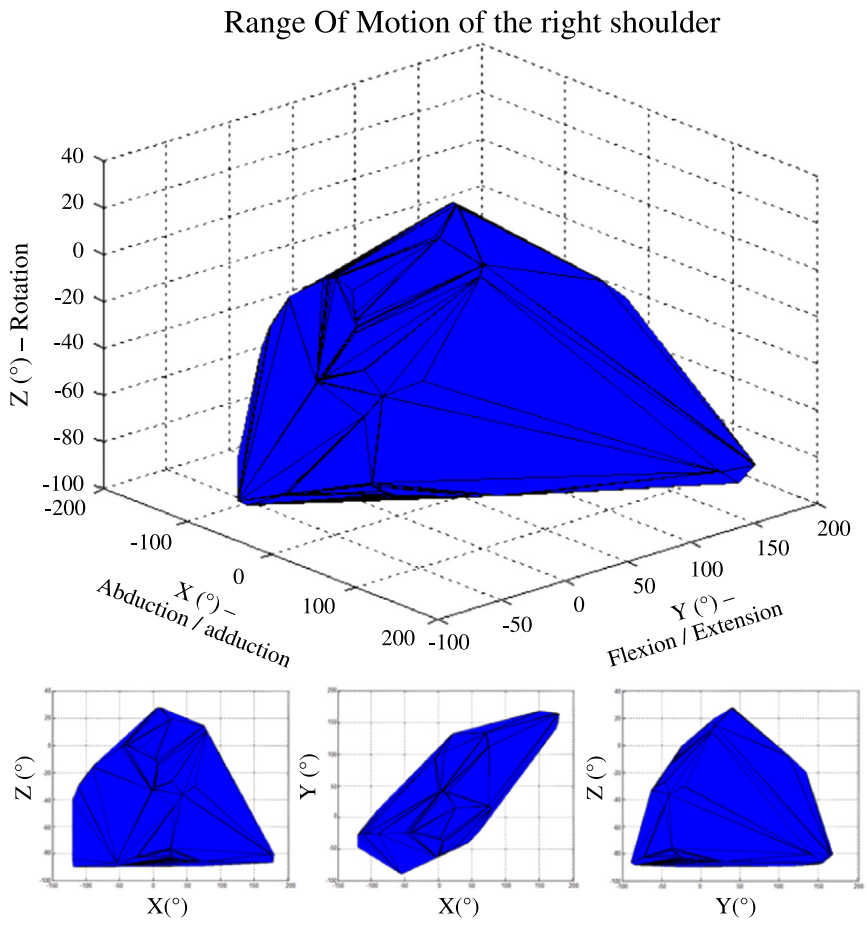

Fig. 6. Representation of the Range Of Motion of the first author's shoulder. The blue volume is the non convex hull $K_{Shoulder}$ including all the shoulder configurations that were recorded in a motion capture session, following the *YXZ* euler angle decomposition.

each frame. Each configuration can be seen as a three-dimensional point, where one coordinate describes the value of one Euler angle –Fig. 6.

We then compute the 3D non-convex hull K including all the recorded angular configurations. This is achieved using the method proposed in [40]. K_{Hip} and $K_{Shoulder}$ represent the Range Of Motion (ROM) of the hip and shoulder. This means that any configuration included in K_{Hip} (respectively $K_{Shoulder}$) is a configuration of the hip (respectively the shoulder) that is valid for a human.

In this work, the protocol was applied on a single male subject, therefore the computed K_{Hip} and $K_{Shoulder}$ are specific to him. Haering et al. define a normalized ROM by including 3D poses common to a maximal number of participants into a hull of average volume. Therefore, the user of the method has the choice between using an average ROM fitting any virtual human or using one specific to a given morphology.

4.2. Generation of samples for a limb

During the offline phase of our method, we sample configurations within the joint limits of a limb. For the case of the human arm and leg, we discard configurations for which the shoulder and hip angular values do not belong to their respective ROM. Therefore the generation of a sample configuration is done in two steps:

Determination of the joint limits: For each degree of freedom θ_i, we identify the minimum and maximum angle values β_i and γ_i that θ_i can take. For the hip and shoulder, those values are deduced from the bounding box of K_{Hip} and $K_{Shoulder}$ –Fig. 7. For other

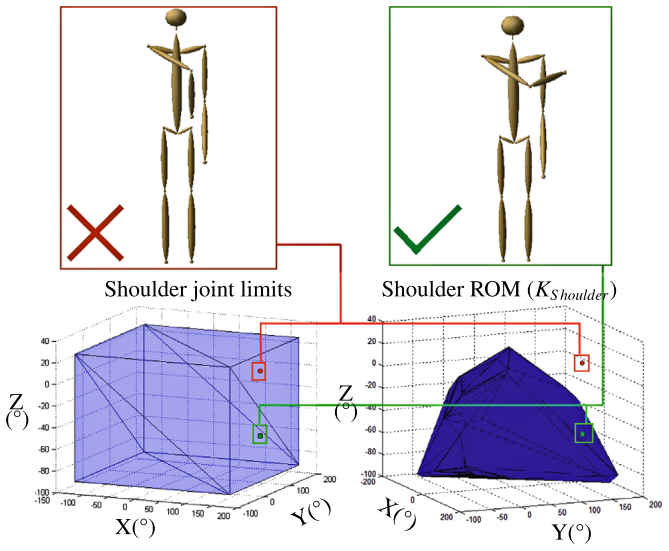

Fig. 7. Generation of two sample configurations. Up: The two configurations lie within the joint limits of the shoulder, as shown in the bottom left plot. However the red configuration is rejected because it does not belong to $K_{Shoulder}$ (bottom right).

articulations which use simple joint limits (such as the elbow angle) we define them manually.

Angle values generation and validation: We randomly generate an angle value $\theta^i \in [\beta_i, \gamma_i]$ for each degree of freedom composing the limb. We reject unnatural arm and legs configurations that are not included in their respective Range Of Motion. In Fig. 7, the

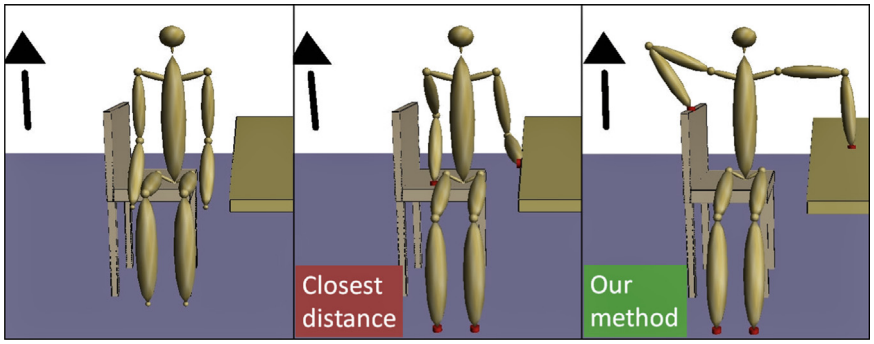

Fig. 8. Our method (right) is compared with the closest distance heuristic (middle) in this example of getting up from a chair. In the latter case, the left hand position (on the side of the table) is not suitable to generate a vertical effort.

unnatural red configuration lies within the fixed joint limits of the shoulder. Using an accurate ROM allows its rejection because it does not belong to $K_{Shoulder}$.

This procedure makes it possible to efficiently sample configurations in a more accurate ROM for virtual humans.

5. Results

We designed several scenarios to demonstrate the benefits of our method, through the variety of creatures and environments that were designed. In this section we give implementation details on those scenarios. We then detail each scenario and comment the results obtained. The section is concluded with a performance analysis.

5.1. Implementation details

In order to allow for a fast and efficient search among the configurations, we implemented Q_j as an octree data structure that offers support for spatial queries. The position of the end-effector $p(\Theta_j)$ is used as an index in Q_j.

The test application was developed using the C++ language. One limitation of our current implementation is that collision checks are only tested against the environment and not between limbs. This is not a limitation of the method and will be corrected in future work.

Environments are described in the obj format, while virtual creatures and scenarios are described using custom xml files. Rendering is achieved using the OpenGl API. No other third-party libraries were used. We performed the runs on a laptop with an Intel Core i7-2760QM 2.40 GHz processor and 4 GB of memory. The application is not multi-threaded.

5.2. Test scenarios

We consider a virtual creature in a constrained environment. Six coordinates describe the position and orientation of its root. By default the chosen initial limb configuration is the reference posture of the creature (as shown for instance in Fig. 9). We consider a directional task, and one or several limbs of the creature. We then use our method to compute a task efficient contact configuration. Each scenario is also illustrated in the companion video.

Standing up –*Figs.* 1 and 8: The environment is composed of a sofa, or a chair and a table. The creature is a virtual human (Fig. 2). In the initial configuration the human is sitting on a sofa or a chair. We formulate the task of getting up as a vertical vector. These examples show the adaptability of our method: the same task in different environments results in different configurations that take advantage of it.

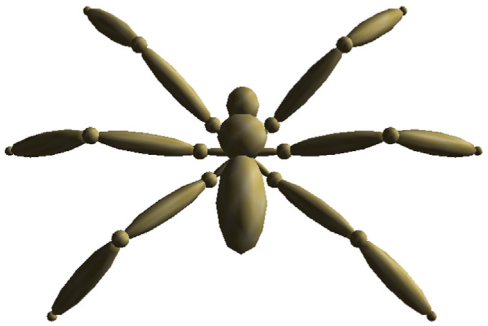

Fig. 9. Reference posture of a virtual insect composed of 6 limbs. Each limb is composed of 5 degrees of freedom.

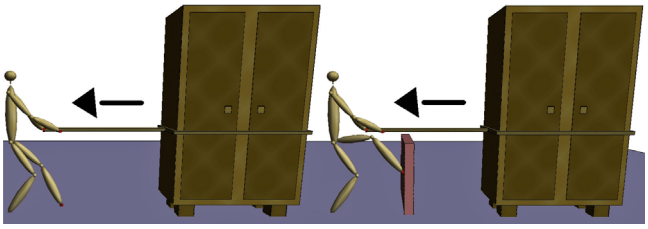

Fig. 10. Configurations found for a pulling task. In the right figure, our creature uses the pink wall as a better support for the foot. The asymmetry between the arm configurations is induced by the sampling phase.

Multi-limb creatures in constrained environments (video): The environment is composed of a challenging set of books placed on a bookshelf. The creature is an insect with six limbs (Fig. 9). The input is a forward directional task. This example shows that our method is generic and can be applied to arbitrary creatures, as opposed to example-based approaches.

Pushing and pulling objects –*Figs.* 10 and 11–: The creature is a virtual human. Two environments are used: In the pushing scenario, the environment consists in a cupboard and the ground; in the pulling scenario it consists in a cupboard, the ground, as well as a rope attached to the cupboard and a small wall. The task consists in pulling (pushing) the cupboard. We formulate the task as a horizontal vector, and compute task efficient configurations for the arms and the left leg of the human. The right foot is already in contact. To push objects it is preferable to create contacts on surface the normals of which are opposite to the pushing direction. Therefore in this case we use a different heuristic $\alpha_{push}(\Theta_j, \mathbf{v_t}) = -\alpha(\Theta_j, \mathbf{v_t})$.

In Fig. 12, the task remains to push the cupboard, but this time the character's back is facing the cupboard. Thanks to a more

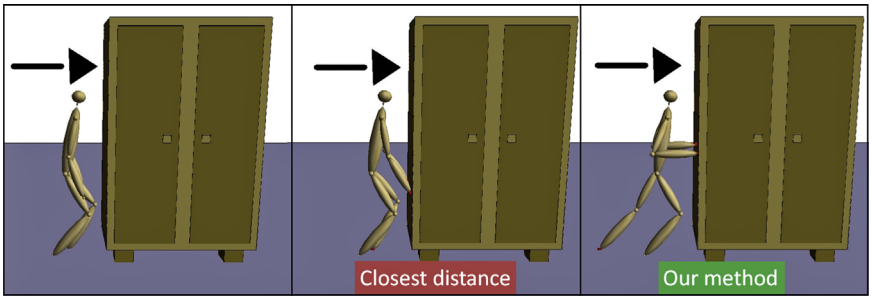

Fig. 11. Our method (right) is compared with the closest distance heuristic (middle) in this example of pushing a cupboard. The closest distance heuristic places the hands and left feet at locations close to their original positions (left) while our method places the end-effectors in configurations relevant for the pushing task.

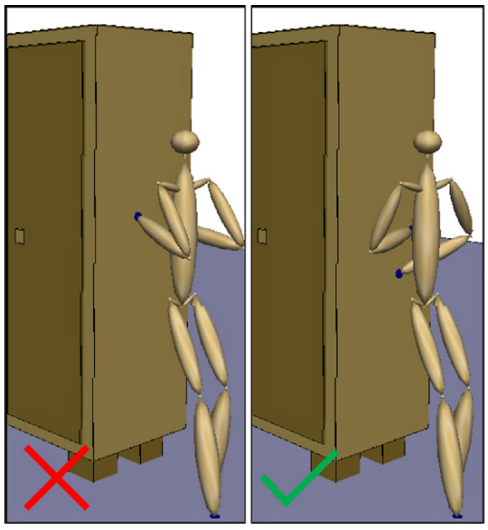

Fig. 12. A more precise definition of the Range Of Motion of the human arm gives more natural results. In this example the character has to push the cupboard but its back is facing it. The configuration on the left is unnatural because the shoulder rotation angle is not compatible with the abduction angle. It is rejected in favor of a more human-like configuration (right).

accurate definition of the ROM (Section 4), undesired configurations (left) are rejected in favor of human-like configurations (right). These examples show that our method can be applied to pushing and pulling tasks, enhancing the autonomy of motion of our virtual characters.

Computation of a sequence of task efficient contact configurations –Fig. 13+climbing example in video–: A virtual creature is set in an environment in its reference configuration. Given a trajectory for the creature root, we use our method to compute a configuration sequence along the trajectory. The first configuration computed is given as an input to compute the second one, and so on. These examples show how a simulation can interact with our method to obtain target configurations and eventually synthesize motion.

5.3. Comparison against the closest distance heuristic

Comparing the results obtained by our method is not trivial because few methods perform the real time automatic computation of contacts: Several previous contributions only address cyclic motions such as walking [7,8]. Hauser et al. manually predefine the set of possible contacts [27]. Bretl et al. use a form of manipulability integrated in a motion planner [33]. Mordatch et al. use a closest distance approach as part of an optimization loop that takes several minutes to compute a result [30]. Therefore we choose to compare the results we obtained with this closest distance heuristic.

In Fig. 8 the environment consists of a chair, the ground and a table. We compare our method with the closest distance heuristic. The configuration of the left arm in particular seems more appropriate to generate a vertical effort with our method.

In Fig. 11 the environment consists of a cupboard and the ground. The task for a virtual human is to push the cupboard. In the middle we can see that the results provided by the closest distance heuristic are highly determined by the original location of the end-effectors. Our method, on the other hand, creates contact configurations relevant for the pushing task.

In Fig. 14 the environment is a climbing wall. The creature is a virtual human (Fig. 2). The initial configuration is the reference posture –Fig. 14 (middle)–. The task consists in navigating along the wall in arbitrary directions. This example shows the advantage of our method over heuristics such as the closest distance, because the selected configurations vary according to the motion task.

5.4. Performances

Three parameters play a role in the performances offered by our method: the number N of samples generated during the offline step, the number of obstacles reachable by the limb when the method is called at run time, and the number m of limbs composing our creatures.

We only have control over the number of samples N, so we focus on this variable. We are interested in finding a value for N that will be as low as possible while maintaining an acceptable quality in the results obtained.

We have observed that in our scenarios, the number of samples N has a limited influence on the average maximum value of α. Therefore the main variables of interest are the number N of samples generated and the number of candidate configurations found consequently. Table 3 shows the time spent during the offline step relative to the number of samples generated. It is interesting to note that even for $N=100000$, the generation time is acceptable, since this step is only performed once. Parallelization and code optimization could probably allow to obtain better performances, but the interest of doing this is limited. Table 1 shows the time spent for one call to our method. Table 2 presents the average number of contact candidates returned by a spatial request.

We observe that for $N \leq 10\,000$, the computation time is short and the average number of candidates is satisfying. The human climbing scenario is an exception: a higher number of samples is necessary to find enough contact candidates. This is because the environment is composed of a small set of small obstacles.

Looking at the worst performances, we observe a correlation between the time spent in the method and the maximum number of hits obtained. This is explained by the growing number of requests that must be made. In each scenario, the number of triangles is about the same (a hundred); the performance variation is explained by their spatial distribution. For the getting up

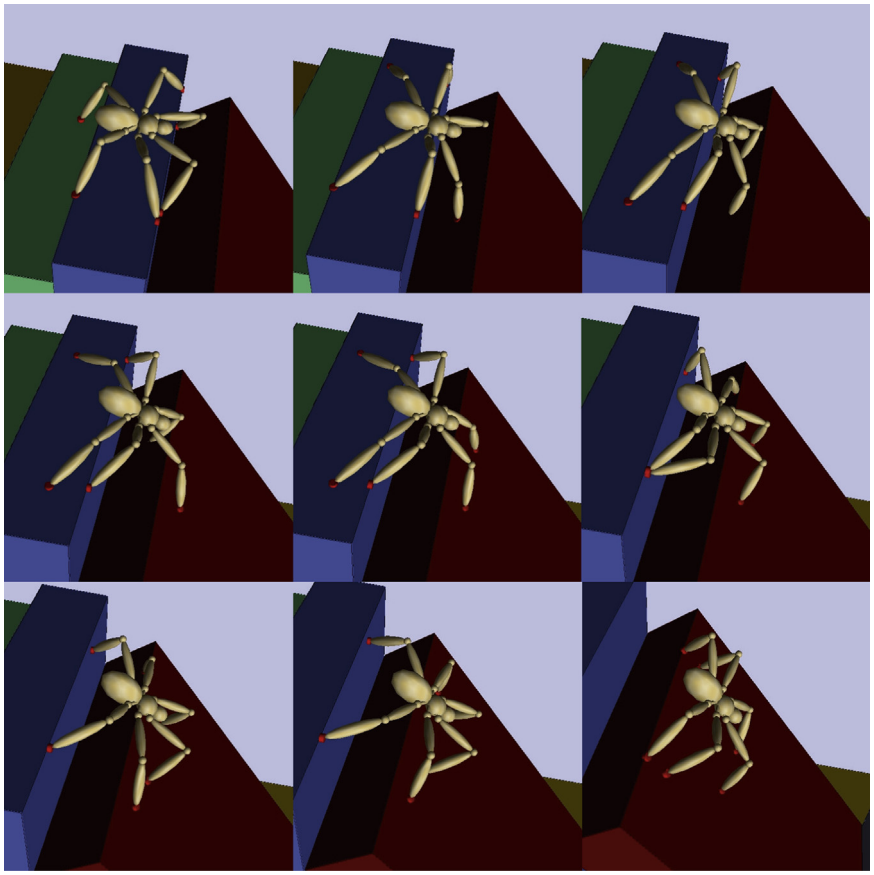

Fig. 13. Configuration sequence for an insect with 6 limbs crossing a bookshelf. Task efficient contact configurations are found along the trajectory.

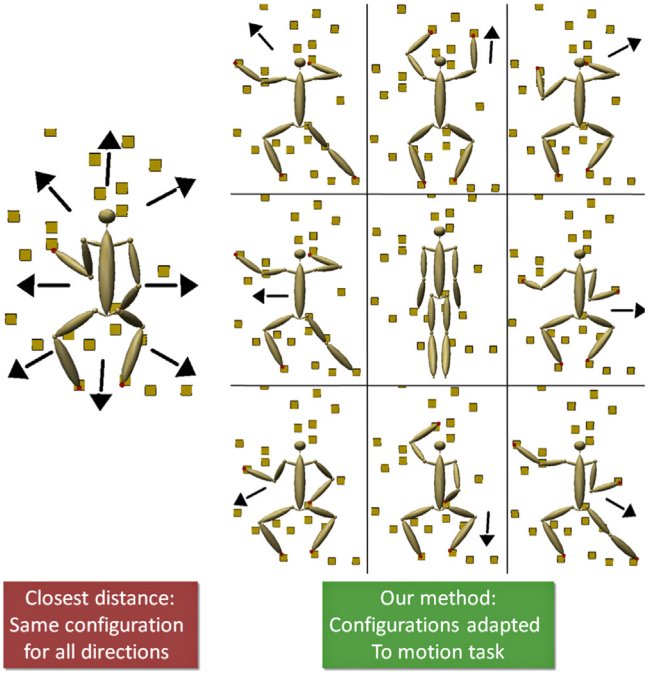

Fig. 14. Configurations for a humanoid on a climbing wall. Left: the closest distance heuristic does not consider the motion task, therefore it always computes the same configuration. Right: From the same initial root location (position and orientation), different configurations are computed depending on the task (black arrow).

scenario for instance, setting $N=1000$ is a reasonable choice, where a value of $N=100\,000$ seems more appropriate in the human climbing scenario. This can be explained by the limited range of motion of the insect limbs, so that a smaller amount of samples is sufficient to cover correctly the reachable workspace of the limb.

Under the appropriate conditions on the number of samples, we observed that the framerate never went below 52 fps even in the worst case scenarios.

Finally, we observe that the memory occupation grows linearly with the number of samples, and remains in reasonable ranges (from 2 MB for 10 000 samples to 166 MB for 1 000 000 samples).

Table 1
Average time (worst time) (in ms) spent for a call to our method relative to the scenario and the number of samples N.

	$N=1000$	$N=10\,000$	$N=100\,000$
Human climbing	**1** (3)	**2** (6)	**3** (30)
Getting up	**4** (7)	**5** (60)	**154** (856)
Insect locomotion	**1** (4)	**8** (40)	**55** (370)
Pushing/pulling	**1** (1)	**5** (6)	**34** (70)

Table 2
Average number of contact configurations found relative to the scenario and the number of samples N.

	$N=1000$	$N=10\,000$	$N=100\,000$
Human climbing	0	1	26
Getting up	41	49	3442
Insect locomotion	20	312	2553
Pushing/pulling	13	142	1387

Table 3
Average time (in ms) spent generating samples relative to the number of samples N.

	$N=1\,000$	$N=10\,000$	$N=100\,000$
Generation time	18000	1500	256

6. Discussion

In this section we review the main limitations of our method and future work, before concluding.

6.1. Limitations

The method is not probabilistically complete: A probabilistically complete method would have the following property: if a solution exists, the probability of finding it as the running time goes to infinity is 1. Our method generates N samples offline used as inputs for the contact queries. Due to this approach, the method is not probabilistically complete. This means that even if a solution exists, our method might fail to find it. Maintaining this property would involve performing regularly new sampling steps at run time. We choose to lose this property in favor of real time performance.

The method does not integrate dynamic constraints: The method assumes a non-dynamic environment. For instance linear and angular velocities are not taken into account, nor are gravity and balance. To extend the method to a wider range of motions, additional efforts will be necessary to integrate those parameters. This would make it possible to predict the future positions of moving objects and create contacts with them.

However, we believe that even though our method does not integrate physical parameters yet, interactive applications such as video games can already benefit from it, in a way similar to the locomotion system proposed in [7].

The accurate definition of the Range Of Motion does not apply to all creatures. In this paper we experimentally determine the ROM of the human limbs, which allows us to obtain more natural configurations. Currently for other virtual creatures this approach is not possible; the current solution is to use simple linear joint limits. It would be interesting to propose an intuitive tool to design and test the ROM of arbitrary limbs.

Performance issues: As the complexity of the environment rises, performance can become an issue because of the important number of requests that must be run. Fortunately, as seen in Section 5.4 it is possible to adjust the number of samples N to reduce the number of requests. However, if there are too many obstacles a trade-off between accuracy and abstraction of the environment should be found to maintain a reactive simulation.

6.2. Future work

Our next step is to integrate the method within existing solutions to further demonstrate its interest. We are also working on several improvements on the method itself.

Dynamic simulation integration: The method can be integrated within an existing physical animation framework, as a complementary tool used only to handle constrained situations. In open situations classical controllers could still be used in this hybrid system. To automatically determine if the local environment is too constrained for a classical approach, we could use a measure such as the one suggested by [41].

Motion planner integration: Offline motion planners can benefit from the method to avoid the manual description of potential contact points. They can also use it as an additional term to an optimization problem, allowing better results in terms of task compatibility at a small cost.

Global posture computation: As of today, in our method the motion task is the same for every limb. However, designing a high level controller to decompose a global motion task into subtasks for each limb would allow us to obtain more accurate results.

Also, currently we do not integrate the fact that actuating several limbs at the same time could result in undesired torques on the body in the case of a dynamic simulation. Therefore we want to combine the method with a complementary global posture optimization technique [42]. Doing this would allow us to optimize the whole body according to the computed limb configurations, and produce more natural results.

Additional heuristics for configuration selection: Combining our heuristic with other criteria could help us obtaining more natural results. Specifically, we would like to integrate an additional comfort criterion for virtual humans. Using the precise representation of the range of motion K of the human shoulder and hip, we could select in priority configurations that lie far from the boundaries of K. The validity of this hypothesis as well and the simultaneous integration of both heuristics will be addressed in future works.

Validating the extended force transmission ratio: The method uses the extended force transmission ratio, based on the manipulability measure. A biomechanical study showed that it is effectively optimized by humans performing grasping tasks with their upper limbs [22]. However, this cannot be demonstrated for all the possible motion tasks, or for non human morphologies. We would like to experimentally validate the application of our heuristic to arbitrary limbs. To do so we intend to conduct a perception study to determine if the results produced by our method are perceived as natural by users.

We also want to improve the heuristic. Several options exist: Treating rotational efforts would allow us to address a larger number of tasks; We would also like to integrate more complex contact and friction models; Lastly, combining the method with other classic ones such as dynamic balance will allow us to obtain more natural results.

A smarter sampling step: The configuration samples of a limb are generated randomly in its reachable workspace. Uniform sampling did not allow us to obtain better results. As explained in Section 5, reducing the number of samples could be interesting for performance. If the task is known, a possible improvement is to design a "task-oriented" sampling that would generate more samples around configurations known to be "good" for a task, in a way similar to [43]. This would probably reduce the number of samples necessary because it would limit the generation of samples in uninteresting areas.

6.3. Conclusion

In this paper we introduced a method to compute task efficient contact configurations for arbitrary creatures in 3D environments. It combines a sampling approach with a heuristic to evaluate the relevance of a configuration for a task. The sampling step is performed offline for enhancing performance, is automatic and independent from the environment. A precise representation of the Range Of Motion of the human limbs is used to reject unnatural configurations and enhance more human-like results. The method is designed to provide contact configurations to motion planners and animation frameworks. It is suitable for real time applications.

Experimental results show that the method can successfully address a large variety of tasks in various constrained environments. Thus it enhances the autonomy of motion and the interactivity proposed by simulations.

Future work will focus on integrating the method within existing frameworks.

Appendix A. Supplementary material

Supplementary data associated with this article can be found in the online version at http://dx.doi.org/10.1016/j.cag.2014.08.005.

References

[1] Esteves C, Arechavaleta G, Pettré J, Laumond J-P. Animation planning for virtual characters cooperation. ACM Transactions on Graphics 2006;25 (2):319–39. http://dx.doi.org/10.1145/1138450.1138457.

[2] Lozano-perez T. Spatial Planning: A Configuration Space Approach c(2).

[3] Escande A, Kheddar A, Miossec S, Garsault S. Planning support contact-points for acyclic motions and experiments on HRP-2. In: Khatib O, Kumar V, Pappas GJ, editors. ISER, Springer Tracts in Advanced Robotics, vol. 54. Springer; 2008. p. 293–302.

[4] Yin K, Loken K, van de Panne M. Simbicon: simple biped locomotion control. ACM Transactions on Graphics 2007;26(3) Article 105.

[5] Chiu S. Control of redundant manipulators for task compatibility. In: Robotics and Automation. Proceedings. 1987 IEEE International Conference on, vol. 4, 1987, pp. 1718–1724. doi: http://dx.doi.org/10.1109/ROBOT.1987.1087795.

[6] Coros S, Karpathy A, Jones B, Reveret L, van de Panne M. Locomotion Skills for Simulated Quadrupeds. ACM Transactions on Graphics 2011;30(4) Article TBD.

[7] Johansen RS. Automated semi-procedural animation for character locomotion. Aarhus Universitet, Institut for Informations Medievidenskab; 2009.

[8] Levine S, Popovic J. Physically Plausible Simulation for Character Animation. In: Symposium on Computer Animation, 2012, pp. 221–230.

[9] Choi MG, Lee J, Shin SY. Planning biped locomotion using motion capture data and probabilistic roadmaps. ACM Transactions on Graphics 2003;22 (2):182–203. http://dx.doi.org/10.1145/636886.636889.

[10] Kanoun O, Laumond J-P, Yoshida E. Planning foot placements for a humanoid robot: a problem of inverse kinematics. Int. J. Rob. Res. 2011;30(4):476–85. http://dx.doi.org/10.1177/0278364910371238.

[11] Kovar L, Gleicher M, Pighin F. Motion graphs. In: ACM Transactions on Graphics, vol. 21, ACM, New York, NY, USA, 2002, pp. 473–482. doi: http://dx.doi.org/10.1145/566570.566605.

[12] Lee J, Lee KH. Precomputing avatar behavior from human motion data. In: Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '04, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 2004, pp. 79–87. doi: http://dx.doi.org/10.1145/1028523.1028535.

[13] Lau M, Kuffner JJ. Precomputed search trees: planning for interactive goal-driven animation. In: Symposium on Computer Animation, 2006, pp. 299–308.

[14] Yamane K, Kuffner J, Hodgins JK. Synthesizing Animations of Human Manipulation Tasks, ACM Trans. on Graphics (Proc. SIGGRAPH 2004).

[15] Kallmann M, Aubel A, Abaci T, Thalmann D. Planning Collision-Free Reaching Motions for Interactive Object Manipulation and Grasping. Computer graphics Forum (Proceedings of Eurographics'03 2003;22(3):313–22.

[16] Ho ES, Komura T. Indexing and retrieving motions of characters in close contact. IEEE Transactions on Visualization and Computer Graphics 2009;15 (3):481–92 doi: doi:10.1109/TVCG.2008.199.

[17] Baerlocher P, Boulic R. An inverse kinematics architecture enforcing an arbitrary number of strict priority levels, The Visual Computer 20(6). doi: http://dx.doi.org/10.1007/s00371-004-0244-4.

[18] Yoshikawa T. Analysis and Control of Robotics Manipulators with Redundancy 1984.

[19] Naksuk N, Lee CSG. Zero moment point manipulability ellipsoid. In: ICRA 2006 Proceedings, 2006, pp. 1970–1975. doi: http://dx.doi.org/10.1109/ROBOT.2006.1641994.

[20] Guilamo L, Kuffner J, Nishiwaki K, Kagami S. Manipulability optimization for trajectory generation. In: Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on, 2006, pp. 2017–2022. doi: http://dx.doi.org/10.1109/ROBOT.2006.1642001.

[21] Siciliano B, Sciavicco L, Villani L, Oriolo G. Robotics: modelling planning and control. 1st Edition. Incorporated: Springer Publishing Company; 2008.

[22] Jacquier-Bret J, Gorce P, Rezzoug N. The manipulability: a new index for quantifying movement capacities of upper extremity. Ergonomics 2012;55 (1):69–77. http://dx.doi.org/10.1080/00140139.2011.633176.

[23] Wampler K, Popović J, Popović Z. Animal locomotion controllers from scratch. Computer Graphics Forum 2013;32:153–62. http://dx.doi.org/10.1111/cgf.12035.

[24] Kallman M, Mataric M. Motion planning using dynamic roadmaps. In: Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on, vol. 5, 2004, pp. 4399–4404 vol.5. doi: http://dx.doi.org/10.1109/ROBOT.2004.1302410.

[25] Ye Y, Liu CK. Synthesis of detailed hand manipulations using contact sampling. ACM Transactions on Graphics 2012;31(4):1–10 doi: doi:10.1145/2185520.2185537.

[26] Goldfeder C, Ciocarlie M, Dang H, Allen P. The columbia grasp database. In: Robotics and Automation, 2009. ICRA '09. IEEE International Conference on, 2009, pp. 1710–1716. doi: http://dx.doi.org/10.1109/ROBOT.2009.5152709.

[27] Hauser K, Bretl T, Latombe J-C. Non-gaited humanoid locomotion planning. In: Humanoid Robots, 2005 5th IEEE-RAS International Conference on, 2005, pp. 7–12. doi: http://dx.doi.org/10.1109/ICHR.2005.1573537.

[28] Kalisiak M, van de Panne M. A grasp-based motion planning algorithm for character animation. The Journal of Visualization and Computer Animation 2001;12(3):117–29. http://dx.doi.org/10.1002/vis.250.

[29] Bouyarmane K, Kheddar A. Multi-Contact Stances Planning for Multiple Agents. In: ICRA'11: International Conference on Robotics and Automation, Shanghai International Conference Center, Shanghai, Chine, 2011, pp. 5353–5546.

[30] Mordatch I, Todorov E, Popović Z. Discovery of complex behaviors through contact-invariant optimization. ACM Transactions on Graphics 2012;31(4) 43:1—-43:8. doi: http://dx.doi.org/10.1145/2185520.2185539.

[31] Al Borno M, de Lasa M, Hertzmann A. Trajectory optimization for full-body movements with complex contacts. IEEE transactions on visualization and computer graphics 2012:1–11 doi: 3A697C56-7DB9-4DDB-9C30-6DCA693F48F9.

[32] Hauser K, Bretl T, Harada K, Latombe J-C. Using motion primitives in probabilistic sample-based planning for humanoid robots. In: Akella S, Amato NM, Huang WH, Mishra B, editors. WAFR, Springer Tracts in Advanced Robotics, vol. 47. Springer; 2006. p. 507–22.

[33] Bretl T, Rock S, Latombe J-C, Kennedy B, Aghazarian H. Free-climbing with a multi-use robot. In: Khatib MHA Jr O, editor. ISER Springer Tracts in Advanced Robotics, vol. 21. Springer; 2004. p. 449–58.

[34] Champandard AJ. Procedural Characters and the Coming Animation Technology Revolution, ⟨http://aigamedev.com/open/editorial/animation-revolution/⟩.

[35] Tonneau S, Pettré J, Multon F. Task efficient contact configurations for arbitrary virtual creatures. In: Proceedings of the 2014 Graphics Interface Conference, GI '14, Canadian Information Processing Society, Toronto, Ont., Canada, Canada, 2014, pp. 9–16.

[36] Haering D, Raison M, Begon M. Measurement and description of three-dimensional shoulder range of motion with degrees of freedom interactions, Journal of biomechanical engineering 136(8). Interref in References: http://dx.doi.org/10.1115/1.4027665.

[37] Buss SR. Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and Damped Least Squares methods (2009) 1–19.

[38] Welman C. Inverse kinematics and geometric constraints for articulated figure manipulation. Master's thesis, Simon Frasor University, 1993.

[39] Labriola JE, Lee TQ, Debski RE, McMahon PJ. Stability and instability of the glenohumeral joint: The role of shoulder muscles (Jan. 2005).

[40] Lundgren J. Inpolyhedron - are points inside a volume?. MATLAB Central File Exchange, ⟨http://tinyurl.com/ktcgohk⟩.

[41] Pan J, Zhang L, Lin MC, Manocha D. A hybrid approach for simulating human motion in constrained environments. Computer Animation and Virtual Worlds doi: http://dx.doi.org/10.1002/cav.365.

[42] Liu M, Micaelli A, Evrard P, Escande A. Task-driven posture optimization for virtual characters. In: Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '12, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 2012, pp. 155–164.

[43] Leven P, Hutchinson S. Using manipulability to bias sampling during the construction of probabilistic roadmaps. IEEE Transactions on Robotics and Automation 2003;19(6):1020–6. http://dx.doi.org/10.1109/TRA.2003.819732.