Task Efficient Contact Configurations for Arbitrary Virtual Creatures

Steve Tonneau* IRISA
Julien Pettré† Inria
Franck Multon‡ Univ. Rennes 2 - M2S IRISA

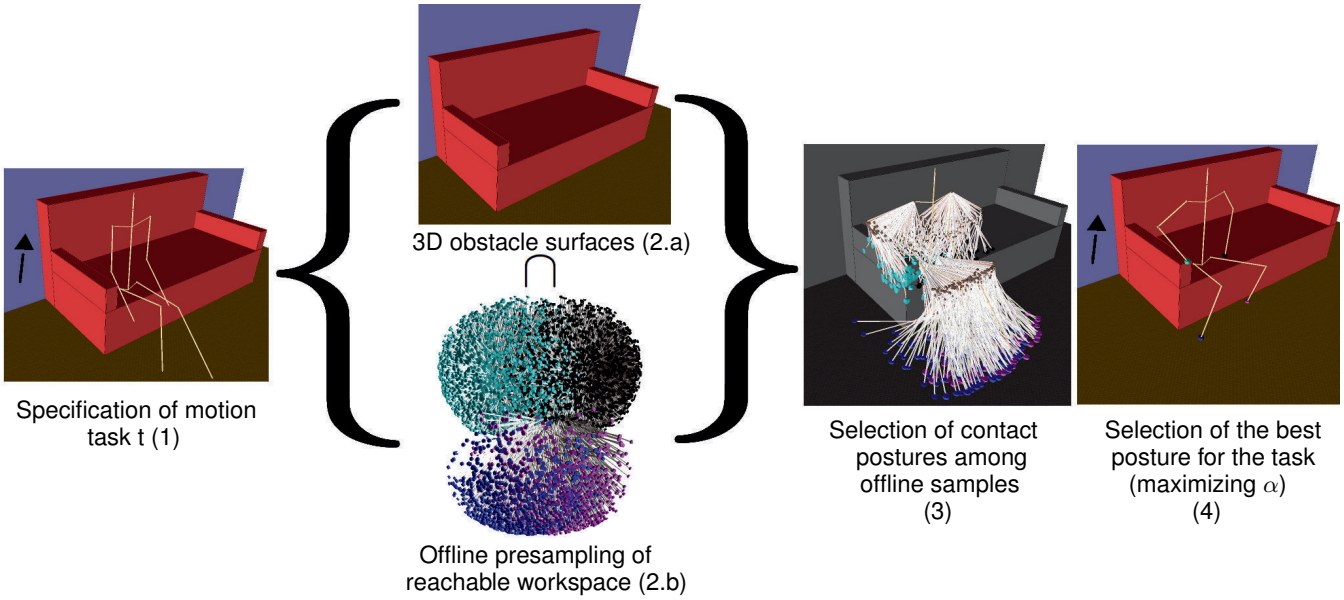

Figure 1: Online step request. Given the task of getting up (1), We transpose the samples from our database into the local environment (2), and select the configurations in contact with the environment (3). Among these candidates, we select the collision-free configurations that maximize the heuristic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

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 reachable workspace of a virtual creature. 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. 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.

Index Terms: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

*stonneau@irisa.fr
†jpettre@inria.fr
‡fmulton@irisa.fr

1 Introduction

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 Figure 9, 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 [11]. Such planners often use randomly generated configurations [29], and select those preserving static stability [10]. 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 [37].

Thus, motion planners and dynamic controllers could benefit from a method to generate appropriate contact configurations. This is our problem statement, formalized in Section 3.

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 [7]. 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 3.4. It is traditionally used to optimize the configuration of a robotic arm, but requires to 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 4). The sampling is performed offline to ensure good performances. Then the samples are filtered online to select configurations in contact with the environment, and free of collisions. This approach is similar to the work introduced in [20], where a precomputed dynamic roadmap is updated as the environment changes.

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.

2 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 2.1); Biomechanical and robotical approaches define relevant contact configurations by quantifying them in terms of force exertion (Section 2.2); Motion planning and optimization methods focus on contact configurations that preserve balance (Section 2.3).

2.1 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 [9] or locomotion on uneven terrain [18, 27]. Similarly foot-step planning techniques proposed hybrid approaches to address this issue [8, 22].

Motion graphs [23, 25] or precomputed search trees [24] 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 [35, 21], 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.

2.2 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 [2]. Yoshikawa presented the manipulability measure for quantifying the ability of robotic mechanisms in positioning and orienting end-effectors [38]. Based on this work, Chiu proposed the force transmission ratio, another index for optimizing a manipulator pose relatively to a specific task [7]. Several manipulability-based methods have since been proposed to either optimize a configuration [31] or a trajectory [13, 33]. Recent works in biomechanics tend to show the relevance of the manipulability measure for human beings [17].

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.

2.3 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 [34].

Contact interactions have been considered for grasping tasks [36, 12], or for motion planning problems. Hauser et al. introduced the *Contact before motion* approach [15], used in several other contributions [19, 10, 3]. 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 [1].

However, to get up from a chair in the environment shown in Figure 5, 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 (Figure 5). 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 [14]. In particular Bretl et al. proposed a heuristic similar to the manipulability measure as a criteria for contact creation [4].

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 [6].

3 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? Figure 2 provides an illustration for such definitions.

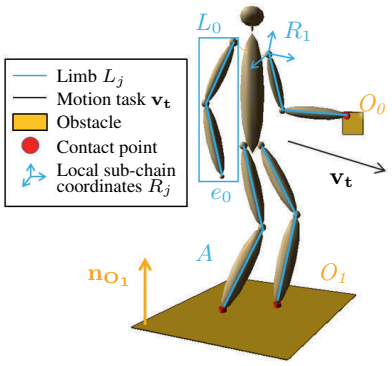

Figure 2: Virtual human composed of 4 limbs. 3 end-effectors are in contact with 2 obstacles.

3.1 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 exactly one end-effector e_j (Figure 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. $p(\Theta_j) = \begin{pmatrix} x_j & y_j & z_j \end{pmatrix}^T$ 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 $\theta^i, i = 1...n$ 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 [5]. 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 $< p(\Theta_j), \Theta_j, Jp(\Theta_j) >$.

3.2 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 Figure 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 Figure 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 Figure 2).

3.3 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 Figure 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.

Task efficient configuration. We want to compute a configuration Θ_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.

3.4 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 [7] showed that this potential force can be quantified by computing, for a given configuration, the force transmission ratio f_T regarding $\mathbf{v_t}$:

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

The force manipulability ellipsoid provides an intuitive representation of the different values taken by the force transmission ratio, as shown in Figure 3 [38]. The value $f_\mathsf{T}(\Theta_j, \mathbf{v_t})$ 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.

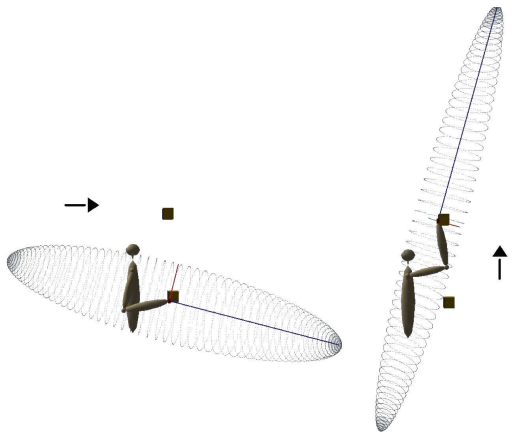

Figure 3: The force manipulability ellipsoid for two different configurations. A longer axis means that a more important force can be exerted in the direction of the axis.

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.

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

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 Figure 5.

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.

With our new heuristic α and those definitions in mind we can describe our contribution.

4 Computation of Task Efficient Contact Configurations

The definitions given in Section 3 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 (Figure 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 (Figure 1).

4.1 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. Figure 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 (Figure 4 – middle left). This is done by independently generating a random angle value for each joint of L_j, comprised between the joint boundary limits.

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 $Jp(\Theta_j)$. $Jp(\Theta_j)$ is needed for the run time computation of the extended force transmission ratio α. Storing it reduces the online computation of α to two simple matrix products.

Computation of $p(\Theta_j)$ (Figure 4 – middle left). The end-effector position $p(\Theta_j)$ is expressed in the limb coordinates R_j.

Insertion of the resulting sample in Q_j (Figure 4 – right). We create the sample denoted by the triplet $< p(\Theta_j), \Theta_j, Jp(\Theta_j) >$, 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.

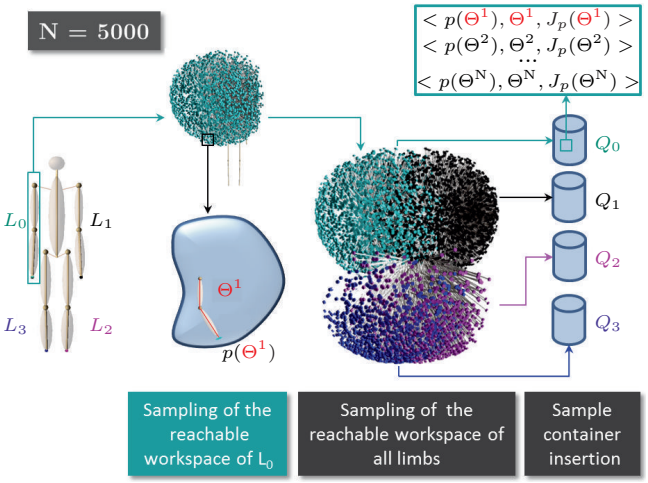

Figure 4: Illustration of the environment-independent offline sampling for $N = 5000$, for the right arm first, then for all the limbs. A sample container is created for each limb. An entry contains a configuration Θ, and its jacobian product J_p. Entries are indexed by the end-effector position $p(\Theta)$.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

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

4.2 Online computation of task efficient contact configurations

We consider the motion task $\mathbf{v_t}$ *–Figure 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 *–Figure 1 (2.a): the sofa, the ground and the wall–*.

Selection of the samples in contact. We request Q_j for all the samples that are in contact with an obstacle of E *–Figure 1 (2.b)–*. The result of the query is a list of limb configurations $Q_j^{contact} \subset Q_j$ *–Figure 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 free of collisions. If all the configurations of $Q_j^{contact}$ are colliding, no configuration is returned *–Figure 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–*.

By decomposing our method into an offline and an online step, we are able to generate task efficient configurations within time limits acceptable for real time applications. Both steps are automatic and do not require manual editing.

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 2.40GHz 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 Figure 6). 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 (Figure 1 and Figure 5 – right). The environment is composed of a sofa, or a chair and a table. The creature is a virtual human (Figure 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.

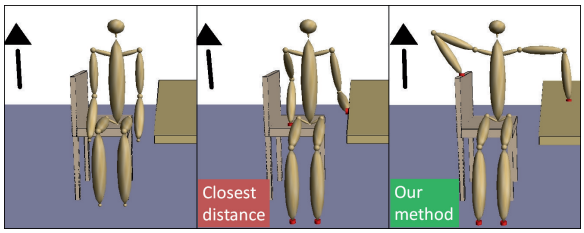

Figure 5: Our method (right) is compared with the closest distance heuristic (middle) in this example of getting up from a chair. In the latter case, the left hand position (on the side of the table) is not suitable to generate a vertical effort.

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 (Figure 6). 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.

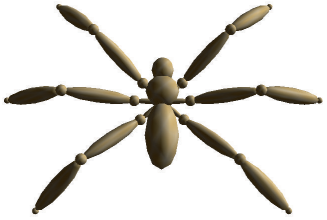

Figure 6: An insect composed of 6 limbs. Each limb is composed of 5 degrees of freedom.

Pushing and pulling objects (Figure 7 and Figure 8 – right). The environment is composed of a cupboard and the ground. In variations of the scenario it is also composed of a rope attached to the cupboard, and a small wall. The creature is a virtual human. 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 $\alpha_{push}(\Theta_j, \mathbf{v_t}) = -\alpha(\Theta_j, \mathbf{v_t})$. These examples show that our method can be applied to pushing and pulling tasks, enhancing the autonomy of motion of our virtual characters.

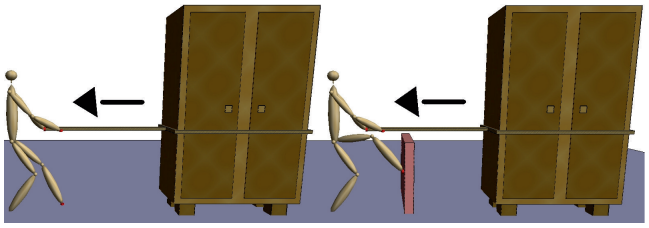

Figure 7: Configurations found for a pulling task. In the right figure, our creature uses the pink wall as a better support for the foot. The asymmetry between the arm configurations is induced by the sampling phase.

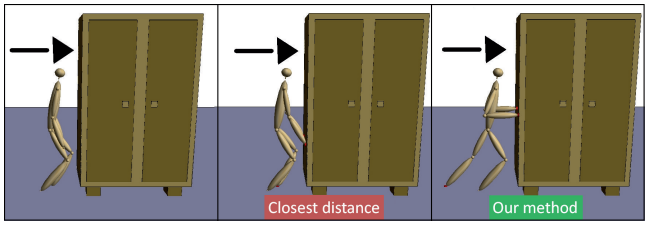

Figure 8: Our method (right) is compared with the closest distance heuristic (middle) in this example of pushing a cupboard. The closest distance heuristic places the hands and left feet at locations close to their original positions (left) while our method places the end-effectors in configurations relevant for the pushing task.

Computation of a sequence of task efficient contact configurations (Figure 9 + 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.

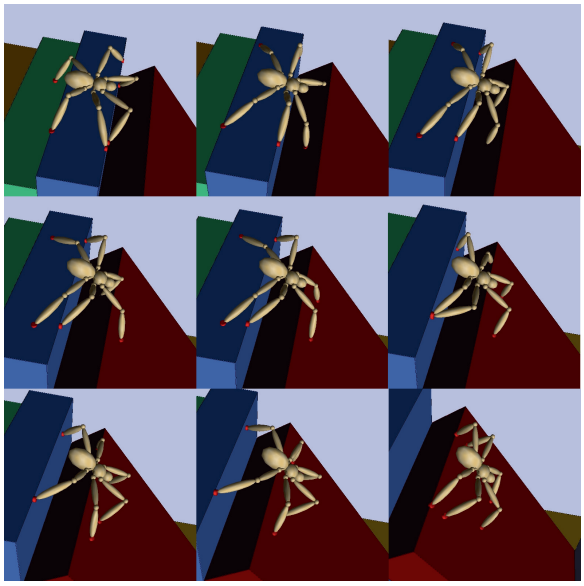

Figure 9: Configuration sequence for an insect with 6 limbs crossing a bookshelf. Task efficient contact configurations are found along the trajectory.

5.3 Comparison of our method with 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 [18, 27]. Hauser et al. manually predefine the set of possible contacts [15]. Bretl et al. use a form of manipulability integrated in a motion planner [4]. 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 Figure 5 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 Figure 8 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 Figure 10 the environment is a climbing wall. The creature is a virtual human (Figure 2). The initial configuration is the reference posture (Figure 10 – right – 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

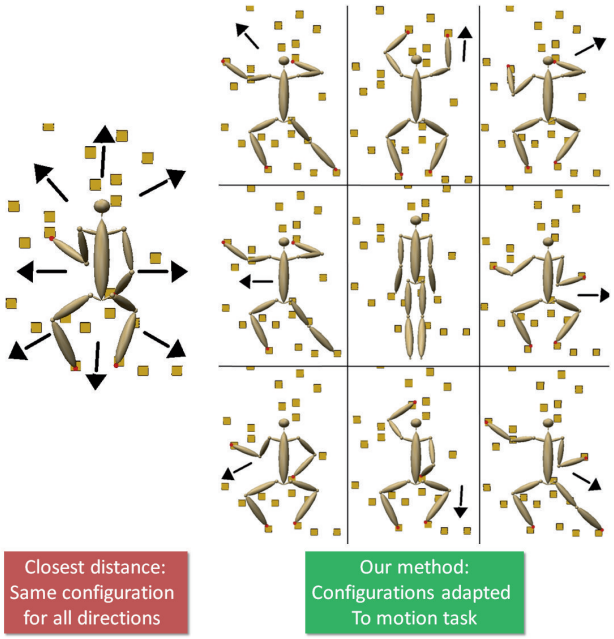

Figure 10: Configurations for a humanoid on a climbing wall. Left: the closest distance heuristic does not consider the motion task, therefore it always computes the same configuration. Right: From the same initial root location (position and orientation), different configurations are computed depending on the task (black arrow).

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 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.

Except for the climbing scenario, the computation time is short for $N \leq 10000$, and the number of average candidates is satisfying in those cases. A higher number of samples is necessary for the

	N = 1 000	N = 10 000	N = 100 000
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 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 = 1 000	N = 10 000	N = 100 000
Climbing	0	1	26
Getting up	41	49	3442
Insect locomotion	20	312	2553
Pushing / pulling	13	142	1387

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

climbing scenario. 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. For the getting up scenario for instance, setting $N = 1000$ is a reasonable choice, where a value of $N = 100000$ seems more appropriate in the climbing scenario. This can be explained by the limited range of motion of the insect limbs, so that a smaller amount of samples is necessary 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).

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. The method generates N samples offline used as inputs for the contact queries. Due to this approach, the method is not probabilistically complete. 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 [18].

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 farther 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 [32].

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 [28]. Doing this would allow us to optimize the whole body according to the computed limb configurations, and produce more natural results.

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 [17]. 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 [26]. 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. 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.

Acknowledgements

The authors wish to thank Fabrice Lamarche for his help on the octree implementation.

References

[1] M. Al Borno, M. de Lasa, and A. Hertzmann. Trajectory Optimization for Full-Body Movements with Complex Contacts. *IEEE transactions on visualization and computer graphics*, pages 1–11, Dec. 2012.

[2] P. Baerlocher and R. Boulic. An inverse kinematics architecture enforcing an arbitrary number of strict priority levels. *The Visual Computer*, 20(6), June 2004.

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

[4] T. Bretl, S. Rock, J.-C. Latombe, B. Kennedy, and H. Aghazarian. Free-climbing with a multi-use robot. In M. H. A. Jr. and O. Khatib, editors, *ISER*, volume 21 of *Springer Tracts in Advanced Robotics*, pages 449–458. Springer, 2004.

[5] S. R. Buss. Introduction to Inverse Kinematics with Jacobian Transpose , Pseudoinverse and Damped Least Squares methods. pages 1–19, 2009.

[6] A. J. Champandard. Procedural Characters and the Coming Animation Technology Revolution. *http://aigamedev.com/open/editorial/animation-revolution/*, 2012.

[7] S. Chiu. Control of redundant manipulators for task compatibility. In *Robotics and Automation. Proceedings. 1987 IEEE International Conference on*, volume 4, pages 1718–1724, 1987.

[8] M. G. Choi, J. Lee, and S. Y. Shin. Planning biped locomotion using motion capture data and probabilistic roadmaps. *ACM Transactions on Graphics*, 22(2):182–203, 2003.

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

[10] A. Escande, A. Kheddar, S. Miossec, and S. Garsault. Planning Support Contact-Points for Acyclic Motions and Experiments on HRP-2. In O. Khatib, V. Kumar, and G. J. Pappas, editors, *ISER*, volume 54 of *Springer Tracts in Advanced Robotics*, pages 293–302. Springer, 2008.

[11] C. Esteves, G. Arechavaleta, J. Pettré, and J.-P. Laumond. Animation planning for virtual characters cooperation. *ACM Transactions on Graphics*, 25(2):319–339, 2006.

[12] C. Goldfeder, M. Ciocarlie, H. Dang, and P. Allen. The columbia grasp database. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 1710–1716, May 2009.

[13] L. Guilamo, J. Kuffner, K. Nishiwaki, and S. Kagami. Manipulability optimization for trajectory generation. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 2017–2022, 2006.

[14] K. Hauser, T. Bretl, K. Harada, and J.-C. Latombe. Using motion primitives in probabilistic sample-based planning for humanoid robots. In S. Akella, N. M. Amato, W. H. Huang, and B. Mishra, editors, *WAFR*, volume 47 of *Springer Tracts in Advanced Robotics*, pages 507–522. Springer, 2006.

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

[16] E. S. Ho and T. Komura. Indexing and retrieving motions of characters in close contact. *IEEE Transactions on Visualization and Computer Graphics*, 15(3):481–492, 2009.

[17] J. Jacquier-Bret, P. Gorce, and N. Rezzoug. The manipulability: a new index for quantifying movement capacities of upper extremity. *Ergonomics*, 55(1):69–77, Jan. 2012.

[18] R. S. Johansen. *Automated Semi-procedural Animation for Character Locomotion*. Aarhus Universitet, Institut for Informations Medievidenskab, 2009.

[19] M. Kalisiak and M. van de Panne. A grasp-based motion planning algorithm for character animation. *The Journal of Visualization and Computer Animation*, 12(3):117–129, July 2001.

[20] M. Kallman and M. Mataric. Motion planning using dynamic roadmaps. In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, volume 5, pages 4399–4404 Vol.5, April 2004.

[21] M. Kallmann, A. Aubel, T. Abaci, and D. Thalmann. Planning Collision-Free Reaching Motions for Interactive Object Manipulation and Grasping. *Computer graphics Forum (Proceedings of Eurographics'03)*, 22(3):313–322, 2003.

[22] O. Kanoun, J.-P. Laumond, and E. Yoshida. Planning foot placements for a humanoid robot: A problem of inverse kinematics. *Int. J. Rob. Res.*, 30(4):476–485, Apr. 2011.

[23] L. Kovar, M. Gleicher, and F. Pighin. Motion graphs. In *ACM Transactions on Graphics*, volume 21, pages 473–482, New York, NY, USA, 2002. ACM.

[24] M. Lau and J. J. Kuffner. Precomputed search trees: planning for interactive goal-driven animation. In *Symposium on Computer Animation*, pages 299–308, 2006.

[25] J. Lee and K. H. Lee. Precomputing avatar behavior from human motion data. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '04, pages 79–87, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association.

[26] P. Leven and S. Hutchinson. Using manipulability to bias sampling during the construction of probabilistic roadmaps. *Robotics and Automation, IEEE Transactions on*, 19(6):1020–1026, Dec 2003.

[27] S. Levine and J. Popovic. Physically Plausible Simulation for Character Animation. In *Symposium on Computer Animation*, pages 221–230, 2012.

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

[29] T. Lozano-perez. Spatial Planning : A Configuration Space Approach. c(2), 1983.

[30] I. Mordatch, E. Todorov, and Z. Popović. Discovery of complex behaviors through contact-invariant optimization. *ACM Transactions on Graphics*, 31(4):43:1—-43:8, 2012.

[31] N. Naksuk and C. S. G. Lee. Zero moment point manipulability ellipsoid. In *ICRA 2006 Proceedings*, pages 1970–1975, 2006.

[32] J. Pan, L. Zhang, M. C. Lin, and D. Manocha. A hybrid approach for simulating human motion in constrained environments. *Computer Animation and Virtual Worlds*, 2010.

[33] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. *Robotics: Modelling, Planning and Control*. Springer Publishing Company, Incorporated, 1st edition, 2008.

[34] K. Wampler, J. Popović, and Z. Popović. Animal Locomotion Controllers From Scratch. *Computer Graphics Forum*, 32:153–162, May 2013.

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

[36] Y. Ye and C. K. Liu. Synthesis of detailed hand manipulations using contact sampling. *ACM Transactions on Graphics*, 31(4):1–10, 2012.

[37] K. Yin, K. Loken, and M. van de Panne. Simbicon: Simple biped locomotion control. *ACM Transactions on Graphics*, 26(3):Article 105, 2007.

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