Ballistic motion planning for jumping superheroes

Mylène Campana, Pierre Fernbach, Steve Tonneau, Michel Taïx, Jean-Paul Laumond

Ballistic motion planning for jumping superheroes

Mylène Campana*, Pierre Fernbach*, Steve Tonneau*,
Michel Taïx*, Jean-Paul Laumond*

LAAS-CNRS, Université de Toulouse, CNRS, UPS, Toulouse, France

Motion in Games Conference 2016†

Abstract. We present a framework capable of automatically planning and animating dynamic multi-contact jumping trajectories for arbitrary legged characters and environments. Our contribution is the introduction of an approximate yet efficient multi-contact impulse formulation, used at the motion planning phase. We combine this criterion with a heuristic for estimating the contact locations without explicitly computing them, which breaks the combinatorial aspect of contact generation. This low dimensional formulation is used to efficiently extend an existing ballistic motion planner to legged characters. We then propose a procedural method to animate the planned trajectory. Our approach thus results from a trade-off between accuracy of the law of physics and computational efficiency. We empirically justify this approach by demonstrating a large variety of behaviors for four legged characters in five scenarios.

Keywords: ballistic trajectory; jump; motion planning; animation

1 Introduction

Synthesizing high quality motions for legged characters in arbitrary environments is challenging: the dimension of the problem is high, equal to the number of degrees of freedom, the environment is big and complex, and the motion is additionally constrained by the dynamics of the world. For this reason, motion synthesis is typically addressed with a decoupled approach. First, in the path planning phase a collision-free path is found for the character using a PRM based approach [1]. Then, in the animation phase a motion is computed and played along the path. Using this decoupled approach, we consider the issue of synthesizing highly dynamic jumping motions for legged characters.

For concision our scope is restricted to the generation of sequences of jumping motions, disregarding alternations with classical walking / running trajectories at the moment. Thus, given start and goal configurations for an arbitrary character in an arbitrary environment, our framework outputs a motion described as a sequence of parabolic jumps [2], respecting a set of kinematic and dynamic constraints for the character.

Our key idea is the introduction of a simplified multi-contact model within a sampling based planner. We relax the dynamics of the problem with two hypotheses, which assimilate our character with a superhero:

- First, we consider an impulse formulation. Our character is thus assumed to be able to exert a large force instantaneously, as proposed in [2].
- Then, when verifying the dynamic constraints on the character, we verify the Newton equation of the motion, but assume that the Euler equation is always satisfied. This means that we assume that the character can generate an arbitrary angular momentum.

*e-mails: {mcampana, pfernbac, stonneau, taix, jpl}@laas.fr
†DOI: http://dx.doi.org/10.1145/2994258.2994279

These two hypotheses lead to an efficient, low dimensional formulation of the motion planning problem, solved with a sequence of simple geometric tests, while partially capturing the dynamics model of the character to compute plausible trajectories. To handle the combinatorial aspect of the contact generation problem, we decouple the trajectory planning phase from the contact generation phase, thanks to a heuristic based on the reachable workspace of the character. Finally, to propose a complete framework, we implement a procedural method to automatically animate the computed trajectory. This is achieved by combining a contact generator proposed by [3] with a dedicated interpolation scheme that we introduce.

Our main contribution is thus a computationally efficient ballistic multi-contact motion planner. It extends the PRM planner proposed by the authors of [2], limited to planning for a single mass point, to arbitrary legged characters generating non-coplanar contact configurations (Sections 3 and 4). Our second contribution is an automatic interpolation method for animating multi-contact jumping trajectories (Section 5).

2 Related Work

We propose a purely procedural motion synthesis methods, able to generate dynamic motions through multi-contact interactions, with a focus on the motion planning phase. Our work is thus related to physically based motion synthesis methods, as well as motion and contact planning techniques. It is also to be compared with respect to data-driven animation techniques (such as motion capture). This related work provides an overview of these approaches, before focusing to specific contributions to the jumping motion synthesis problem.

A standard motion synthesis technique is to use pre-existing animations, produced either by 3D artists or through motion capture. These approaches are favored because of the high quality of the resulting animations [4]. When the environment differs from the original one, local editing methods can be used to adapt the motion [5, 6, 7, 8], but they produce unnatural results when the deformation becomes too important.

Rather than using reference animations, physics-based methods synthesize motions with algorithms based on a simplified model of the law of physics [9]. Walking pattern generators combine finite state machines with proportional derivative control to generate periodic locomotion [10]. The number of states increases with the complexity of the environment and the desired motions, such that reinforcement based learning techniques are required to overcome the limitations [11, 12]. However, they are based on simplifying assumptions regarding the location and periodicity of the contacts, which do not hold in arbitrary environments.

Sampling based planning algorithms such as PRM [1] and RRT [13] are the standard methods used to find a collision-free path in a complex 3D environment. Our approach follows [14], with an interactive method that breaks the combinatorial by decoupling the planning of a feasible path and the contact generation.

Overall, contributions that have studied jumping motions do not focus explicitly on path planning, rather on the physically accurate adaptation or synthesis of the jump animation, which is complementary with our approach.

As such, most robotics contributions to jumping scenarios assume coplanar contacts on flat surfaces, and do not consider collision avoidance in general [15, 16]. Similarly computer graphics contributions assume the ballistic jumping motion is already precomputed [17], and focus on the preparation phase [18, 19], or the reception phase [20].

To plan highly dynamic motions, recent contributions have proposed hybrid approaches, using both data-driven and physics based methods to generate motions [21, 22]. In particular, [23] propose to deform motion capture trajectories using physics based heuristics. Interestingly, while the authors make similar assumptions to us (they consider characters with surhuman capabilities), their approach is opposite to ours: they constrain a motion adaptation to respect the Euler equation of motion, given customizable bounds on the angular momentum of the character. Because the contact locations are predefined relatively to the center of mass by the reference motion capture animation, the linear part is necessarily validated. However, this limitation once again prevents generalization to arbitrary environments, where the contacts must be changed to obtain a valid motion. We, on the other hand, focus on the Newton equation that regularizes the linear momentum, and do not constrain the angular

momentum, making our method able to handle arbitrary contact locations.

3 Relaxed contact model for impulse based motion planning

Our contact model is based on Coulomb's non-slipping constraint, which we recall briefly in this section. We generalize the constraint to handle an arbitrary number of contacts, by expressing it at the center of mass (COM) of the robot. From this formulation, assuming an impulse formulation of the model, we propose a conservative condition for non-slipping, that is sufficient, but not necessary. This condition, reduced to a simple geometric test, is then used in our motion planner, presented in Section 4.

Definitions. We consider a character of mass $m \in \mathbb{R}$, of COM position $\mathbf{c} \in \mathbb{R}^3$, assimilated to the geometric root. $\dot{\mathbf{c}}$ is the velocity of $\mathbf{c}$.

For the i-th contact point $\mathbf{p}_i$, $1 \leq i \leq h$, $\mathcal{K}_i$ is the associated convex friction cone, considering a friction coefficient μ_i, and a surface normal $\mathbf{n}_i$. $\mathbf{f}_i$ is the contact force applied at $\mathbf{p}_i$. Finally, $\mathbf{g} = [0, 0, -9.81]^T$ is the gravity acceleration.

Non-slipping constraint for an arbitrary number of contacts. The non-slipping condition is given by Coulomb's law: the contact will not slip if the contact reaction force $\mathbf{f}_i$ lies strictly within the cone $\mathcal{K}_i$.

Now, if we consider an arbitrary number h of contacts, each reaction force $\mathbf{f}_i$ must lie in its associated cone $\mathcal{K}_i$. The resulting force $\mathbf{f_c}$ applied at the center of mass $\mathbf{c}$ of the character, is defined as the sum of all the forces $\mathbf{f}_i$. It follows that the set of admissible resulting forces $\mathcal{K}$ such that the non-slipping condition is respected is defined as the Minkowski sum of each individual friction cone:

$$\mathcal{K} = \{\mathbf{f}_1 + \cdots + \mathbf{f}_h | \mathbf{f}_i \in \mathcal{K}_i\} \tag{1}$$

As a Minkowski sum of convex cones, $\mathcal{K}$ is itself a convex cone [24].

However, the analytical form of $\mathcal{K}$ is unknown. Therefore in this work, we use a conservative approximation $\mathcal{K}_\mathbf{c}$.

We define $\mathbf{n_c} = \sum_{i=1}^{h} \mathbf{n_i}$ and $\mu_\mathbf{c} = \min_{i=1\ldots h} \mu_i$.

The cone $\mathcal{K}_\mathbf{c}$, originating at $\mathbf{c}$, of normal $\mathbf{n_c}$, and friction $\mu_\mathbf{c}$, is included in $\mathcal{K}$ by construction, and is thus a conservative approximation of $\mathcal{K}$. Force closure contact configurations, where the resulting normal $\mathbf{n_c}$ is null, are considered invalid in this formulation. Although this formulation is intuitively really conservative, it has an analytical form that makes it extremely efficient to compute. We justify the interest of this formulation with the variety of the solutions found by our planner.

Impulse model of the non-sliding constraints. We extend the model proposed in [2] to multi-contact motions, by applying it to the centroidal cone $\mathcal{K}_\mathbf{c}$. Assuming that the motion is determined as a sequence of jumps, we want to verify the non-slipping conditions between the landing and takeoff phases of two consecutive jumps. With an impulse model, we make the hypothesis that the character can only apply large contact forces instantaneously, and assume that the transition between the landing and takeoff phases is instantaneous.

Under the conservative non-slipping condition, Newton's equation is written:

$$m \frac{d\dot{\mathbf{c}}}{dt} - m\mathbf{g} = \mathbf{f_c} \mid \mathbf{f_c} \in \mathcal{K}_\mathbf{c}$$

Under the impulse hypothesis dt is instantaneous, such that the action of gravity is not measurable:

$$m\ d\dot{\mathbf{c}} \approx \mathbf{f_c} dt \mid \mathbf{f_c} \in \mathcal{K}_\mathbf{c}$$

Thus, $d\dot{\mathbf{c}}$ and $\mathbf{f_c}$ are colinear, and $d\dot{\mathbf{c}} \in \mathcal{K}_\mathbf{c}$.

Furthermore, the instantaneous velocity shift $d\dot{\mathbf{c}}$ is constrained by the relationship

$$d\dot{\mathbf{c}} = \dot{\mathbf{c}}_t - \dot{\mathbf{c}}_l \tag{2}$$

where $\dot{\mathbf{c}}_t$ and $\dot{\mathbf{c}}_l$ are respectively the takeoff and landing velocities.

We now make the observation that, by definition of a convex cone:

$$-\dot{\mathbf{c}}_l \in \mathcal{K}_{\mathbf{c}} \text{ and } \dot{\mathbf{c}}_t \in \mathcal{K}_{\mathbf{c}} \implies d\dot{\mathbf{c}} \in \mathcal{K}_{\mathbf{c}} \tag{3}$$

(3) provides a sufficient condition for satisfying the non-slipping condition. This conservative condition reduces the dimensionality of the motion planning problem, because it removes the relationship between the entering velocity and the exiting velocity of a node, required by classical kinodynamic approaches [25]. With a kinodynamic planner, to verify whether a trajectory can be connected with another one, it is required to extend the state space with the velocities. In our case, this is required to verify whether there exists a $\dot{\mathbf{c}} \in \mathcal{K}_{\mathbf{c}}$ that satisfies Equation 2 each time we want to connect a new trajectory. Including the velocities doubles the dimensionality of the problem. The conservative condition removes this need: any takeoff velocity $\dot{\mathbf{c}}_t$ that belongs to the cone is automatically achievable from the previous one, independently of $\dot{\mathbf{c}}_l$.

4 Path planner

We integrate the multi-contact non-slipping condition within a PRM-based motion planner to consider the dynamics constraints that apply to the character at the planning phase. To reduce the dimensionality of the problem and break the combinatorial, the planner does not consider the complete character model at this phase, nor the explicit computation of contact configurations.

We first present and justify the reduced character model used by the planner. We then introduce a heuristic to estimate the contact location of the configuration without computing them explicitly. Then, we give the details of the planning algorithm.

A reduced character model for contact location estimation. We consider a legged **character**, described by a kinematic chain R, composed of a root R^0, and l limbs $R^k, 1 \leq k \leq l$.

The root has a minimum of $r \geq 6$ degrees of freedom (dofs), which describe its position and orientation in the world frame. The additional dofs describe the articulations of the back of the character. R is fully described by a configuration $\mathbf{q} \in \mathbb{R}^{r+n}$. We define $\mathbf{q}^0 \in \mathbb{R}^r$ as the configuration of the root of the character.

At the planning phase, to verify the non-slipping condition, we need an estimation of the contact locations. To avoid dealing with the combinatorial and computational complexity of contact generation, we introduce a contact estimation heuristic, based on a dual, low-dimensional representation of the character, introduced by [14]. We recall it here for completeness.

Rather than considering the complete body configuration, the planner only considers the root configuration $\mathbf{q}^0$. To perform collision detection, the complete body is approximated with a bounding shape $\mathbf{W}^0$. Additionally, for each limb k, we attach to the root a shape $\mathbf{W}^k$, computed as the reachable workspace of the limb. For details on the computation of the $\mathbf{W}$ shapes, we refer the reader to [14].

Our contribution is to use the reachable workspace of each limb for contact estimation. Given a configuration $\mathbf{q}^0$, we assume that if $\mathbf{W}^k$ is in collision with the environment, it is possible for the character to create a contact (Figure 1) between the limb R^k and the environment. The contact location is estimated to be at the center of the intersection between the largest colliding contact surface and $\mathbf{W}^k$. This heuristic provides an efficient method to approximate the contact location, and allows the verification of the non-slipping condition without considering the expensive complete model.

Motion planning algorithm for the reduced model. Our algorithm is an extension of the ballistic PRM planner [2], adapted to generate multi-contact jumping motions. It is based on an oriented version of the PRM planner [1]. The PRM algorithm processes offline, to generate a graph of connected configurations, which captures the topology of the environment. The resulting graph can be efficiently queried online to compute a path between two specified configurations. The nodes of the graph are configurations. They are connected if there exists a collision-free path between them. The path is computed using a steering method, which takes into account the kinematic (and eventually, the dynamic) constraints of the model to generate a path.

We modify the original algorithm in three ways:

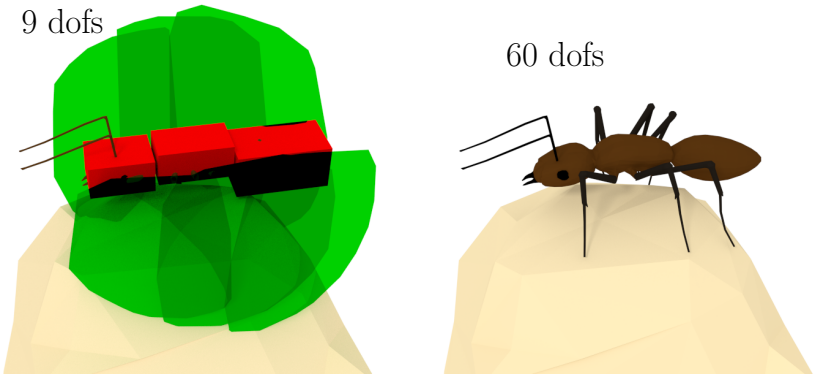

Figure 1: Illustration of a necessary condition for contact creation. If the trunk bounding box is collision-free (Left–red), and the reachable workspaces of the limbs are in collision with the environment (Left–green), we assume that a contact configuration can be created between the effectors and the environment (Right).

- First, as proposed by [14], the configuration sampling is biased towards configurations that allow to generate contacts. Without loss of completeness, we only consider configurations for which at least one contact creation is possible, that is that if a shape $\mathbf{W}^k$ is in collision for this configuration (Figure 1);
- Then, to generate a trajectory between two configurations, we use the parabolic steering method introduced by [2], used to represent jumps;
- Our contribution lies in the validation of the generated trajectory. A trajectory between two configurations is only validated if the multi-contact non-slipping condition is validated.

Algorithm 1 gives the pseudo-code of the motion planning algorithm. It generates a graph, called a roadmap, which can be requested online to compute a sequence of jumps between a start and a goal configuration.

VALIDTRUNKRANDOMSAMPLE returns a root configuration $\mathbf{q}^0$. $\mathbf{q}^0$ is such that $\mathbf{W}^0$ is collision-free, and $\exists k, \mathbf{W}^k$ is in collision.

FINDCONTACTS estimates the contact positions on the surfaces resulting of the intersections between the $\mathbf{W}^k$ shapes and the environment. From the estimated contact locations, the centroidal cone $\mathcal{K}_\mathbf{c}$ is computed with the method COMPUTECONE.

VALIDSTEER is an extension from the parabola-steering-method of [2]. To generate a trajectory between two configurations $\mathbf{q}_a^0$ and $\mathbf{q}_b^0$, the method determines whether there exists a parabola that verifies the non-sliding condition. This means that the takeoff velocity belongs to the centroidal cone $\mathcal{K}_\mathbf{c}(a)$, while the landing velocity belongs to $\mathcal{K}_\mathbf{c}(b)$. The takeoff and landing velocities are also limited by user-defined bounds $\dot{\mathbf{c}}_t^{max}$ and $\dot{\mathbf{c}}_l^{max}$. These bounds are manually chosen according to the maximal allowed jump range X^{max} of the character (e.g. on a flat ground, $X^{max} = \dot{\mathbf{c}}_t^{max\,2}/9.81$). The trajectory is validated if the resulting parabola is collision-free, that is if $\mathbf{W}^0$ is collision-free along the path. If no valid parabola can be found, VALIDSTEER returns an empty trajectory $\mathbf{T}$.

5 Online path request and motion synthesis

Once a roadmap has been generated, requesting a trajectory between two configurations consists in adding them to the roadmap using Algorithm 1. If both configurations have been successfully added, the shortest trajectory connecting them is obtained by computing the shortest traversal of the graph.

A remaining step is to extend the trajectory $\mathbf{q}^0(t)$ computed for the root of the character into a full body animation. In this phase, despite the impulse model formulation, to obtain a plausible animation, we consider that contact duration is not instantaneous. This requires identifying the transition times between contact and flight for each effector.

The whole-body animation technique proceeds as follows for each jump:

Algorithm 1 Probabilistic roadmap planner for ballistic motion planning of a dual-shape system.

Input: *environment*, $\dot{\mathbf{c}}_l^{max}$, $\dot{\mathbf{c}}_t^{max}$
Output: Graph of valid root configurations connected with parabolas

```
finished ← False
while not(finished) do
    q⁰_rand ← validTrunkRandomSample()
    findContacts(q⁰_rand)
    K_c ← computeCone(q⁰_rand)
    addToRoadmap(q⁰_rand, K_c)
    for q⁰ ∈ Roadmap do
        T ←validSteer(q⁰, q⁰_rand, ċ_l^max, ċ_t^max)
        addToRoadmap(T)
        T ←validSteer(q⁰_rand, q⁰, ċ_l^max, ċ_t^max)
        addToRoadmap(T)
    end for
    finished ← TerminationCondition()
return Roadmap
```

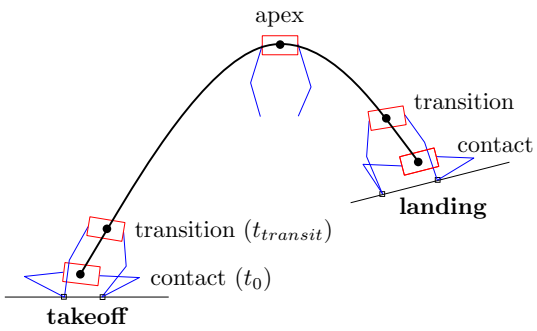

Figure 2: Key configurations computed along a parabola trajectory.

- First, several key configurations are automatically computed at specific times of the trajectory: two contact configurations at the take-off and landing phases; one configuration at each moment where an effector contact is broken (respectively created); one configuration at the apex of the trajectory (Figure 2).
- Then, a linear interpolation is performed for each dof of the character between each key configurations. It is designed in such a way that collisions are avoided and contact location constraints are maintained.

We first explain how the complete contact configurations are computed, and how the moment where a contact is broken / created is identified. Then, we present the interpolation method to generate the continuous complete animations.

Computation of whole-body contact configurations, and identification of takeoff and landing phases. We first address the identification and animation of the takeoff and landing phases, where contacts occur between the character and the environment. To keep the explanation simple, in this section we consider an input trajectory $\mathbf{q}^0(t)$ which consists in a single jump, and assume that contacts are created and broken simultaneously, although the method handles the general cases.

To identify the duration of the landing phase, we first generate a collision-free, whole-body contact configuration that verifies the non-sliding condition at the impact frame. To generate such a configuration, we use a dedicated contact generator proposed in [3]. To increase the plausibility of the animation, contact generation is biased towards configurations as close as possible to a manually defined reference configuration (Figure 3). Contact locations may differ from the planned ones as long as the new centroidal cone $\mathcal{K}_\mathbf{c}$ satisfies the criterion (3).

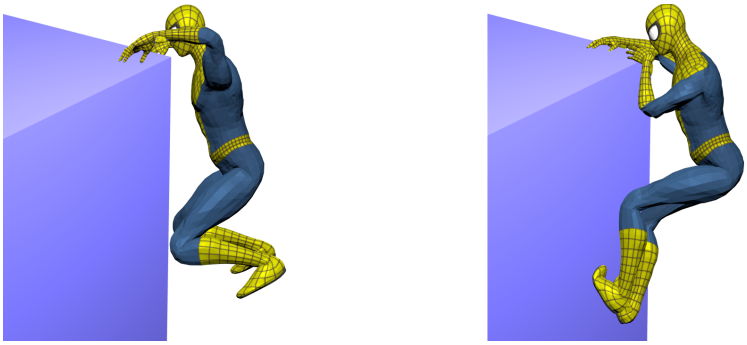

Figure 3: Illustration of the generation of a contact configuration. (Left) The root of the reference configuration is placed close to an obstacle so that the obstacle is in the accessible space of the limbs. (Right) Result of the configuration projection on the obstacle to create contacts on hands and feet.

We then identify the transition time between the take-off and flight phases using an iterative approach. We go forward in time from $t_0 = 0$. At each time step t_i, we update the root configuration $\mathbf{q}^0(t_i)$, and solve an inverse kinematics problem to maintain the contacts active with the obstacle surface. The last time $t_{transit}$ before the inverse kinematics fails is the transition time with the flight phase. The landing phase is handled similarly, with the exception that we go backwards in time from the impact time.

An additional key configuration is generated at the apex of the parabola. It is sampled to be collision-free, with a bias to lie in the neighborhood of a manually defined reference configuration.

Whole body animation of the jump trajectory. The final animation is obtained by linear interpolation between the computed key configurations (Figure 2). While the root motion is guaranteed to be collision-free, the linear interpolation can result in one limb being in collision with the environment. In this case a local re-sampling phase is performed to find a collision-free trajectory. During this re-sampling only the dofs of the limb in collision are modified. In particular the root motion remains unchanged.

6 Simulations

Our framework has been implemented using the open source motion planning software Humanoid Path Planner (HPP) [26]. The method source code is available online. Final renderings have been generating with Blender 2.7 using the automatic blender export tool of HPP.

We demonstrate the genericity of our approach with four characters in five environments. The characters have different morphologies with from two to six limbs. They also have different jump abilities, expressed as the friction coefficients of the environment, as well as velocity bounds. The Figure 4 illustrates a few simulation results. All the key qualitative results of the method are detailed in the companion video : they include removals of unnatural jumps by the non-sliding constraints, jump transitions with arbitrary numbers of contacts and collision avoidance.

Time performances. Table 1 provides a quantitative analysis of our approach performances in four scenarios. We separate the offline step, including the roadmap construction, from the online step, including the trajectory query and the jump synthesis. Since a full-path motion synthesis is dependent on the number of waypoints found by the planner, we give the average computation times to synthesize one jump and one contact configuration. All benchmarks were run on a PC with 64 GB of main memory and using one core of an Intel Xeon E5-1630 processor running at 3.7 GHz.

From the results we obtain, we observe that the complete computation time for one jump is inferior to the actual animation time. This is also true for our worst case scenario, the houses rooftops with Jumper-man. In this case, the large numbers of triangles that describe both the scene and the character explain the drop in the performances, due to the collision tests. As classically done in video games, using simplified meshes at the planning phase would probably allow us to obtain better performances

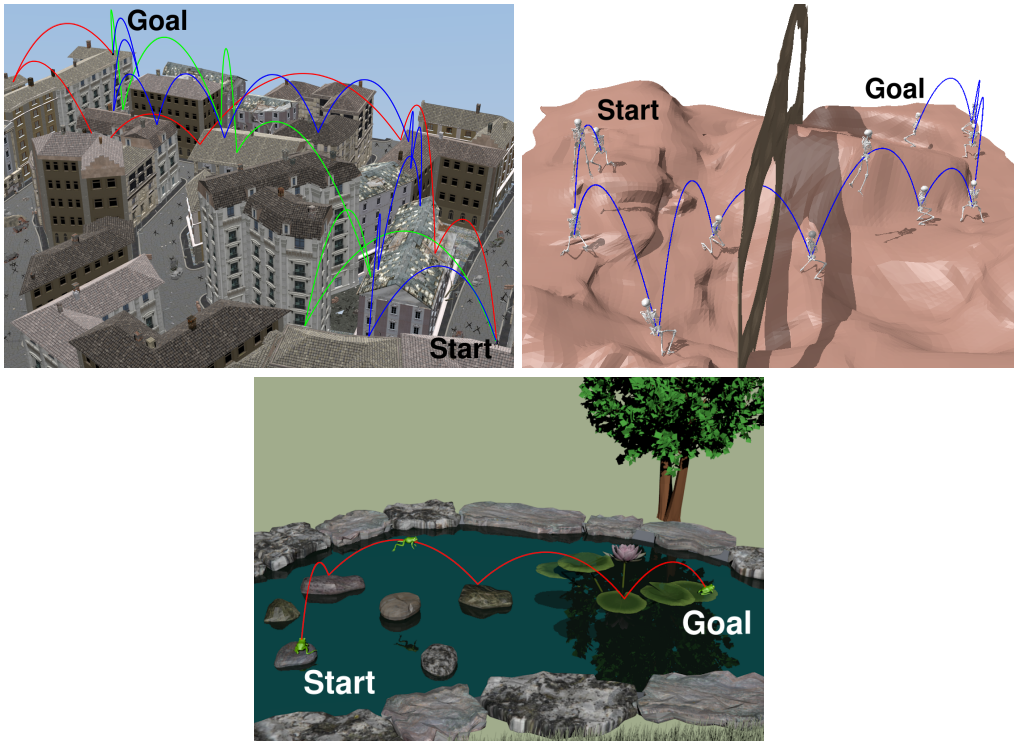

Figure 4: Sequences of jumps computed by our planner in three environments.

Scenarios	dofs	Offline		Online				
		Roadmap CT (s)	N. of nodes	Full trajectory CT (s)	One-jump CT (s) Average	One-jump CT (s) Min — Max	N. of jumps	Contact generation (s)
Skeleton in desert	42	0.51	22.4	2.8718	0.3231	0.2167 - 0.9147	9.2	0.0253
Frog in pond	45	0.76	46.6	0.4742	0.1185	0.1077 - 13.25	5.3	0.0326
Jumper-man on houses*	38	44.93	142.4	7.2497	1.3894	0.3307 - 3.9250	6.1	0.0765
Skeleton on houses*	45	107.13	1256.3	7.2737	0.5057	0.4381 - 0.6923	14.4	0.0264

Table 1: Computation time (CT) averages of the method stages for 50 runs of each of the scenarios. Offline columns concern the planning procedure while online columns deal with the path query in the roadmap and the motion synthesis of the whole trajectory. The number of nodes reflects the roadmap memory occupation. Number of triangles: house 449267, desert 47733, pond 6994, Jumper-man 10052. *Skeleton parameters: $\mu_{\mathbf{c}} = 0.6$, $\dot{\mathbf{c}}_t^{max} = 18$ m/s, $\dot{\mathbf{c}}_l^{max} = 25$ m/s. *Jumper-man parameters: $\mu_{\mathbf{c}} = 1.2$, $\dot{\mathbf{c}}_t^{max} = \dot{\mathbf{c}}_l^{max} = 25$ m/s.

without impacting the quality of the results. However we chose to preserve the complexity of the scene in this work, to demonstrate that the method scales well with the number of triangles.

More importantly, the critical observation is that both the trajectory query and the contact generation are real time in every scenario. We recall that those are the main contributions of the paper, and the performances obtained allow us to consider interactive applications with more advanced animation methods.

7 Conclusion

This work introduces an efficient motion planner, able to compute and animate complex trajectories for jumping legged characters, by extending a ballistic motion planner proposed by [2]. In particular, contrary to previous works based on finite state machines or data-driven animation, the planner is able to automatically compute non-coplanar multi-contact configurations, without making hypotheses about the number of contacts required and their locations. As such, it is able to address arbitrary environments.

Despite the integration of a dynamic impulse contact model, our planner is computationally efficient,

thanks to the introduction of a low dimensional conservative criterion for verifying the non-slipping condition without explicitly computing the contact locations. This reduction of the problem's dimension only approximates the kinematic constraints of the character, and in rare cases, the whole body contact generation at the animation phase will fail.

A current limitation of the model is that it does not consider angular momentum in the planning phase. Furthermore, our approximation of the non-slipping condition is highly conservative. While the analytical computation for the Minkowski sum of friction cones is not tractable, we could consider another conservative approximation of the centroidal cone: the Minkowski sum resulting of the linearized shape of the cones [27, 28] is algorithmically computable. We wish to compare the quality and computation times of the two approximations for future work.

Regarding the quality of the obtained animations, we believe that the non-slipping criterion increases the plausibility of the obtained motion, despite a current limitation of the model, which does not consider angular momentum in the planning phase. We leave the validation of this hypothesis through a perception user study for future work. To integrate angular momentum constraints within the planning phase, we consider extending the centroidal cone to a 6 dimensional wrench-cone, following the formulation of [29].

Lastly, we aim at extending our planner to integrate other phases than jumping, to be able to alternate running, rolling or crawling sequences, for a larger spectrum of application.

Acknowledgements

This work is supported by the European Research Council through the Actanthrope project (ERC Grant Agreement 340050) and the French National Research Agency Entracte project (ANR Grant Agreement 13-CORD-002-01). We would like to thank the anonymous reviewers for their time and comments.

References

[1] L.E. Kavraki, P. Svestka, J.C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Rob. Autom.*, 12(4):566–580, 1996.

[2] M. Campana and J.-P. Laumond. Ballistic motion planning. In *IEEE International Conference on Intelligent Robots and Systems (IROS) (to appear)*, 2016.

[3] Steve Tonneau, Julien Pettré, and Franck Multon. Using task efficient contact configurations to animate creatures in arbitrary environments. *Computers & Graphics*, 45(0), 2014.

[4] Lucas Kovar, Michael Gleicher, and Frédéric Pighin. Motion graphs. In *ACM Trans. on Graphics (TOG)*, volume 21, New York, NY, USA, 2002. ACM.

[5] A. Witkin and Z. Popović. Motion warping. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, pages 105–108, New York, NY, 1995. ACM.

[6] Julien Pettré, Jean-Paul Laumond, and Thierry Siméon. A 2-stages locomotion planner for digital actors. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '03. Eurographics Association, 2003.

[7] Claudia Esteves, Gustavo Arechavaleta, Julien Pettré, and Jean-Paul Laumond. Animation planning for virtual characters cooperation. *ACM Trans. on Graphics (TOG)*, 25(2):319–339, 2006.

[8] D. Holden, J. Saito, and T. Komura. A deep learning framework for character motion synthesis and editing. *ACM Trans. on Graphics (Proc. SIGGRAPH) 35, 4*, 2016.

[9] Igor Mordatch, Emanuel Todorov, and Zoran Popović. Discovery of complex behaviors through contact-invariant optimization. *ACM Trans. on Graphics (TOG)*, 31(4), 2012.

[10] K. Yin, K. Loken, and M. van de Panne. Simbicon: Simple biped locomotion control. *ACM Transactions on Graphics (Proc. ACM SIGGRAPH)*, 2007.

[11] Xue Bin Peng, Glen Berseth, and Michiel van de Panne. Dynamic terrain traversal skills using reinforcement learning. *ACM Transactions on Graphics (to appear)*, 2015.

[12] Libin Liu, Michiel Van De Panne, and KangKang Yin. Guided learning of control graphs for physics-based characters. *ACM Transactions on Graphics (TOG)*, 35(3):29, 2016.

[13] S. M. LaValle and J. J. Kuffner, Jr. *Algorithmic and Computational Robotics: New Directions*, chapter Rapidly-Exploring Random Trees: Progress and Prospects, pages 293–308. Wellesley (MA), 2001.

[14] Steve Tonneau, Nicolas Mansard, Chonhyon Park, Dinesh Manocha, Franck Multon, and Julien Pettré. A reachability-based planner for sequences of acyclic contacts in cluttered environments. In *Int. Symp. Robotics Research (ISRR), (Sestri Levante, Italy), September 2015*, 2015.

[15] E. Papadopoulos, I. Fragkos, and I. Tortopidis. On robot gymnastics planning with non-zero angular momentum. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1443–1448, 2007.

[16] P.M. Wensing and D.E. Orin. Development of high-span running long jumps for humanoids. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 222–227, 2014.

[17] P.S.A. Reitsma and N. Pollard. Perceptual metrics for character animation: Sensitivity to errors in ballistic motion. *ACM Trans. Graph.*, 22(3):537–542, 2003.

[18] A. Sulejmanpašić and J. Popović. Adaptation of performed ballistic motion. *ACM Trans. Graph.*, 24(1):165–179, 2005.

[19] P.S.A. Reitsma, J. Andrews, and N.S. Pollard. Effect of character animacy and preparatory motion on perceptual magnitude of errors in ballistic motion. *Comput. Graph. Forum*, 27(2):201–210, 2008.

[20] S. Ha, Y. Ye, and C. K. Liu. Falling and Landing Motion Control for Character Animation. *ACM Transactions on Graphics*, 31(6):1, November 2012.

[21] Sergey Levine and Jovan Popovic. Physically Plausible Simulation for Character Animation. In *Symposium on Computer Animation*, pages 221–230, 2012.

[22] Steve Tonneau, Rami Ali Al-Ashqar, Julien Pettré, Taku Komura, and Nicolas Mansard. Contact re-positioning under large environment deformation. *To appear in Computer Graphics Forum (Proc. Eurographics)*, 2016.

[23] K. Yamane and K. W. Sok. Planning and synthesizing superhero motions. In *Conference on Motion in Games*, pages 254–265, 2010.

[24] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[25] Tobias Kunz and Mike Stilman. Probabilistically complete kinodynamic planning for robot manipulators with acceleration limits. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 3713–3719, September 2014.

[26] J. Mirabel, S. Tonneau, P. Fernbach, A. Seppälä, M. Campana, N. Mansard, and F. Lamiraux. HPP: a new software for constrained motion planning. In *IEEE International Conference on Intelligent Robots and Systems (IROS) (to appear)*, 2016.

[27] Timothy Bretl and Sanjay Lall. Testing static equilibrium for legged robots. *IEEE Trans. Robotics*, 24(4):794–807, 2008.

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

[29] S. Caron, Q-C. Pham, and Y. Nakamura. Leveraging cone double description for multi-contact stability of humanoids with applications to statics and dynamics. In *Robotics: Science and System*, 2015.