A leap for physics-based character animation: automated gait cycle generation for characters of arbitrary morphologies using trajectory optimisation

Joshua Harford[1], *Stacey* Shield[2]

[1]Mechatronics, University of Cape Town, Cape Town, South Africa
[2]Professor of Mechatronics, University of Cape Town, Cape Town, South Africa

Abstract. This paper presents a framework that uses Trajectory Optimisation to automatically generate physically feasible 3D full body animations for legged characters of arbitrary morphology. The method is designed to be efficient, with animations generated in a matter of minutes. The user need not have any specific knowledge of physics, biomechanics or optimisation as the framework requires only a high-level description of the character's skeleton to generate realistic looking walking and running animations. This paper demonstrates a general, lightweight, and easy to use tool that will allow animators to bring their imaginary characters to life without the time, cost and expertise usually required to do so.

1 Introduction

The standard for realistic character animation in video games and films has risen dramatically over the past two decades, driven by rapid advances in 3D motion graphics. Innovations in rendering (e.g. ray tracing), motion capture, and physics-based simulation have all contributed to this progress. These simulations apply not only to characters, but also to complex environmental effects such as fluids, hair, and cloth.

Each technological leap raises audience expectations [1], and traditional techniques like keyframing, where animators manually pose characters, are increasingly unable to meet the threshold for realism. This challenge is particularly apparent for fictional or fantastical characters, where no real-world counterpart exists for motion capture. In such cases, the burden of realism falls entirely on the artist's intuition. A physics-based approach instead models the character and simulates its interaction with the world, enabling more believable motions grounded in physical laws rather than artistic interpretation.

While physics-based approaches like trajectory optimisation have shown great promise in generating physically realistic motions for specific robotic morphologies, such as quadrupeds [2, 3, 4] and humanoids [5, 6, 7], they have largely remained within the domain of robotics and have been applied only to specific morphologies, with the goal of achieving specific locomotion or manipulation tasks. There currently exists no widely adopted framework that leverages trajectory optimisation to generate animations for characters of

arbitrary morphologies. This paper presents such a framework, using trajectory optimisation to generate optimal motions of the character in 3D space, ensuring it conforms to the laws of physics throughout the simulation. It answers the question, "If such a character existed in the real world, how would it move?", and it answers it at a fraction of the time, cost and expertise usually required to do so.

Although this framework's primary use case is for film and video game animation, the underlying physics-based approach is accurate enough to apply to real-world robotics, granting it promise as a tool for quickly and easily generating candidate trajectories for real world robots of arbitrary morphologies as well.

2 Approach

The user starts by defining a list of rigid links that make up the character's skeleton. A specification of whether a link's endpoint is permitted to contact the ground is also given. It is an important consideration, as additional contact points greatly increase computational complexity. The initial state of the character must also be provided as it lets the optimiser know what position and orientation the character must start in. Additional information such as the individual link properties (length, mass, damping, rotational inertia) are provided manually, along with the location of the characters knee/elbow joints. This will inform the allowable range of motion and actuator torque limit of these joints, helping to ensure results remain believable. Finally, a target or goal state is defined to guide the desired motion

2.1 Trajectory optimisation formulation

The typical trajectory optimisation problem is structured as a continuous time problem in the following way. The objective is to determine the control inputs $\boldsymbol{u}(t)$ that produce a trajectory $\boldsymbol{x}(t)$ over some time interval T. The aim is to reach a target state from an initial state while minimising some objective function $J(\boldsymbol{x},\boldsymbol{u})$ and being subject to the satisfaction of the system's dynamics $\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u})$ and path constraints $\boldsymbol{h}(\boldsymbol{x},\boldsymbol{u})$ [2].

$$\text{find} \quad \boldsymbol{x}(t),\boldsymbol{u}(t) \quad for\ t \in [0,T] \tag{1.1}$$
$$\text{subject to} \quad \boldsymbol{x}(0) - x_0 = 0 \quad \text{(given initial state)} \tag{1.2}$$
$$\boldsymbol{x}(T) - x_T = 0 \quad \text{(desired final state)} \tag{1.3}$$
$$\dot{\boldsymbol{x}}(t) - \boldsymbol{f}\big(\boldsymbol{x}(t),\boldsymbol{u}(t)\big) = 0 \quad \text{(dynamic model)} \tag{1.4}$$
$$\boldsymbol{h}\big(\boldsymbol{x}(t),\boldsymbol{u}(t)\big) \le 0 \quad \text{(path constraints)} \tag{1.5}$$
$$x_{lower} \le \boldsymbol{x}(t) \le x_{upper} \quad \text{(path bound on state)} \tag{1.6}$$
$$u_{lower} \le \boldsymbol{u}(t) \le u_{upper} \quad \text{(path bound on control)} \tag{1.7}$$
$$\boldsymbol{x}(t),\boldsymbol{u}(t) = \arg\min\ J(\boldsymbol{x},\boldsymbol{u}) \tag{1.8}$$

The state vector $\boldsymbol{x}(t)$ is a vector containing $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$, where $\boldsymbol{q}$ contains the generalised coordinates of the system, i.e the variables related to the degrees of freedom of the model, and $\dot{\boldsymbol{q}}$ is the rate of change of these generalised coordinates. The objective function refers to the quantity in the optimisation process that we want to optimise. If we wish to mimic nature's tendency to minimise energy consumption, we can for example, set the objective function to minimise the sum of the actuator torques squared.

2.1.1 Discretisation into a nonlinear program

In a typical trajectory optimisation problem, as described in equations 1.1–1.8, the goal is to solve a system of differential equations where the decision variables are functions of time.

This formulation is inherently infinite-dimensional and highly nonlinear due to the underlying dynamics, contact interactions, and control constraints. Such problems are clearly intractable in their continuous form. Therefore, the task becomes one of transcription, i.e. converting the continuous-time problem into a finite-dimensional nonlinear program (NLP), where the decision variables are discrete vectors of real numbers. This discretization enables the use of gradient-based solvers to efficiently approximate solutions to the original problem. Pyomo, an open-source Python library was used to transcribe the problem into an NLP and was solved using an open source, gradient-based NLP solver called IPOPT which was used due to its effectiveness in large scale problems and because it is open source.

There are two main classes of transcription methods used in trajectory optimisation: shooting methods and collocation methods. Shooting methods, which can be classed into single or multiple shooting, solve the dynamics forward in time using initial guesses, while collocation methods transcribe both the state and control trajectories as decision variables and enforce the dynamics at discrete points along the trajectory. In this paper, we chose to use a direct collocation approach. The system dynamics, or equations of motion (EOMs), were enforced as algebraic constraints at N discrete time steps, using forward Euler integration, a first-order approximation, to propagate the state forward. While more advanced methods like trapezoidal or Hermite–Simpson collocation use higher-order integration schemes, Euler integration was chosen for its simplicity and computationally efficiency.

2.2 Dynamic model

A character can be thought of as a chain of $n+1$ bones, or links, connected by n actuated joints. The configuration of these links and joints defines the morphology of the character. By virtue of being in 3D space, the base link, or floating base, contains 3 translational degrees of freedom (DOFs), which represents the character's location in space in relation to the world frame. Each link contains 3 rotational DOFs. We only need to track the angle of the joints because we know how each link is connected to each other and can therefore calculate their position in space relative to the floating frame using basic trigonometry. Unlike the actuated joints, the floating base cannot be controlled directly. Instead, its motion is governed implicitly by the dynamics of the system, through the motion of the other links, gravitational effects, and contact forces with the environment. As a result, the full system has a total of $3(n+1)$ degrees of freedom, although only $3n$ of these are directly controllable. Therefore, our generalised coordinate vector takes the following form; where x, y, z represents the location of the character in 3D space and θ, ϕ, ψ represent the roll, yaw and pitch angles of the individual links:

$$\boldsymbol{q} = [x, y, z, \theta^1, \phi^1, \psi^1, \theta^2, \phi^2, \psi^2, \dots, \theta^n, \phi^n, \psi^n]^T \tag{2.1}$$

The animator must indicate whether the end of a link is permitted to make contact with the ground. While the character can reorient itself by actuating its joints, it can only translate through space by exerting forces against the environment. The initial and final states of the character must also be specified to inform the optimiser of the required starting and ending positions and orientations. In addition, the physical properties of each link, like mass, length, and rotational inertia, must be provided. The user also needs to specify which joints correspond to knees or elbows, as this information constrains the allowable joint ranges and torque limits, helping to ensure the resulting motion remains physically plausible and as intended.

2.2.1 Generating the equations of motion

In order to simulate some model, one would need to know how that model's state changes over time. To determine this, one can develop an ordinary differential equation (ODE) in the form:

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}) \tag{2.2}$$

Where $\dot{\boldsymbol{x}}$ is a function of $\boldsymbol{x}$ and $\boldsymbol{u}$. Since $\boldsymbol{x}$ contains $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$, it follows that $\dot{\boldsymbol{x}}$ contains $\ddot{\boldsymbol{q}}$, the equations of motion for our generalised coordinates $\boldsymbol{q} \in \mathbb{R}^{6(n+1)}$. These EOMs are a set of expressions that relate joint accelerations to $\boldsymbol{u}$, the control inputs of the system, which consist of the internal torques applied directly from the character's joints, gravity and the ground reaction forces (GRFs) which act on the characters feet to prevent ground penetration. The following section will briefly describe how these EOMs are dynamically derived using Euler-Lagrange dynamics for a constrained rigid body system of arbitrary morphology.

The process starts by defining $\boldsymbol{r}_n^0 \in \mathbb{R}^3$, the position of the n'th links center of mass (COM) in the inertial reference frame.

$$\boldsymbol{r}_n^0 = \boldsymbol{R}_0^b . \boldsymbol{r}_n^b \tag{2.3}$$

This is done by applying a Euler rates rotation matrix, $\boldsymbol{R}_0^b \in \mathbb{R}^{3\times3}$, to the COM position in the base frame, $\boldsymbol{r}_n^b$, which is derived from the user's physical description of the character's bone structure.

The velocity of each link can be found by multiplying the rate of change of $\boldsymbol{q}$ with the Jacobian of the position vector $\boldsymbol{J_p}$, which takes the partial derivative of $\boldsymbol{r}_n^0$ with respect to $\boldsymbol{q}$. Essentially, $\boldsymbol{J_p}$ maps changes in generalized coordinates to changes in cartesian position and allows $\boldsymbol{u}$ to express the linear velocity of link n as:

$$\dot{\boldsymbol{r}}_{\boldsymbol{n}} = \boldsymbol{J_p} . \dot{\boldsymbol{q}} \tag{2.4}$$

the Lagrangian, L, is used to calculate the EOMs and is defined in equation (2.5) as the difference between the kinetic and potential energy of a system. These quantities are calculated according to equations (2.6) and (2.7) respectively.

$$L = T - V \tag{2.5}$$

$$T = \frac{1}{2}\sum_{n=1}^{j}(m_n . \dot{\boldsymbol{r}}_n^2 + \boldsymbol{I}_n . \boldsymbol{w}_n^2) \tag{2.6}$$

$$V = \sum_{n=1}^{j}(m_n . g . z_n) \tag{2.7}$$

Where m_n, $\boldsymbol{I}_n$ and $\boldsymbol{w_n}$ are the masses, inertias and angular velocities of the n'th links respectively. z_n is the vertical height of the n'th link from the ground and g is the gravitational constant.

The generalised force vector $\boldsymbol{Q}_j$ represents the non-conservative forces that act on the body and allows us to describe the effect of external forces, $\boldsymbol{F}_i$, acting at position $\boldsymbol{r_i}$ in terms of the system's generalised coordinates, $\boldsymbol{q_j}$, where $j \in [1,2,\ldots,n]$. The number of forces, k, is determined by how many foot contacts where specified by the user.

$$\boldsymbol{Q}_j = \sum_{i=1}^{k} (\boldsymbol{F}_i \cdot \frac{\partial \boldsymbol{r_i}}{\partial \boldsymbol{q_j}}) \tag{2.8}$$

As a final step, one can use the Lagrangian and the generalised force vector to acquire expressions for each of the generalised coordinate's accelerations. These are known as the equations of motion and are calculated using the equation below:

$$\boldsymbol{EOM} = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\boldsymbol{q}}_i}\right) - \frac{\partial L}{\partial \boldsymbol{q_i}} - \boldsymbol{Q} \tag{2.9}$$

This gives us an ODE in the form $\ddot{\boldsymbol{q}} = f(\boldsymbol{q}, \boldsymbol{Q})$.

2.3 Contact model

The contact model is a crucial component of the framework, as it governs how the character interacts with the world. In the real world, when we take a step, at the moment of contact, the ground applies the exact force required to stop our foot from penetrating the floor. Simulating this force can prove to be a non-trivial task due to the inherently discontinuous nature of walking. Just before a foot strikes the ground, it has a negative velocity. At impact, an impulsive ground reaction force (GRF) abruptly brings that velocity to zero. This discontinuity poses a challenge for gradient-based solvers like IPOPT, which require smooth, differentiable dynamics.

2.3.1 Complementarity

We use a complementarity condition to address the discontinuity problem. This treats contacts as a continuous process throughout the simulation, rather than a discrete process. The condition is stated as follows:

$$A.B = 0 \qquad where\ A \geq 0, B \geq 0 \tag{3.1}$$

The product of A and B must equal 0, where A represents the foot's height above the ground and B represents the magnitude of the vertical ground reaction force. This ensures that contact forces only activate when the foot makes contact with the ground. Only one value may be non-zero at a time, allowing the optimiser to discover contact timing implicitly while preventing foot-floor interpenetration. This is known as a contact-implicit approach. Although this can be more computationally demanding, it tends to result in more stable and physically accurate simulations [8]. Since we are dealing with offline animation instead of online robotics, time is much less of a concern. A result that looks and behaves better is prioritised over fast results.

In practice, this problem formulation does have to be altered slightly to make it easier to solve. IPOPT struggles with complementarity conditions because the solution is along the boundary of A or B, meaning there is no surface area of the solution space. IPOPT, being an interior point solver [9], needs to have an interior to search through. This can be remedied by equating the product of AB to some small value, epsilon (ε), that relaxes the complementarity constraint and gives a small interior to the solution space. The revised complementarity constraint can be shown below:

$$z_{foot}.GRF_z = \varepsilon \qquad where\ z_{foot} \geq 0, GRF_z \geq 0 \tag{3.2}$$

The objective function can now be to find the smallest value of ε that generates a feasible solution with no ground penetration, i.e., We try to minimise ε, whilst enforcing the constraint that $z_{foot} \geq 0$ throughout the entire trajectory.

2.3.2 Variable time step

The discontinuous nature of contact dynamics poses a significant challenge for trajectory optimisation. Direct collocation enforces constraints at discrete time steps, making it difficult to accurately capture impulsive events such as ground contacts without extremely high sampling rates, which can increase computation time substantially. To mitigate this, a variable time step scheme was introduced, where the duration of each time step is treated as a decision variable. This allows the solver to allocate higher resolution where it is most needed (e.g., near impact events) and lower resolution elsewhere. Specifically, a scalar decision variable $h \in [0.8,1]$ modulates a fixed master time h_m between 80% and 100% its base value. Thus, the effective time step used during integration is $h.h_m$.

2.3.3 Friction cone approximation

The horizontal components of the ground reaction force vector are the frictional forces. They are defined via a polyhedral approximation of a friction cone, which restricts the 3D GRF vector to act within the interior of a friction cone whose gradient is defined by the coefficient of static friction, μ. In the event of a sliding contact, the GRF vector will lie on the boundary of the friction cone, providing the maximum allowable friction, and will act in opposition to the relative tangential velocity of the foot, V^T. The polyhedral approximation avoids the use of nonlinear trigonometric functions by discretizing the cone into a set of k unit vectors, d_k. Higher values of k lead to a greater approximation of the cone. In this paper, k was chosen to be 10. The friction force can therefore be written as:

$$GRF_x = \mu . GRF_z \sum_{i=1}^{k} (d_k . \alpha_k) \tag{3.3}$$

The direction of the force is determined by which unit vector is activated via the activation variable α_k, which can take on a value between 0 and 1. Another constraint enforces that the sum of all the activation variables must be equal 1 as shown below:

$$\sum_{i=1}^{k} \alpha_i^k = 1 \tag{3.4}$$

Additionally, equation (3.5) is another complementarity condition that ensures only one activation variable is none-zero when sliding occurs.

$$\alpha_i^k \perp \gamma - d_i^k . V_i^T \qquad where\ \gamma - d_i^k . V_i^T \geq 0 \quad \forall\ i = 1,2,\dots,k \tag{3.5}$$

Here $\perp$ is the symbol indicating the compliment and γ is a positive auxiliary variable that represents the maximum tangential component of the velocity in any given direction. It helps guide the model to apply the friction force in the direction that opposes the motion of the foot. When the difference between the maximally aligned projection to the actual tangential

velocity, V^T, and the current $d_i^k.V_i^T$ is zero, then α_i^k has the freedom to be non-zero. But if this is not the case, then α_i^k must be zero.

3 Results and discussion

This section showcases four examples of characters that are generated and controlled using the framework outlined in this paper. All of the characters were initialised at rest in some neutral position. Throughout the movement, the motion was restricted to the character's sagittal plane to encourage natural looking movement. In this same vein, each character's height was constrained to around ±0.4m about its neutral position to discourage erratic movements. Path constraints like these also help the solver converge quicker as they reduce the search space. As a final condition, the characters were tasked with reaching a distance of at least 8m, which gave sufficient distance to showcase their gait.

The objective function chosen was simply to minimise the small value ε to ensure the complementarity conditions were applied accurately. A two-stage objective function was tested, with the second stage being an energy minimisation function, but this resulted in much longer solve times with no significant visual improvement. Each simulation was run using 150 nodes, a master timestep of $h_m = 0.02s$, 10 direction vectors for the friction cone and a high coefficient of static friction of $\mu = 5$ was chosen to discourage sliding.

A key metric of interest for most users of this method is the time required to solve the equations of motion and generate the resulting animation. Importantly, once the EOMs have been derived for a given character, they are saved to a file and never need to be recomputed. If the user wants to tweak limb lengths or masses, alter certain constraints, or change the end goal, they can do so without needing to wait to recalculate the EOMs again.

Table 1 presents key evaluation metrics for each character. The number of degrees of freedom had the most significant impact on both the time required to derive the equations of motion and to solve the optimisation problem. Additionally, an increased number of contact points, each requiring the enforcement of complementarity constraints for contact and friction, correlated with longer optimiser solver times.

Table 1. Comparison of model complexity and solve times for each character.

	Finger Runner	**Canine**	**Humanoid**	**Giraffe**
Degrees of freedom	5	11	11	13
Number of contacts	2	4	2	4
EOM solve	18m 55s	126m 20s	133m 10s	277m 03s
IPOPT solve	1m 8s	24m 34s	10m 20s	38m 29s

An optimal solution was found in each case using a M1 MacBook air laptop, which has the processing power roughly equivalent to an intel core i5 CPU. The results were visualised in a 3D plot using the matplotlib python library with each frame of the video corresponding to the updated state of the character at the corresponding collocation node. Quantitatively evaluating how natural a generated animation appears is inherently difficult, as it is largely subjective and guided by intuition. While objective metrics, such as those displayed in Table 1, exist for specific aspects of motion; overall realism is best assessed visually. To that end, a sequence of video frames is shown below to illustrate the generated locomotion for each character. Although the static nature of a paper limits the ability to convey motion, we believe the animations demonstrate realistic and physically plausible movements. It should be noted

that the illustrations below serve solely to demonstrate the trajectories of the character's body and do not incorporate additional visual effects that contribute to realism, such as body shape, surface textures, or lighting. These aesthetic elements are beyond the scope of this work and are left for future development of the framework.

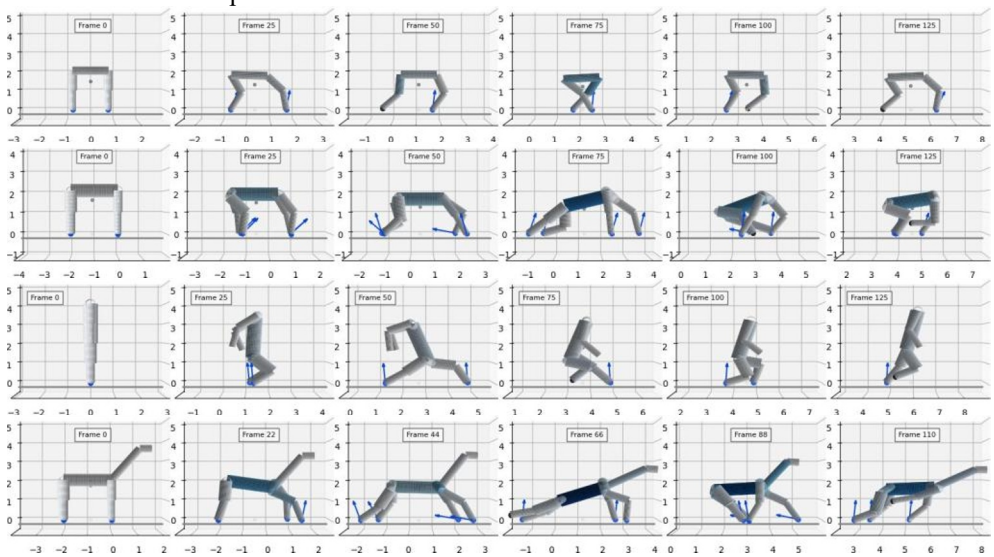

Fig. 1. Generated animation sequences for each character. From top to bottom: Finger Runner, Canine, Humanoid, and Giraffe.

A key feature of this framework is that the characters are never explicitly "taught" how to walk. Instead, the underlying dynamics required to move from point A to point B emerge naturally from the optimisation process. For instance, the difference in gait between the two-legged Finger Runner and the four-legged Canine highlights how the optimiser adapts its strategy to different morphologies. More limbs allow for richer and more dynamic solutions, leading to diverse locomotion behaviours. As shown in Fig. 1, the framework is capable of handling humanoid models as well, which are widely regarded as some of the most challenging to control. The Giraffe, with its long neck, further illustrates the flexibility of this method, enabling animators to experiment with unique and imaginative designs.

3.1 Applications in industrial robotics

While the primary application of this work lies in the film and video game industry, the core foundation of this approach is grounded in real world robotics [2-7]. Implementing this control technique in real robots would require a more accurate actuator model and possibly the addition of a feedback control loop which would help with responding to noise and disturbances in real time.

With the addition of these features, this framework would offer a rapid method for prototyping and evaluating locomotion strategies for novel robotic designs in industrial settings where environmental and technical specifications are rapidly changing. This would allow engineers to quickly explore candidate designs while being flexible enough to easily iterate on design specifications such as payload capacity, joint limits and energy usage.

3.2 Future work

Future work could explore the inclusion of more complex terrain to better demonstrate the framework's ability to generate robust and adaptable locomotion strategies. Additionally, replacing full-body dynamics with simplified models could significantly reduce solve times, enabling faster iteration and therefore greater experimentation. Ultimately, integrating the system with 3D rendering software such as Blender or Unreal Engine would allow for high-quality visualisation and practical deployment in animation pipelines.

References

[1] H. Van Welbergen, B. J. Van Basten, A. Egges, Z. M. Ruttkay, and M. H. Overmars, "*Real time animation of virtual humans: A trade-off between naturalness and control*" Computer Graphics Forum, vol. **29**, no. 8, p. 2530–2554 (2010). https://doi.org/10.1111/j.1467-8659.2010.01822.x

[2] A. Winkler, F. Farshidian, D. Pardo, M. Neunert, J. Buchli. "*Fast Trajectory Optimization for Legged Robots Using Vertex-Based ZMP Constraints*". IEEE Robotics and Automation Letters. PP. (2017). https://doi.org/10.48550/arXiv.1705.10313

[3] M. Kalakrishnan, J. Buchli, P. Pastor, M. Mistry, and S. Schaal, "*Fast, robust quadruped locomotion over challenging terrain*", IEEE International Conference on Robotics and Automation, pp. 2665–2670 (2010). 10.1109/ROBOT.2010.5509805

[4] J. Z. Kolter, M. P. Rodgers, and A. Y. Ng, "*A control architecture for quadruped locomotion over rough terrain*", IEEE International Conference on Robotics and Automation, pp. 811–818 (2008). 10.1109/ROBOT.2008.4543305

[5] I. Mordatch, E. Todorov, and z. Popović, "*Discovery of complex behaviors through contact-invariant optimization",* ACM Transactions on Graphics*, 31(4)* (2012). https://doi.org/10.1145/2185520.218553

[6] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa, "*Biped walking pattern generation by using preview control of zero-moment point*", *IEEE* International Conference on Robotics and Automation, pp. 1620–1626, 2003. 10.1109/ROBOT.2003.1241826

[7] M. Al Borno, M. de Lasa, A. Hertzmann, "*Trajectory optimization for full-body movements with complex contacts*". IEEE Transactions on Visualization and Computer Graphics (2013). 10.1109/TVCG.2012.325

[8] S. Le Cleac'h *et al.*, "*Fast Contact-Implicit Model Predictive Control*," in IEEE Transactions on Robotics, vol. **40**, pp. 1617-1629 (2024). 10.1109/TRO.2024.3351554

[9] A. Waechter and L. T. Biegler, "*On the implementation of an interior- point filter line-search algorithm for large-scale nonlinear programming"*, vol. **106**, no. 1. (2006). https://doi.org/10.1007/s10107-004-0559-y