Procedural Locomotion Engine for Digital Humans

Joshua Aurand
CSE MSc
aurandj@ethz.ch

Marie Jaillot
CS MSc
mjaillot@ethz.ch

Rafael Steiner
CSE MSc
steinraf@ethz.ch

Yucheng Wang
CS BSc Exchange
yucwang@ethz.ch

Abstract

Human locomotion can be challenging to reproduce, as it involves synchronizing the legs and arms motion and requires a careful representation of ankle movements to produce natural results. We present a real-time procedural locomotion controller for interactively controlled digital humans. Our approach is based on translating Biomechanical principles into control and planning strategies. It is able to synthesize walking and running motion in straight and curved lines and supports walking on uneven terrains. Our implementation can be found on GitHub [6].

1. Introduction

In an increasingly digital society, digital humans are becoming more and more ubiquitous. These digital characters are important for many applications ranging from entertainment [12] to healthcare [5] and even industry applications such as in manufacturing [36]. In many of these applications the characters need to interact with their environment which often involves walking around. Hence, it is important to be able to realistically animate the motion of digital humans.

Creating natural motion is challenging since humans can easily notice if something about the motion is off. Hence, the main challenge lies in avoiding the uncanny valley [21]. While there are various different methods, they all have their own issues and shortcomings which will be explained in Section 2. We aim to address these problems by relying on biomechanical principles.

Our method is based on a heavily modified version of a simple controller for a quadrupedal robot. The quadrupedal motion was based on simple precomputed limb trajectories according to a footstep planner. The joint configuration of the robot is then set using Inverse Kinematics, which maps target limb positions to joint configurations that result in those targets. Each leg of the robot was controlled by tracking a single point (an end effector) on that leg. However, for the bipedal motion a single end effector per limb is not sufficient to achieve realistic motion. Instead, we track both the position of the heel and the toe. The computation of the heel and toe trajectories is heavily inspired by the gait cycle, a biomechanical concept [13] [26]. Hence, we needed more fine-grained control than what was offered by the quadrupedal controller. We achieved this by rewriting all of planning and controls from scratch. Bipedal motion is also supported by upper body movement which we directly control by setting the respective joint angles.

In summary, our main contribution lies in the formulation of biomechanically inspired control and planning strategies that are able to synthesize realistic bipedal motion based on interactive user inputs.

2. Related Work

Several approaches have emerged to enable the generation of natural locomotion for articulated models. These approaches include Kinematics-based, Physics-based, Reinforcement-Learning-based, and Motion-Matching-based methods. In this section, we will provide a concise overview of these methods, highlighting their respective shortcomings. Finally we will give a review of the main biomechanical concept that guided our solution: the human gait cycle.

Kinematics: Kinematics-based control is a procedural locomotion control method that utilizes various techniques such as Motion Fields [17], Interpolated Motion Graphs [28], and Low-Dimensional Embeddings [18] to generate motion. The effectiveness of the kinematic model is directly linked to the quality and diversity of the motion data [23]. Moreover, the generalization of the kinematic-based control approach is constrained when faced with new or unfamiliar environments. The model's performance tends to degrade when confronted with scenarios outside the scope of its training data.

Physics: To enhance the versatility and robustness of the controller, physics-based models are employed. These models simulate the underlying dynamics and constraints of the system, enabling the controller to respond to external forces and environmental conditions [11, 34]. By utilizing a single model-free physics-based framework, a broad spectrum of motion can be simulated [3]. While physics-based models have proven useful in generating motion controllers for both humanoid and non-humanoid robots, they still encounter challenges related to motion quality and long-term planning [23]. They often struggle with generating coherent and realistic long-term motion trajectories.

Reinforcement Learning: Reinforcement learning methods have gained significant popularity in the locomotion generation field, particularly for developing controllers that target various actions. These methods employ either the Value Iteration [10, 18, 24] or utilize Deep Neural Network models [19, 20, 25] with techniques like Policy Gradient [29, 30]. Traditionally, reinforcement learning algorithms require well-defined reward functions to guide the learning process. It becomes challenging to design conditioned reward functions that effectively capture the complexity and nuances of natural motion [33]. Common problems such as speed control, gait synchronization, and natural arm swing are not easily addressed by artificially designed reward functions [23]. Moreover, difficult tasks such as maintaining balance, adapting to different terrains, and achieving smooth transitions between different locomotion modes further complicate the reward functions.

Motion Matching: Motion matching is another approach to motion generation that utilizes deep learning methods [23]. It relies on generative neural network models to establish responsive simulated character controllers based on unstructured motion capture data [1, 7]. This technique exhibits strong predictability, allowing for accurate and realistic motion generation. However, the effectiveness and quality of motion matching heavily depend on the diversity and richness of the motion capture dataset. Another challenge associated with motion matching is the increasing memory demand as the motion capture data array grows [15]. Storing and processing large amounts of motion capture data can be resource-intensive, requiring significant memory resources.

The Human Gait Cycle: In biomechanical literature, the human gait cycle is generally modelled as two phases; The stance and the swing phase [13] [26]. In the stance phase, the foot is flat on the ground and starts ahead of the center of mass. The weight is fully supported by the foot and as the gait cycle continues, the foot moves further back until the heel eventually leaves the ground. Towards the end of the stance, the weight is slowly released from the toe which then stops contact with the ground as well. This final lifting of the toe initializes the swing phase. In this phase the leg moves forward, aiming at the target location of the heel strike. During heel strike, the heel starts contact with the ground while the toe slowly moves towards the ground as well, starting to support weight on this foot. This completes one gait cycle. This cycle is repeated for both legs, with an offset of half a cycle length [16].

3. Method

To be able to procedurally synthesize full-body motion in real time, we need to convert our targets for the end effectors to joint angles. This can be achieved by using Inverse Kinematics, which will be explained in detail in Section 3.1. Because these targets are sparse, we make use of various interpolation techniques explained in Section 3.2 to generate smooth motion, which is essential for realism. To find the end effector targets in the first place, we use a biomechanically-inspired controller for the legs (Section 3.4). The motion of the legs then influences the movement of the base (Section 3.5) and the swing of the arms (Section 3.6). As extensions to walking in a straight line, we discuss solution to three additional gait styles: Walking along curved trajectories Section 3.7, walking on uneven terrain Section 3.8 and finally we also investigate a controller for running Section 3.9.

3.1. Inverse Kinematics

3.1.1 The Inverse Kinematics Problem

Inverse Kinematics (IK) refers to the problem of finding a joint configuration $\Theta := \{\theta_j\}_{j=1}^{n_{\text{joints}}}$ that results in specified target positions $\{\vec{p}_i\}_{i=1}^{n_{\text{EE}}}$ (in the world coordinate frame) for a set of n_{EE} different end effectors. More specifically, IK refers to the following non-linear least squares problem:

$$\Theta^* = \arg\min_{\Theta} \sum_{i=1}^{n_{\text{EE}}} ||\vec{p}_i - \mathbf{FK}(\vec{q}_i;\Theta)||_2^2 \tag{1}$$

where $\vec{q}_i$ are the end effector positions in the body coordinate system and $\mathbf{FK}(\vec{q}_i;\Theta)$ is the Forward Kinematics function that maps the end effector positions from the body to the world coordinate system. A common approach for solving this optimization problem is the Gauss-Newton method where the joint configuration is iteratively updated according to the update rule

$$\Theta^{k+1} = \Theta^k + (\boldsymbol{J}^T\boldsymbol{J})^{-1}(\boldsymbol{J}^T(\vec{p}_i - \mathbf{FK}(\vec{q}_i;\Theta^k)) \tag{2}$$

where $\boldsymbol{J}$ is the Jacobian $\frac{\partial \mathbf{FK}(\vec{q}_i;\Theta^k)}{\partial \Theta^k}$. Despite its simplicity, this method achieves good tracking performance of the desired end effector positions.

3.1.2 Joint Constraints

Naively applying the Gauss-Newton iteration will result in unnatural poses as seen in Figure 1. This is due to the

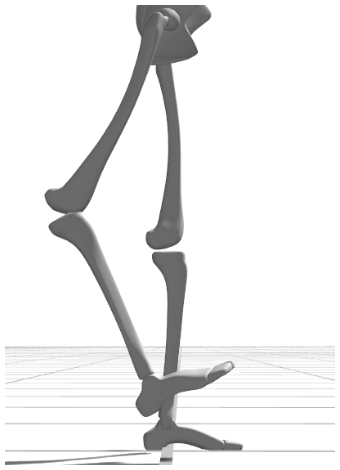
Figure 1. While the target position of the heel is perfectly tracked, the resulting pose is unnatural due to not respecting joint limits.

unconstrained nature of Equation (1). As there are multiple possible configurations that result in a given target position, it is vital to choose the pose that is most natural. This can be achieved by introducing joint limits, as human joints can not be arbitrarily rotated. Hence, we need to add the constraints $\theta_j^{\text{min}} \leq \theta_j \leq \theta_j^{\text{max}}$ to Equation (1). A straightforward modification would be to apply clipping to the solution proposed by the iterative method, i.e. setting $\tilde{\theta}_j = \min(\max(\theta_j, \theta_j^{\text{min}}), \theta_j^{\text{max}})$. While this results in configurations that respect the joint limits, the tracking performance can be significantly hindered as the solver can not recover from local minima corresponding to unnatural poses. A more sophisticated solution is applying regularization, i.e. adding an energy function $\lambda R(\Theta)$ to the objective function that nudges the solver towards realistic poses. We tested the following formulations:

$$R_1(\theta_j) = \begin{cases} 0 & \text{if } \theta_j \in [\theta_j^{\text{min}}, \theta_j^{\text{max}}] \\ (\theta_j - \theta_j^{\text{max}})^2 & \text{if } \theta_j^{\text{max}} < \theta_j \\ (\theta_j - \theta_j^{\text{min}})^2 & \text{if } \theta_j < \theta_j^{\text{min}} \end{cases}$$

$$R_2(\theta_j) = \begin{cases} -(\theta_j - \theta_j^{\text{max}})^2 & \text{if } |\theta_j - \theta_j^{\text{max}}| < \epsilon \\ -(\theta_j - \theta_j^{\text{min}})^2 & \text{if } |\theta_j - \theta_j^{\text{min}}| < \epsilon \\ 0 & \text{else} \end{cases}$$

$$R_3(\theta_j) = \frac{(\theta_j^{\text{max}} - \theta_j^{\text{min}})}{4(\theta_j^{\text{max}} - \theta_j)^2(\theta_j - \theta_j^{\text{min}})^2}$$

In all cases, the full energy function was simply the sum of all the individual terms. R_1 is essentially a soft version of the clipping operation and hence has the same problem of not being able to recover from bad poses. R_2 can be thought of as a spring model with finite range ϵ. The issue with this model is the required fine-tuning of the spring range ϵ. R_3 is the index of avoidance proposed by Wan et. al. [32]. It is minimized for $\theta_j = (\theta_j^{\text{min}} + \theta_j^{\text{max}})/2$, i.e. when θ_j is exactly in the middle of its two limits. While this leads to good limit avoidance, it is not suited for humans, where the joint limits are often asymmetric with the rest pose being far from the middle of the limits (e.g. the limits for a knee are roughly $[0°, 150°]$, with the rest pose at $0°$). In the end, we achieved the best results by using an industry standard, dedicated solver for constrained non-linear least squares problems, namely Ceres [2]. Ceres achieved high tracking performance while mostly avoiding unnatural poses and being computationally efficient.

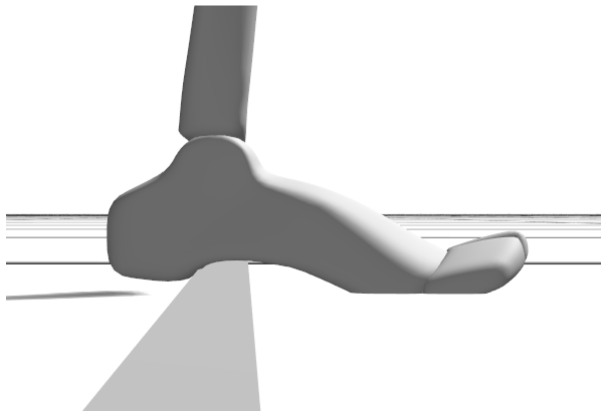
Figure 2. Due to loose joint constraints, the Inverse Kinematics solver can create unnatural poses.

3.1.3 End Effector Placement

However, even Ceres resulted in slightly unnatural poses in some cases. This is due to its capabilities of exploiting loose joint constraints. These artifacts mostly occurred for the foot itself, where the solver could exploit heel and toe joint rotations as shown in Figure 2. For walking, this problem can be solved by careful placement of the end effectors: for the purpose of the IK solver, the heel end effector is treated as an extension of the shin and the toe end effector is considered part of the foot and not the actual toe rigid body. Due to this simplification, ankle joint rotations do not affect the tracking error of the heel and toe joint rotations do not affect the tracking error of the toe. This simplification works well since all of the poses encountered during walking are relatively close to the idle pose where there are no heel and toe rotations.

3.2. Interpolation

3.2.1 Piecewise Linear Interpolation

The simplest type of interpolation is linear. Given two control points at times t_i, t_{i+1} with values y_i, y_{i+1} (which can be scalars or vectors) the interpolant f on the interval $[t_i, t_{i+1}]$ is defined as

$$f(t) = y_i + (y_{i+1} - y_i)\frac{t - t_i}{t_{i+1} - t_i}. \tag{3}$$

This method is simple, efficient and produces continuous curves. However, the resulting curves are not smooth. In general the interpolant will not be differentiable at any of the control points.

3.3. Spline Interpolation

Spline interpolation is a generalization of piecewise linear interpolation. The values of the spline are still defined by the control points but there are additional smoothness constraints. We chose Catmull-Rom splines [9]. On each interval between two control points this spline is a cubic polynomial. Given control points at t_i, t_{i+1} with values y_i, y_{i+1} the interpolant at time $t \in [t_i, t_{i+1}]$ is computed according to

$$\begin{aligned} f(\tau) = y_i&(2\tau^3 - 3\tau^2 + 1) \\ &+ m_i(\tau^3 - 2\tau^2 + \tau) \\ &+ y_{i+1}(-2\tau^3 + 3\tau^2) \\ &+ m_{i+1}(\tau^3 - \tau^2) \end{aligned} \tag{4}$$

where $\tau = (t - t_i)/(t_{i+1} - t_i)$ and m_i is the slope estimate at control point i defined as

$$m_i = \frac{1}{2}\left(\frac{y_{i+1} - y_i}{t_{i+1} - t_i} + \frac{y_i - y_{i-1}}{t_i - t_{i-1}}\right). \tag{5}$$

So the slope estimate m_i is simply the average of the average slopes in the two intervals next to control point i. Catmull-Rom splines have a number of desirable properties for animation. Due to their locality (i.e. the value at control point i only affects the spline in the two neighboring intervals), the interpolant can be computed efficiently using the closed form given in Equation (4). Additionally, the resulting interpolant is continuously differentiable, resulting in smooth looking animations.

3.3.1 Quaternion Interpolation

Naive linear interpolation connects two points in $\mathbb{R}^n$ by the shortest path between them, i.e. a straight line. However, quaternions representing orientations correspond to points on a unit sphere. Hence, linearly interpolating them does not produce valid orientations. Spherical linear interpolation respects the quaternion structure and instead follows the shortest path between two points restricted to the unit sphere. It can be computed as follows:

$$\text{slerp}(q_0, q_1, t) = \frac{\sin((1-t)\theta)}{\sin(\theta)}q_0 + \frac{\sin(t\theta)}{\sin(\theta)}q_1 \tag{6}$$

where $\theta = q_0 \cdot q_1$ is the angle between the two quaternions.

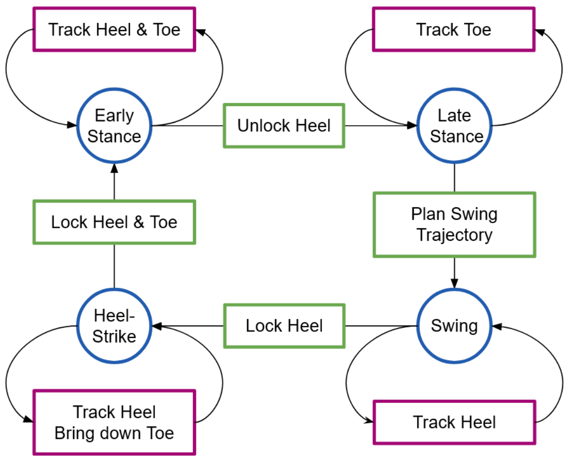

Figure 3. Overview over the control scheme for the leg movement. The blue boxes correspond to the different states in the gait cycle. Green boxes correspond to transitions between different states where new targets for the Inverse Kinematics solver are computed. Purple boxes show the targets that are set for the Inverse Kinematics solver during the corresponding states of the gait cycle.

3.4. Leg Controller

As a representation in code, we chose to partition the gait cycle for the lower body into three phases: 50% `Stance`, 40 % `Swing` and 10% `HeelStrike`. The Stance phase also includes the Toe-Off, which occurs in the second half. Each phase comes with its own targets and constraints, which are explained in detail in the following sections. Figure 3 shows a concise summary of how the leg controller works. The gait cycle for each leg is handled separately but synchronized for an offset of half a cycle length. Because the possible targets are sparse, we make use of interpolation to create smooth trajectories.

3.4.1 Stance Phase

The Stance Phase is subdivided into two parts, each occupying half of the total duration. The early stance is characterized by the foot being fully in contact with the floor, but not sliding around. This contact constraint is realized by setting the IK targets to the current position of the toe and heel. The heel constraint is slowly released, automatically causing the heel to lift off as intended. To establish firm ground contact during the stance the toe joint is automatically set such that the toe is parallel to the ground.

3.4.2 Swing Phase

The targets of the swing phase get precomputed at the start of the trajectory and then interpolated through Catmull-Rom splines. The swing trajectory contains four target

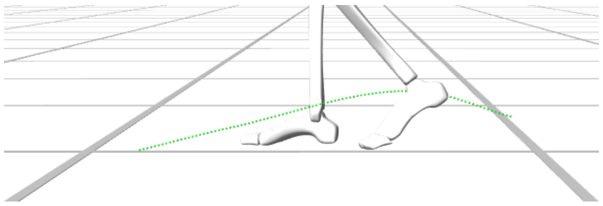
Figure 4. Visualization of the Swing Trajectory of the heel

points for the heel end effector. It starts out at the current position and ends at a target position that is computed as $\vec{p}_{end} = \vec{p}_{resting} + k \cdot \vec{v}_{current} + s \cdot \hat{\vec{v}}_{current}$. The intuition behind this formula is that for a zero velocity, we would like to return to the default configuration and not keep the heel lagging behind the body. k is the duration of the swing phase and accounts for the movement of the base, while s is an offset depending on the step length. The two additional control points are chosen such that the vertical offsets of the trajectory match up with observation. The first control point is chosen at time *t = 0.25* and has a vertical offset of *0.07*. The second control point is located at *t = 0.75* and vertical offset *0.03*. This results in a trajectory as shown in Figure 4. In addition the toe joint is moved back to it's default position during the first 10% of the swing phase.

3.4.3 Heel Strike Phase

During heel strike, the heel is fixed to the ground. This is achieved by setting the IK target to the position that was targeted at the end of the swing phase. This helps the heel reach the ground, even if that was not achieved during the swing phase. This can occur when the joints hit their limits during the full extension of the knee in the swing phase. The toe is linearly interpolated towards the resting position on the ground in front of the heel, making it slowly approach the ground for touch down.

3.5. Base Movement

In reality the base of a bipedal is not actuated. It's movement follows the movement of the actuated limbs due to multibody dynamics. However, since our animation is not physics-based we treat the base as if it was actuated and directly control its position and orientation based on user given velocity commands. In particular the user can input a forward velocity $\vec{v}$ and a turning speed ω. The base position $\vec{p}$ and orientation φ during each timestep are then updated using a simple Euler integrator:

$$\begin{aligned} \vec{p}_{i+1} &= \vec{p}_i + \Delta t q_{heading,i} * \vec{v}_i \\ \varphi_{i+1} &= \varphi_i + \Delta t \omega_i \end{aligned} \tag{7}$$

where $q_{heading,i}$ is a quaternion corresponding to the heading orientation at timestep i. While the vertical movement of the base is often modelled using an inverted pendulum model [11] we opted for a simpler solution. A small sinusoidal offset is added to the vertical position that achieves close results to an inverted pendulum model. In particular the vertical offset is $0.02 \sin(4\pi\tau)$ where $\tau \in [0, 1]$ is the current progress during the gait cycle.

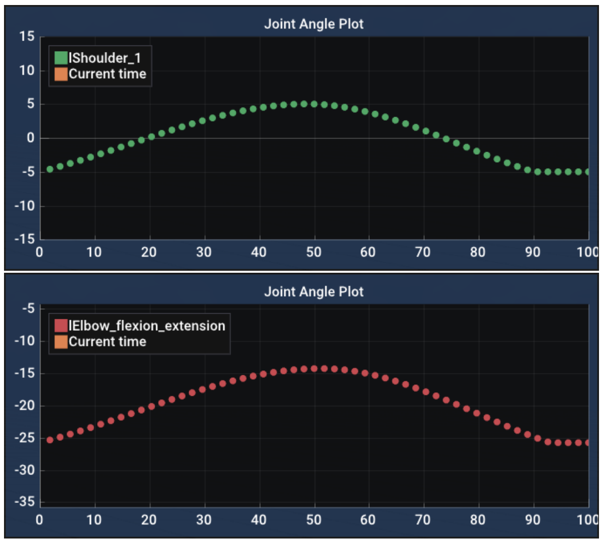

Figure 5. The trajectory of the left shoulder and elbow rotation in the sagittal plane during one gait cycle for walking.

3.6. Arm Swing

A distinctive feature of the human gait is the arm swing, where each arm swing is synchronized with the opposite leg. Mechanically, the main function of the arm swing is counteracting the torque on the vertical body axis induced by the leg swing [27]. Due to this, the gait becomes more energy efficient and has reduced metabolic costs [4]. Additionally, the arm swing can help stabilizing the gait against external perturbations [8].

3.6.1 Controlling the Arm Swing

As previously explained, the leg motion is controlled by tracking heel and toe positions via IK. By planning a hand trajectory the arms can also be controlled with IK. This approach however, proved to result in unsatisfactory animations. Planning a realistic hand trajectory is difficult, as even slight errors compared to a real arm swing result in unnatural movement. A simpler solution, resulting in more natural movement, is to directly control all the arm joints. Joint angle trajectories for one gait cycle are precomputed based on Catmull-Rom splines. The control points of the splines are set according to a combination of measured values found in Biomechanics literature [14], [35] and manual adjustments to improve the look of the animations. Fig-

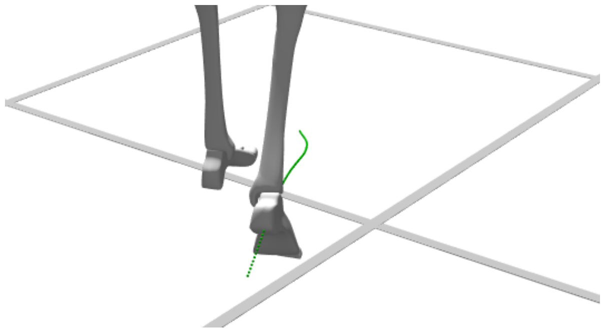
Figure 6. Visualization of a curved Swing Trajectory

ure 5 shows the resulting trajectories (note that our gait cycle is in the order (Stance, Swing, Heelstrike), while other literature typically starts with the Heelstrike). These trajectories are in phase with the right leg. The trajectories for the right shoulder and elbow are exactly analogous, except they are in phase with the left leg. In addition to the arm swing we also included small upper body rotations around the spine. This rotation follows a simple sinusoidal curve given as $\theta_{\text{spine}}(\tau) = 2.0° \sin(2\pi\tau - \frac{3}{5}\pi)$ where $\tau \in [0, 1]$ is the current progress in the gait cycle.

3.7. Turning

To account for curved trajectories, the heading of the robot base needs to be taken into consideration. To make the swing phase follow a curved trajectory, the default swing path (Figure 4) is modified in the following way:

```
p0 = heelStart;
p1 = p0 +      (heelEnd - heelStart)/4;
p2 = p1 + rot1*(heelEnd - heelStart)/2;
p3 = p2 + rot2*(heelEnd - heelStart)/4;
```

The rotations *rot1* and *rot2* are a product of spherically linearly interpolated quaternions, standing for the rotation during the current segment, and the inverse of the current rotation. This conserves the length of the path the heel takes and rotates it along a curved path as shown in Figure 6.

Phases where ground contacts occur are explicitly not adjusted for turning. If there was rotation added in those parts, the ground contacts would need to rotate in place, which produces unrealistic behaviour because, when accounting for physics, this would create unnecessary friction. Instead these rotations are absorbed by the joints.

3.8. Uneven Terrain

To enable walking on uneven terrain, we introduced a new ground model, implemented as part of the bonus in assignment one [31]. The height of the terrain gets computed by casting a ray down from the sky and checking for the y-coordinate of the hit. This height offset was then added to all the target positions that are in contact with the ground.

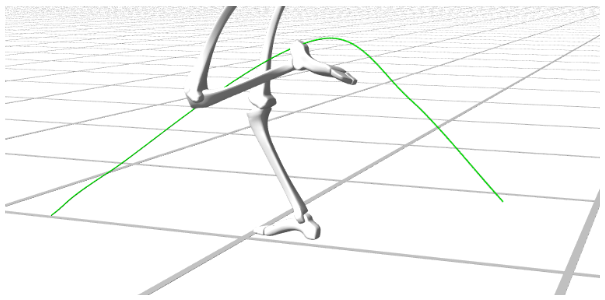
Figure 7. Visualization of a trajectory while running.

3.9. Running

Creating a realistic running animation requires a number of modifications from the walking controller.

The first issue lies in more articulated poses occurring while running than during walking. This leads to the simplified end effector placement explained in Section 3.1 failing. To properly track the heel and toe during running their end effectors had to be placed on their actual rigid bodies. In order to prevent unnatural poses during the final part of the stance phase, the toe joint limits then have to be dynamically updated during the animation. During the contact phase they are constrained to be parallel to the ground. These limits are lifted towards the end of the stance phase where they return to their default values. Let joint j be a toe joint, with default joint limits $\theta_j^{\text{min}}, \theta_j^{\text{max}}$. If θ_j^* is the required toe angle for parallel toe-ground contact then the toe limits are computed according to

$$\begin{aligned}\tilde{\theta}_j^{\text{min}} &= (1-\tau)(\theta_j^* - 10^{-6}) + \tau\theta_j^{\text{min}} \\ \tilde{\theta}_j^{\text{max}} &= (1-\tau)(\theta_j^* + 10^{-6}) + \tau\theta_j^{\text{max}}\end{aligned} \tag{8}$$

where $\tau \in [0, 1]$ is defined to be 0 at 90% of the stance phase, 1 at the start of the swing phase and increases linearly between these two points.

The swing trajectory is computed similarly to the swing for walking. The first control point has a higher target height with a vertical offset of *0.5*. During running the leg has a much faster swing and due to the added momentum it is typical that the foot first ”overshoots” the next contact position. To achieve this, an additional offset of *0.3* in the running direction is added to the second control point. Finally one additional control point is added between the first and second control point at time *t = 0.35* to further improve the realism. This point is linearly interpolated between the first and second control point and has an added vertical offset of *0.05*. The resulting trajectory can be seen in Figure 7.

The base movement behaves similarly to that of the walking animation. The only difference is a changed vertical offset that is based on measured running trajectories [22]. The arm movement was also adjusted. Most runners keep their elbows close to a 90° angle. However, keeping

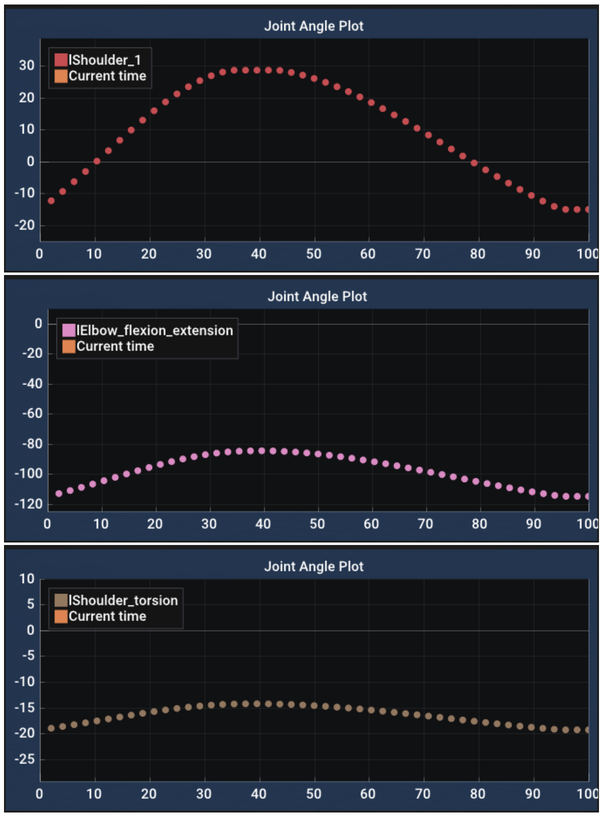

Figure 8. Joint angle trajectories of the left shoulder, elbow and the left shoulder torsion during one gait cycle for running.

the elbow angle fixed to that number looks too stiff, hence we added some offset to create a more natural motion. In addition, arm torsion is included so that the arms are slightly pulled in front of the chest during the swing. The joint angle trajectories for running can be seen in Figure 8. The amplitude of the spine rotation was also increased to 7°. Finally, the lengths of the different phases during the gait cycle were adjusted, as the swing phase takes up a larger percentage of a full cycle, and the length of a cycle itself was reduced to match the higher stepping cadence during running.

4. Results

Unfortunately it is difficult to objectively quantify and assess the quality and naturalness of procedurally generated motion. Reinforcement learning based approaches can be assessed by their accumulated rewards and motion matching techniques can be judged by how close the resulting motion is to the used motion capture data. For a fully procedural approach such an evaluation based on a scalar metric is not straightforward. It would be possible to compare the joint angle trajectories to measured data, however due to relatively large, personality-based variations in those trajectories it is not clear what exact trajectory should be used as a reference. While a mean trajectory seems sensible at first we found that it is not actually a good measure as animations purely based on this mean trajectory look uncanny. Hence, we did not include any scalar performance measures and instead resort to just showing the generated motion.

4.1. Animation Quality

Figure 9 and Figure 10 show a few snapshots during one full gait cycle of the walking and running animations. Our method is able to reproduce distinctive features of the human gait. The gait starts with firm ground contact of the foot that transitions into the Toe-Off followed by the swing phase and is concluded by the heelstrike. Note that during walking there is a phase of double support since the stance phases of the two different legs overlap (fourth image in Figure 9). For running this is replaced by a phase without any ground contact as the swing phase makes up a larger part of the whole gait cycle leading to overlapping swing phases for both legs (fourth image in Figure 10). For animation videos we refer to our GitHub repository [6].

4.1.1 Performance

To benchmark our performance we ran the simulation in different scenarios with a frame rate capped at 1024 fps. We compared the performance when rendering the full scene, i.e. the full body model, ground, background + debug info, and when rendering only the skeleton. We found that when rendering the full scene, performance is bottlenecked by the rendering of the floor and not the locomotion controller. When not rendering the the full scene performance is significantly increased and the high rate at which the controller is able to perform can be seen. However, running and circular movement decrease the performance. This is due to the IK solver needing more time to find the joint configurations for the more articulated motion. The tests were performed on a consumer grade laptop. These tests show that our planning and control pipeline is more than capable of running at interactive rates and would hence also be suited to be applied to larger crowds. The exact results can be found in Table (1).

5. Conclusion

In this work we presented biomechanically grounded strategies to synthesize realistic motion for digital humans. We presented conceptually simple planning of foot trajectories that can replicate the key features of human leg motion. In addition we presented strategies able to synthesize realistic supporting movement of the arms and upper body. Finally we extended these principles to allow walking in curved lines, on uneven terrain and introduced a different type of locomotion, namely running.

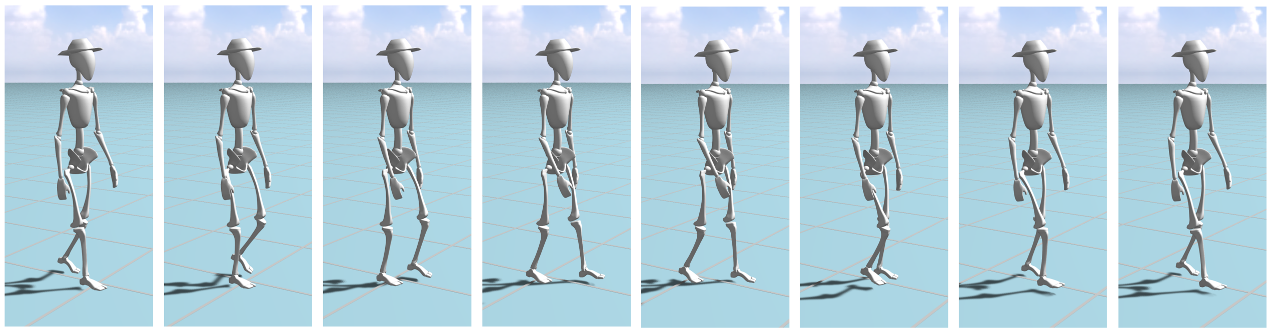

Figure 9. Visualization of one gait cycle while walking.

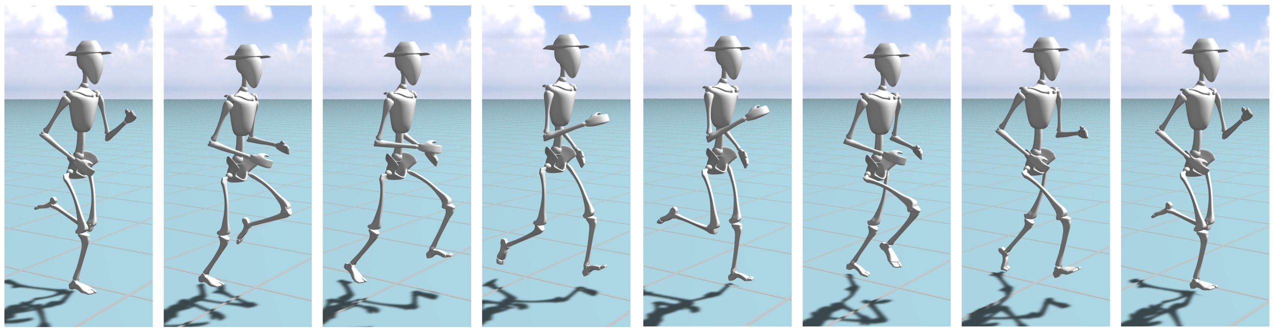

Figure 10. Visualization of one gait cycle while running.

Scenario	Full Rendering	Rendering skeleton
Walking in a line	195 FPS	746 FPS
Walking in a circle	188 FPS	298 FPS
Running in a line	159 FPS	242 FPS
Running in a circle	145 FPS	221 FPS

Table 1. Performance Benchmark on a Laptop with a Ryzen 7 5800H CPU (@3.2 GHz)

However, there are some limitations to our work. Firstly, physics is not taken into account. Hence, without modifications our method will not work in applications where the laws of physics have to be considered. One possible solution would be to keep the planning strategies as presented and use the target joint angles as targets for a simple Proportional-Derivative controller. While our synthesized motion is able to handle some forms of uneven terrain there is no explicit object avoidance during the path planning. It would be good to check for possible collisions during planning and adjust the trajectories to avoid those obstacles. Furthermore, the height of the base would need to take into account the terrain change as well. Our current implementation strictly differentiates between walking and running. Having a method that can smoothly transition from one type of locomotion to the other based on the target velocity would be desirable. Finally, it might also be useful to walk backwards or sideways without turning.

6. Contributions of team members

Some of the mentioned features were tested but not included in the final version. The exact work distribution is as follows:

- Joshua: gait cycle; contact planning; contact constraints; swing trajectories; IK solver; joint based arm/upper body control; modifications for running
- Marie: base trajectory; pelvis movement; mesh improvements
- Rafael: contact planning; contact constraints; swing trajectories; turning motion; uneven terrain; background render, joint angle plotting
- Yucheng: IK based arm/upper body control; pelvis movement; improvements to arm-leg synchronization

References

[1] Simon Clavet. Motion matching and the road to next-gen animation, 2016. 2

[2] Sameer Agarwal, Keir Mierle, and The Ceres Solver Team. Ceres Solver, 3 2022. 3

[3] Shailen Agrawal, Shuo Shen, and Michiel van de Panne. Diverse motion variations for physics-based character animation. *Symposium on Computer Animation*, 2013. 2

[4] Christopher J. Arellano and Rodger Kram. The metabolic cost of human running: is swinging the arms worth it? *Journal of Experimental Biology*, 217(14):2456–2461, 07 2014. 5

[5] De Rossi LM Diaferia L Meregalli S Gatti A. Armeni P, Polat I. Digital twins in healthcare: Is it the beginning of a new era of evidence-based medicine? a critical review. 2022. 1

[6] aurandj, mjaillot, steinraf, yucwang. Procedural locomotion engine for digital humans. `https://github.com/Joshua31415/procedural-locomotion`, 2023. 1, 7

[7] Kevin Bergamin, Simon Clavet, Daniel Holden, and James Richard Forbes. Drecon: Data-driven responsive control of physics-based characters. *ACM Trans. Graph.*, 38(6), nov 2019. 2

[8] Sjoerd M. Bruijn, Onno G. Meijer, Peter J. Beek, and Jaap H. van Dieën. The effects of arm swing on human gait stability. *Journal of Experimental Biology*, 213(23):3945–3952, 12 2010. 5

[9] Edwin Catmull and Raphael Rom. A class of local interpolating splines. In ROBERT E. BARNHILL and RICHARD F. RIESENFELD, editors, *Computer Aided Geometric Design*, pages 317–326. Academic Press, 1974. 4

[10] Stelian Coros, Philippe Beaudoin, and Michiel Panne. Robust task-based control policies for physics-based characters. *ACM Trans. Graph.*, 28, 12 2009. 2

[11] Stelian Coros, Philippe Beaudoin, and Michiel van de Panne. Generalized biped walking control. *ACM Transctions on Graphics*, 29(4):Article 130, 2010. 1, 5

[12] Simeon Yuen Vladimir Mastilovic Danielle Costa, Doug Roble. Progress toward real-time, photo-realistic digital humans: Nvidia on-demand. `https://www.nvidia.com/en-us/on-demand/session/siggraph2019-sig908/`. 1

[13] Animesh Hazari, Arun G. Maiya, and Taral V. Nagda. *Kinematics and Kinetics of Gait*, pages 181–196. Springer Singapore, Singapore, 2021. 1, 2

[14] Babak Hejrati, Sam Chesebrough, K. Bo Foreman, Jake J. Abbott, and Andrew S. Merryweather. Comprehensive quantitative investigation of arm swing during walking at various speed and surface slope conditions. *Human Movement Science*, 49:104–115, 2016. 5

[15] Daniel Holden, Oussama Kanoun, Maksym Perepichka, and Tiberiu Popa. Learned motion matching. *ACM Trans. Graph.*, 39(4), aug 2020. 2

[16] John Hughes and Nicole Jacobs. Normal human locomotion. *Prosthetics and Orthotics International*, 3:12 – 4, 1979. 2

[17] Yongjoon Lee, Kevin Wampler, Gilbert Bernstein, Jovan Popović, and Zoran Popović. Motion fields for interactive character locomotion. *ACM Trans. Graph.*, 29(6), dec 2010. 1

[18] Sergey Levine, Jack M. Wang, Alexis Haraux, Zoran Popović, and Vladlen Koltun. Continuous character control with low-dimensional embeddings. *ACM Transactions on Graphics*, 31(4):28, 2012. 1, 2

[19] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning, 2019. 2

[20] Libin Liu and Jessica Hodgins. Learning to schedule control fragments for physics-based characters using deep q-learning. *ACM Transactions on Graphics*, 36(3), 2017. 2

[21] Masahiro Mori, Karl F. MacDorman, and Norri Kageki. The uncanny valley [from the field]. *IEEE Robotics Automation Magazine*, 19(2):98–100, 2012. 1

[22] Christopher Napier, Xianta Jiang, Christopher L. MacLean, Carlo Menon, and Michael A. Hunt. The use of a single sacral marker method to approximate the centre of mass trajectory during treadmill running. *Journal of Biomechanics*, 108:109886, 2020. 6

[23] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. DeepMimic. *ACM Transactions on Graphics*, 37(4):1–14, jul 2018. 1, 2

[24] Xue Bin Peng, Glen Berseth, and Michiel van de Panne. Dynamic terrain traversal skills using reinforcement learning. *ACM Trans. Graph.*, 34(4):80:1–80:11, July 2015. 2

[25] Xue Bin Peng, Glen Berseth, and Michiel van de Panne. Terrain-adaptive locomotion skills using deep reinforcement learning. *ACM Trans. Graph.*, 35(4), jul 2016. 2

[26] Walter Pirker and Regina Katzenschlager. Gait disorders in adults and the elderly. *Wiener klinische Wochenschrift*, 129(3):81–95, Feb 2017. 1, 2

[27] Herman Pontzer, 4th Holloway, John H., David A. Raichlen, and Daniel E. Lieberman. Control and function of arm swing in human walking and running. *Journal of Experimental Biology*, 212(4):523–534, 02 2009. 5

[28] Alla Safonova and Jessica K. Hodgins. Construction and optimal search of interpolated motion graphs. *ACM Trans. Graph.*, 26(3):106–es, jul 2007. 1

[29] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization, 2017. 2

[30] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. 2

[31] Rafael Steiner. Digital humans assignment 1, bonus. `https://github.com/Digital-Humans-23/a1-steinraf/tree/bonus`, 2023. 6

[32] Jun Wan, HongTao Wu, Rui Ma, and Liang'an Zhang. A study on avoiding joint limits for inverse kinematics of redundant manipulators using improved clamping weighted least-norm method. *Journal of Mechanical Science and Technology*, 32:1367–1378, 03 2018. 3

[33] Jack M. Wang, Samuel R. Hamner, Scott L. Delp, and Vladlen Koltun. Optimizing locomotion controllers using biologically-based actuators and objectives. *ACM Trans. Graph.*, 31(4), jul 2012. 2

[34] Yuting Ye and C. Karen Liu. Optimal feedback control for character animation using an abstract model. *ACM Trans. Graph.*, 29(4), jul 2010. 1

[35] Andrew K. Yegian, Yanish Tucker, Stephen Gillinov, and Daniel E. Lieberman. Straight arm walking, bent arm running: gait-specific elbow angles. *Journal of Experimental Biology*, 222(13), 07 2019. jeb197228. 5

[36] Wenmin Zhu, Xiumin Fan, and Yanxin Zhang. Applications and research trends of digital human models in the manufacturing industry. *Virtual Reality Intelligent Hardware*, 1(6):558–579, 2019. 1