PlaMo: Plan and Move in Rich 3D Physical Environments

ASSAF HALLAK* and GAL DALAL*, NVIDIA Research, Israel

CHEN TESSLER, NVIDIA Research, Israel

KELLY GUO, NVIDIA Research, USA

SHIE MANNOR, NVIDIA Research, Israel
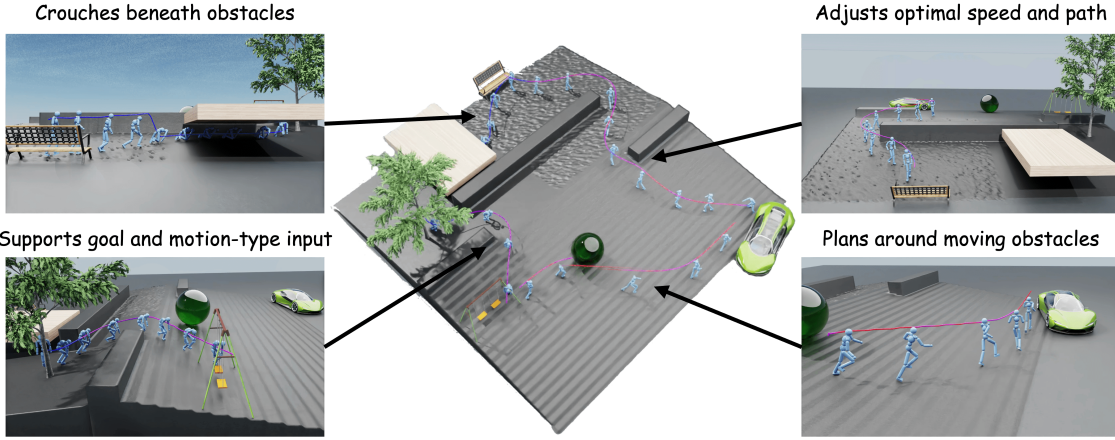
GAL CHECHIK, NVIDIA Research, Israel

Fig. 1. **Path planning and motion control** *PlaMo* animates a humanoid in a complex 3D scene. The input consists of a physically simulated 3D scene and a series of text instructions describing high-level navigation landmarks and locomotion type ("Crouch-walk from the tree to the swing"). The scene may contain diverse terrain (gravel, stairs), 3D obstacles, and dynamic obstacles (here, a green ball). The output is a sequence of motor actuations controlling a humanoid character. *PlaMo* produces a planned path together with a head-height and speed profile that matches the textual guidance (locomotion type) and the constraints of the movement controller. A. The controller adapts to the 3D environment, producing a crawl locomotion under the obstacle. B. Supports various motion types. C. continuously re-planning the path allows to avoid moving obstacles. D. Locomotion speed is adjusted depending on terrain and obstacles.

Controlling humanoids in complex physically simulated worlds is a long-standing challenge with numerous applications in gaming, simulation, and visual content creation. In our setup, given a rich and complex 3D scene, the user provides a list of instructions composed of target locations and locomotion types. To solve this task we present PlaMo, a scene-aware path planner and a robust physics-based controller. The path planner produces a sequence of motion paths, considering the various limitations the scene imposes on the motion, such as location, height, and speed. Complementing the planner, our control policy generates rich and realistic physical motion adhering to the plan. We demonstrate how the combination of both modules enables traversing complex landscapes in diverse forms while responding to real-time changes in the environment.

Video: https://youtu.be/wWlqSQlRZ9M.

---

*Both authors contributed equally to this research.

Authors' Contact Information: Assaf Hallak; Gal Dalal*, NVIDIA Research, Israel, ahallak@nvidia.com,gdalal@nvidia.com; Chen Tessler, NVIDIA Research, Israel; Kelly Guo, NVIDIA Research, USA; Shie Mannor, NVIDIA Research, Israel; Gal Chechik, NVIDIA Research, Israel.

Physics-consistent animation of human characters is a challenging task with numerous applications, from automating game design through simulating factories and urban areas to training autonomous vehicles in pedestrian scenarios. In some cases, one may want to control the animation using text instructions that capture high-level goals, like "Run to the road once the car passes by" or "Hide behind the large tree". These tasks involve two types of control problems: First, planning a sequence of motions for reaching the goal with a natural behavior; Then, controlling the actual motion of the character to generate a natural-looking movement.

Previous studies in physics-based animation have usually focused on one of these two challenges, often addressing a simplified setup. For planning, Peng et al. [2017] developed a method for long-range planning of a bipedal character, and [Rempe et al. 2023] generates a distribution of trajectories based on user-defined characteristics. Other studies focused on movement control: mimicking [Peng et al. 2018], interleaving motion types [Tessler et al. 2023], interactions with objects [Wang et al. 2023] or people [Zhang et al. 2023a] and crowd simulation [Rempe et al. 2023] (see related method section).

However, the two tasks are interdependent. Planning which motions to take depends on the physical characteristics of each motion and the character. Then, executing the motion depends on the sequence and longer-horizon goal. Importantly, this interdependence becomes stronger when environments become more complex. Handling 3D obstacles, uneven terrain, and dynamic objects may require the planner to override user instruction and revert to motions that are feasible in the specific environment. In this work, we address the problem of training physics-based motion models for humanoids that can handle long-horizon planning and motion control in intricate environments.

Our approach *PlaMo*, for *Plan and Move*, combines long-horizon path planning with a novel matching motion controller. This fusion creates an agent that responds to semantic instructions from users but can adapt the plan and motion for navigating naturally through various terrains and obstacles.

When designing a path planner for this task, we need to consider several aspects. First, the planner must only produce plans that can be executed by the character's physics-based motion controller. More specifically, the character typically has a range of possible motions, such as walking, running, or crouching under an obstacle. The planner should devise plans that are within the feasibility envelope of the motion controller. Second, the capabilities of the humanoid character depend on the environment. For example, it is harder to run uphill than downhill. The planner therefore must be aware of the slope and obstacles. A similar problem occurs when turning around obstacles. Running too fast makes it harder to control a character, so the planner must match the character's velocity to the curvature of the path.

To address these challenges, we develop a planner inspired by state-of-the-art planners in the fields of robotics and autonomous driving [Alcántara et al. 2021; Hsu and Gau 2020; Wang et al. 2020b]. These techniques have proved very useful in hard real-world problems and we show here how they can be applied in physics-based character animation. Our planner operates in three stages. First, it constructs a low-resolution path using the A^* algorithm. Second, it creates a high-resolution height trajectory that determines the position of a character's head along the path, taking into account 3D obstacles. Third, it sets the locomotion velocity along the path, based on the slope and curvature of the path and

Preprint

on the feasible movement envelope of the character, which we learn offline. This is achieved by solving a quadratic program that finds the optimal speed along the path [Alcántara et al. 2021]. At the end of planning, it produces a sequence of head positions and velocity along the path and sends that to the motion controller. As far as we know, this is the first long-horizon planner that is aware of a complex 3D environment and character motion envelope.

Guided by the planned path, the motion controller actuates the character's joints at every time step. We condition the controller on the pose and velocity of the humanoid, surrounding terrain, and near-term path waypoints. To train the motion controller, we designed a reward function with two main components: path-following and style. The path-following component rewards the agent for successfully reaching target 3D waypoints at designated times. Our reward design considers 3D path tracking, coupled with additional terms for improving orientation and posture. For style, we use a discriminative objective [Peng et al. 2021, AMP] encouraging natural movement. This controller is trained on random paths that take into account the correlation between height and speed, for example, crawling is typically performed slowly.

In summary, this paper makes the following novel contributions: (1) We introduce a methodology for combining path planning and motion control within a detailed physical simulation, which we call *PlaMo*. (2) We present an efficient method for planning long-horizon paths for animated humanoid characters, producing a feasible and safe path across uneven terrain around static, dynamic, and three-dimensional obstacles. (3) We develop a robust motion controller trained through reinforcement learning. Training to imitate human motions, the controller produces motion guided by a sequence of 3D head positions and velocities.

1 Related Work

Physics-based animation. Prior work shows that physics simulation can be utilized for learning scene-aware controllers [Luo et al. 2023a, 2022; Won et al. 2022; Yuan and Kitani 2019, 2020]. These controllers are robust to changes in the scene like irregular terrain, obstacles, and adversarial perturbations [Lee et al. 2023; Luo et al. 2023b; Wang et al. 2020a]. Typically, prior work focused either on complex motions in simple scenes [Juravsky et al. 2022; Peng et al. 2022; Tessler et al. 2023], or simple motions within complex scenes [Hassan et al. 2023; Xie et al. 2020]. Closest to our work is TRACE and PACE [Rempe et al. 2023], which focuses on pedestrian locomotion with a diffusion-based planner. To generate paths, the planning module TRACE is trained on pedestrian data, typically upright walking motions. In contrast, we focus on long-term 3D-aware navigation and locomotion. *PlaMo* combines a short-sighted terrain-aware locomotion controller with a long-term and adaptive scene-aware path generator. The path generator plans a 3D trajectory. This plan takes into account the scene and the capabilities of the motion controller. It determines the speed and height to avoid both static and dynamic obstacles. This plan is provided to the controller. The controller then tracks the requested path along rich and complex terrains.

Scene-aware animation. Most prior work has used kinematic animated characters to tackle a wide range of problems such as object interaction [Li et al. 2023; Starke et al. 2019], long-term motion generation [Ling et al. 2020] and storytelling [Qing et al. 2023]. However, as physics-based animation is less forgiving, prior works that use simulation have typically focused on solving specific object-interaction tasks [Hassan et al. 2023], or simple locomotion (walking or running) with short-term planning across complex terrains [Rempe et al. 2023]. We tackle the problem of long-term motion generation across complex scenes using physics-based control.

Path planning. There are numerous works in the world of planning motion in different levels of hierarchy. For high-level planning, the A* algorithm and its variations [Bell 2009; Duchoň et al. 2014; Yao et al. 2010] use dynamic programming to produce a discrete path that considers the cost and connectivity of moving across the grid. Alternatively,

navigation meshes [Kallmann 2010; Kallmann and Kapadia 2014; Van Toll et al. 2012] use the geometric structure of the obstacle to find the shortest path. Finer control models such as Model Predictive Control (MPC; [Holkar and Waghmare 2010; Qin and Badgwell 1997]) construct an optimization problem by considering the dynamics of all objects and the physical constraints of the agent while using a lookahead of a receding horizon, these models usually require an explicit knowledge on the dynamics of the agent itself. In cases where the dynamics in the scene are unknown or complicated, learning methods can be employed instead such as RL for a given simulator [Chen et al. 2022; Michels et al. 2005; Zhang et al. 2023b] or diffusion models when there is access to relevant data [Barquero et al. 2023; Karunratanakul et al. 2023]. In this context, our works are most related to recent approaches for planning in autonomous driving, extending A* with additional smoothing and obstacle avoidance modules to form feasible trajectories [Maw et al. 2020; Yeong-Ho et al. 2020; Zhong et al. 2020].

2 Preliminaries

We begin with a formal introduction of the reinforcement learning (RL) setup for training our motion controller, together with the mathematical representation of our simulated scene.

2.1 Reinforcement Learning

We train a goal-conditioned motion controller which we model as an RL [Sutton and Barto 2018] problem. In RL, an agent interacts with an environment according to a policy π. At each step t, the agent observes a state s_t and samples an action a_t from the policy $a_t \sim \pi(a_t|s_t)$. The environment then transitions to the next state s_{t+1} based on the transition probability $p(s_{t+1}|s_t, a_t)$. The goal is to maximize the discounted cumulative reward, defined as

$$J = \mathbb{E}_{p(\sigma|\pi)}\left[\sum_{t=0}^{T} \gamma^t r_t \middle| s_0 = s\right],$$

where $p(\sigma|\pi) = p(s_0)\Pi_{t=0}^{T-1}p(s_{t+1}|s_t, a_t)\pi(a_t|s_t)$ is the likelihood of a trajectory $\sigma = (s_0, a_0, r_0, \ldots, s_{T-1}, a_{T-1}, r_{T-1}, s_T)$, and $\gamma \in [0, 1)$ is a discount factor that determines the effective horizon of the policy.

To find the optimal policy π^* that maximizes the discounted cumulative reward, the popular policy gradient paradigm seeks to directly optimize the policy using the utility of complete trajectories. Perhaps the most widely used algorithm in this context is PPO [Schulman et al. 2017], which gradually improves the policy while maintaining stability without drifting too far from it.

2.2 3D Scenes

In this work our characters navigate across rich 3D scenes. We describe the simulated scene as:

$$Scene = \{h, O_{\text{static}}, O_{\text{dynamic}}, O_{\text{top}}, \mathcal{L}\}. \tag{1}$$

The 5 components of the scene are as follows:

(1) A terrain given as a height map $h(x, y) \in \mathbb{R}$, specifying the height for each location in the grid.
(2) A set of non-passable static obstacles O_{static}.
(3) A set of dynamic obstacles and their properties O_{dynamic}.
(4) A set of height-limiting ("top") obstacles requiring the agent to crouch or crawl when traversing through them O_{top}.
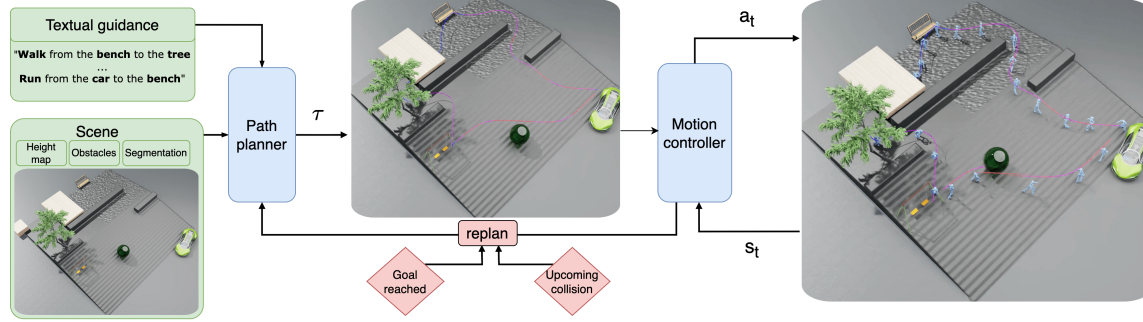(5) A set of landmarks $\mathcal{L} = \{L_i\}_{i=0}^{l}$.

Fig. 2. **Method overview.** A simulated scene is provided to the path planner, together with a series of textual instructions, requesting a humanoid to reach landmarks in the scene using various locomotion types. The high-level planner computes a path that is fed to a reinforcement-learning-based low-level motion controller, which controls a humanoid to follow the path using the request locomotion type.

The locations and speeds of the dynamic objects are updated in each simulation step, based on their provided properties. The goal in these scenes is to plan a path between a series of landmarks $L_1, \ldots, L_k$ and successfully execute the plan.

3 Method

Our method, *PlaMo*, consists of two main components (Figure 2):

High-level path planner. The planner receives two inputs: a physically simulated 3D scene and a templated text instruction describing a sequence of (locomotion type, start point, target point). The planner then outputs a path τ consisting of velocity and 3D head positions at each time-step. The path is optimized to lead the agent through a smooth, physically feasible path to reach the goal while avoiding obstacles. It is then sent to the motion controller.

Low-level locomotion controller. This controller receives as input the height map h of the local terrain and a desired path τ. It outputs a sequence of motor actions for the humanoid joints, resulting in the humanoid following the desired path. The locomotion controller is trained on real-world motion captures using RL to generate realistic human-like behavior.

Together, at each time-step of the simulation, the path planner translates the current humanoid location and target landmark to a local path that the controller aims to follow. The controller then outputs actions on the joints to the simulator, which then advances to the next simulation step.
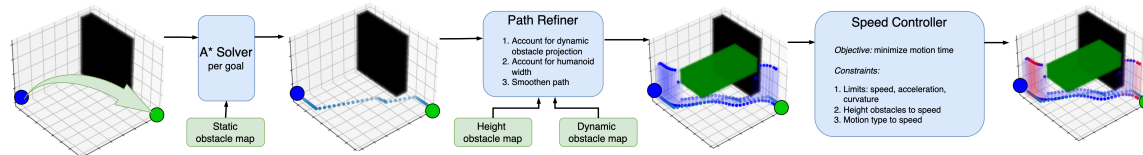

Fig. 3. The three stages of our dynamic path planner: (i) A^* solver, (ii) path refiner, and (iii) speed controller.

3.1 Dynamic Path Planner

We present a comprehensive module for path planning, structured into distinct phases. The full planning flow is visualized in Figure 3.

Textual guidance parser. The input to the planner is a 3D simulated scene *Scene* and a series of k text instructions in the form "Run from the tree to the lake". These instructions refer to known landmarks in the scene where we assign each landmark to its corresponding set of (x, y) coordinates. We then translate each instruction to a triplet of (source landmark, target landmark, locomotion type). Together, they form a sequence of instructions that go through the series of landmarks $L_1, \ldots, L_k$ connected by paths. We observe that two main characteristics that differentiate between desired forms of locomotion are speed and height. As such, each path is represented as a series of 3D positions. We determine velocities by controlling the distance between the points on the path. These paths are fed to the motion controller.

Our planner is inspired by state-of-the-art robotics and autonomous driving techniques [Alcántara et al. 2021; Hsu and Gau 2020; Wang et al. 2020b]. It operates in three stages: low-resolution path construction using the A* algorithm, high-resolution height trajectory creation considering 3D obstacles, and locomotion velocity setting based on path slope and curvature. It ultimately produces a sequence of head positions and velocities for the motion controller, being the first long-horizon planner aware of complex 3D environments and character motion envelopes.

A* with awareness to slope. We form a 2D grid on the terrain level, connecting every point to its eight neighbors. We then remove connections based on walls, top obstacles too low to bypass, and infeasible slopes. The remaining connections are assigned a weight based on the distance and height difference, representing how difficult and time-consuming it is for the agent to cross it. Then, for every landmark L_i, we mark all its assigned (x, y) coordinates as goals for A*. Finally, the A* algorithm [Bell 2009] constructs a shortest-path solution to the nearest coordinate of the landmark. To support long paths across large areas, we implement A* efficiently using KD-trees [Zhou et al. 2008] and perform it once.

Path refiner. The path created at the previous A* phase is coarse and impervious to dynamic obstacles and height constraints. We refine it to achieve smoother paths that avoid all obstacles. Thus, these paths are easier for the low-level controller to follow, increasing its success rate. For *dynamic obstacles*, we detect potential collisions by estimating the movement pattern of the obstacle and comparing it with the last designated path τ. If a collision is likely to occur within a 1.5 $[s]$ time frame, we replan the path, with the dynamic obstacle set at its future location as an added constraint. Finally, the path refiner adjusts the trajectory for smoothness and obstacle avoidance. The path refiner then sets the desired head height (z-axis) values that guide the character beneath top obstacles.

Motion-aware speed controller. The path created in the previous phases contains only spatial information and lacks consideration of the humanoid character's performance envelope. The third step assigns a velocity vector to each frame in the path, consistent with the requested motion type (walk, run, crouch, crawl) and the terrain. Specifically, a quadratic program (QP) is solved to minimize travel time along the refined path while avoiding infeasible areas. The QP is constrained by valid speed ranges for each locomotion type and by bounds on longitudinal and lateral accelerations that the character can achieve. The speed controller accounts for path curvature, adapting longitudinal speed based on lateral acceleration and adjusting the speed according to elevation and slope. This results in a refined speed profile that ensures the character slows down before turns and around obstacles.

Dependence between control and planning: The effectiveness of our method hinges on the strong interdependence between the control and planning phases. The A* algorithm incorporates height differences into its cost function, considering the difficulty and time required for the character to traverse different elevations. This notion can be easily extended to different agents with different capabilities; for example, an accomplished humanoid climber will choose a steeper, shorter path than an agent having difficulties with high slopes. The tight coupling continues through the path refinement and speed control phases, where adjustments for slope, curvature, and terrain are directly informed by the
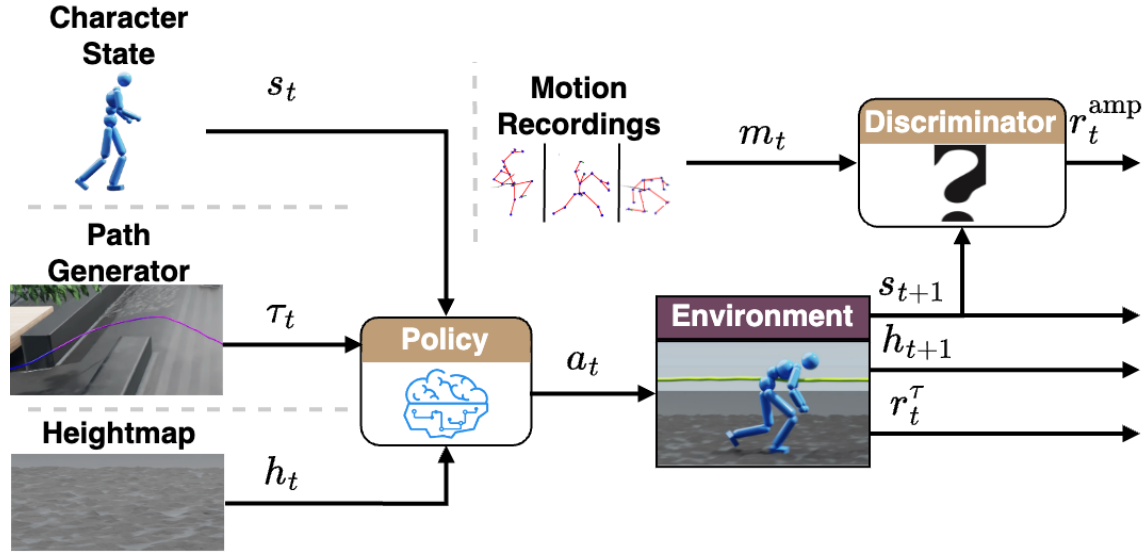
Fig. 4. **Locomotion controller module.** The locomotion policy observes the character state s_t, the height map of the terrain h, and the requested path τ. During training, the trajectories are randomly sampled. In inference, the trajectories consider the terrain characteristics, such as obstacles. After simulating the predicted action the resulting state is used for providing the style reward r^{amp} and for the path following reward r^{τ}.

character's performance limits. This ensures that the final path is not only theoretically optimal, but also practically executable, maintaining the balance and stability of the character in real-world conditions.

3.2 Locomotion controller

Plans produced by the high-level path planner are provided to a low-level locomotion controller. This controller, illustrated in Figure 4, is tasked with executing the provided plan using human-like physical motion. The provided plans control the height, location, and speed at which the character should move. This is performed across a wide range of irregular terrains. To achieve this we train our locomotion controller using reinforcement learning. We provide a reward with two components: one for following the path and the other for producing realistic motion. In this paper, we accommodate four forms of locomotion: running, walking, crouch walking, and crawling. It is interesting to note that the planner does not explicitly guide a specific locomotion type, like running or crouching; but since the motion controller is tasked with creating a natural-looking movement, the right type of locomotion naturally emerges for every speed and head height.

State. At each step the controller observes the current pose of the character s_t, the target trajectory τ_t, and the surrounding heightmap h_t. The trajectory is provided as 5 seconds into the future with ten samples, one every 0.5 seconds. The height map allows the controller to be aware of its surroundings, ensuring motions are robust to varying scenes. We represent the height map as a rectangular grid rotated to the character's root orientation. It consists of 64 equidistant samples spanning 0.8 $[m]$ on each axis.

Reward. The goal of the controller is to track a target path τ, generated by the planner (Section 3.1). We identify that speed and height are the two main characteristics for controlling our desired forms of locomotion (run, walk,

crouch-walk, crawl). As such, the path is constructed in 3D. At each time step t, the point τ_t represents the 3D coordinates for the head. By controlling the distance between the points, we determine the requested speed, and changing the Z-axis provides control over the desired character's height.

Due to the importance of height control, we split the reward into four components. The first considers the x-y displacement of the character. This guides the path the character should follow, in addition to speed. The second component considers the height of the head. The third considers the head's direction (pitch and yaw), aligning it with the path. Since all path rewards target the head, to produce more realistic movement when following lower heights we add the reward term: $r_t^{\text{body}} = c_{\text{body}}(z_{\text{head}} - \max_{\text{body}} z)$ with $c_{\text{body}} = 0.1$. This term encourages the head to be the highest point to prevent a hunchback posture when crouching and crawling.

$$r_t^\tau = e^{-c_{\text{pos}}||\tau_t^{x,y} - s_t^{x,y}||^2} + e^{-c_{\text{height}}||\tau_t^z - s_t^z||^2} + e^{-c_{\text{dir}}||\tau_t^{y,p} - s_t^{y,p}||^2} + r_t^{\text{body}},$$

where $c_{\text{pos}} = 2$, $c_{\text{height}} = 10$ and $c_{\text{dir}} = 20$ are fixed.

Complementing the task reward, we leverage AMP [Peng et al. 2021] for motion stylization. AMP trains a discriminator to differentiate between reference recorded motions m_t and those simulated by the motion controller s_t. The resulting reward combines the tracking objective with the discriminative reward: $r_t = r_t^{\text{amp}} + r_t^\tau$.

Reference motions. The adversarial reward in AMP is maximized when the simulated pose distribution matches the data. Naively providing a diverse dataset of human motions may result in unwanted behaviors. For example, back-flips and break-dancing are distinctly different from locomotion. To ensure that the motion distribution matches the motions needed to solve the task, we hand-picked 75 distinct motions from the AMASS dataset [Mahmood et al. 2019], covering the four locomotion forms performed in various styles. An appropriate ablation study is provided in Section 4.

Random path generation. During training, the path generator samples random paths for the character to follow. If the task demands unlikely motion given the data, we observe a degradation in both motion quality and tracking performance. For example, crawling at $v = 5m/s$ is unrealistically fast and will not appear in our motion capture data. Thus, the task and discriminative rewards will contradict each other, impairing the training process. To provide a realistic path, the generator first samples the requested heights along the path, ranging from $0.4\,[m]$ (crawling) to $1.47\,[m]$ (the simulated humanoid's height), with smooth transitions in between. Then, the speed is sampled uniformly between 0 and $v_t^{\max}$, where:

$$v_t^{\max} = \min\left(1 + 4 \cdot (z_t - z^{\min})/(1.2 - z^{\min}),\ V_{\max}\right),$$

for a minimal height $z^{\min} = 0.4\,[m]$ and maximal speed $V_{\max} = 5\,[m/s]$. This reasults in a linearly increasing max-speed, ranging from $v_t^{\max} = 1\,[m/s]$ for crawling motions, and up to $v_t^{\max} = 5\,[m/s]$ for any height above $1.2\,[m]$.

Actions. Similar to prior work [Peng et al. 2018; Rempe et al. 2023], we opt for proportional derivative (PD) control. We represent the action distribution using a multi-dimensional Gaussian with a fixed diagonal covariance matrix.

4 Experiments

We test *PlaMo* both qualitatively and quantitatively. First, we evaluate the motion controller with several common forms of motion across a range of randomized terrains. In addition, we evaluate our design decision regarding the workflow for training the controller, using ablation experiments regarding both data curation and height-speed coupled path generation.

Then, we analyze the quality of the path planner. We focus on evaluating the paths produced by the QP solver in terms of how well they fit the capability envelope of the locomotion controller. Finally, we provide qualitative examples

both for the locomotion controller and the whole system in a complex scene with multiple landmarks. We stress that our test scenes were not observed during training. Our agent can navigate during inference in every scene as defined in (1), without requiring any further training. We provide video demonstrations and comparisons in the supplementary material.

4.1 Experiment details

We focus here on the SMPL humanoid [Loper et al. 2015] with a neutral body structure. The humanoid is simulated in IsaacGym [Makoviychuk et al. 2021], operating at 120 [Hz]. The controller is trained using PPO [Schulman et al. 2017] and takes decisions at a rate of 30 [Hz]. The motion controller is trained for one week on a single NVIDIA V100 GPU.

Training Data. We utilize motion capture data from the AMASS dataset [Mahmood et al. 2019] that contains more than 10 hours of diverse motion recordings across thousands of distinct motions. Complementing AMASS, HumanML3D [Guo et al. 2022] provides textual labels for each motion.

5 Results

We now discuss the resulting paths and motions produced by *PlaMo*.

5.1 Locomotion

We test our locomotion module on various terrains for every motion type; **example videos are given in the supplementary material**, and screen captures are shown in Figure 7. Table 2, reports the XY-displacement along the path, which shows how well the controller tracks the requested path, and the Z-displacement, which measures how well it maintains the requested height. The results show a hierarchy of complexity between tasks such that stairs are harder than rough gravel-like terrain, which is harder than locomotion across a flat and smooth sloped surface. Although the data only contained motions traversing flat, smooth floors, the controller learned to generalize to a wide range of uneven terrains and produce robust motions resembling those in the data. Also, as expected, moving at lower speeds enables the character to follow the path more precisely, thus maintaining a lower error.

Locomotion ablation. We consider the crawling on rough terrain. We focus on this task as it was shown in Table 2 as the hardest type of motion to generate. For our ablation study, we consider two design decisions – data curation, and random path generation. **Data curation:**[1] We evaluate the importance of data curation by comparing training on the full dataset (*none*), automated curation (*naive*, based on the HumanML3D [Guo et al. 2022] textual labels), and manual curation (*curated*). **Random path generation:** To showcase the importance of coupling height and speed in the randomly sampled paths, we also compare our *coupled* motion-aware path sampler that samples the requested speed along a random path according to the requested height, with a *naive* baseline of uncoupled sampling of both speed and height.

The results, shown in Table 1, emphasize the importance of these design decisions. We find that the best results are obtained by combining the manually curated dataset, alongside the coupled random path generator. By ensuring a tight alignment between the requested motions, the data representing them, and the randomly generated paths, we find an improvement of 50% in the XY-err, compared to the next best combination.

---

[1]We provide additional information in the supplementary material.

Table 1. **Locomotion ablation.** We compare various design choices for data purification and path generation during training. The analysis focuses on crawling on rough terrain, with additional results provided in the supplementary material. The results were averaged over 10^6 runs.

Data purification	Path generator	XY-err	Z-err
Ours (Curated + Coupled)		0.25	0.10
Curated	Naive	0.50	0.13
Naive	Naive	0.78	0.09
Naive	Coupled	1.16	0.17
None	Coupled	0.66	0.09
None	Naive	1.39	0.19

Table 2. **Mean path displacement for various locomotion types .** We test four motion types over four types of terrains. For each, we sample a path with constant speed and height, and compute the mean error between the position of the humanoid and the desired path. The results were averaged over 10^6 runs.

	Crawl		Crouch-walk		Walk		Run	
Terrain	XY-err [m]	Z-err[m]	XY-err [m]	Z-err [m]	XY-err[m]	Z-err [m]	XY-err[m]	Z-err[m]
Flat	0.20	0.07	0.20	0.03	0.24	0.21	0.52	0.26
Slope	0.21	0.09	0.22	0.03	0.25	0.21	0.60	0.27
Rough	0.25	0.10	0.25	0.04	0.29	0.24	0.79	0.31
Stairs	0.45	0.13	0.41	0.07	0.39	0.27	1.01	0.35

5.2 Terrain-dependent path planning

When animating a character we often want it to exhibit behaviors that align with the character's story. For example, while a young athlete may be capable of running up a hill, an old person may prefer to take a longer and less physically challenging path. Here, we show that these preferences can be encoded into our path planner. To do so, we re-weight the connectivity graph based on the slope. The distance between each two points is increased by $e^{c \cdot \text{slope}} - 1$. As illustrated in Figure 5, as a result of increasing the weight, our planner produces a path around the hill instead of directly across it.

5.3 Speed control

The third phase of our path planner uses a QP solver to optimize the speed along the path curves. On the one hand, if the speed is too high for the curvature, the character will not be able to maintain the desired route, resulting in large displacement errors and possible collisions with obstacles. On the other hand, if the speed is too low, it takes the character an unnecessarily long time to complete the route. To avoid jerks, the optimization program also considers the maximum acceleration ($a_{\max} = 0.5 \left[m/s^2 \right]$) and deceleration ($a_{\min} = -0.1 \left[m/s^2 \right]$).

We demonstrate this trade-off in the "Slalom" scene (Figure 8). This scene requires the character to take steep turns. Here, we compare constant speed versus adaptive speed. We generate both routes using our A^* solver and path refiner. However, we only refine the "adaptive speed" run using our speed controller module. Without adapting the acceleration profile to the curvature of the path, the humanoid fails to follow the path accurately and often collides with the walls.

For a quantitative evaluation of the Slalom test, we compare various constant speeds, as well as various configurations of our QP solver. The latter includes different combinations of maximal lateral and longitudinal acceleration. We summarize the results in Figures 10a and 10b. Each choice of constant speed and QP configuration corresponds to a single point in the plots and is the average of 100 runs. The constant speeds were chosen between 1 [m/s] and 3.5 [m/s] .
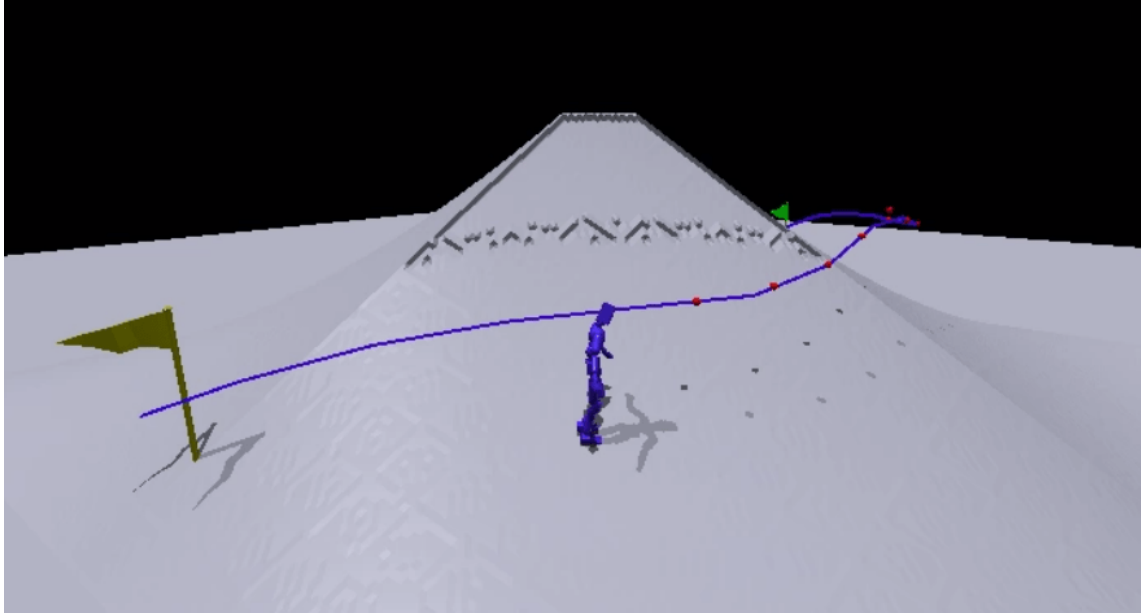Preprint

Fig. 5. **Terrain dependent path planning** The humanoid does not go through the shortest path between the two landmarks but prefers a smoother, flatter surface due to the height differences.

In Figure 10a, we report the success rate versus the average completion time. Success is defined as successfully reaching the target landmark. In Figure 10b, we show the same, but with the average 3D disposition error on the y-axis. As seen, for every QP configuration we tested, our planner enables the humanoid to traverse its path faster and without compromising on accuracy thus achieving the Pareto front.

5.4 Randomized routes in a complex randomized scenes

Finally, we showcase *PlaMo*'s ability to solve previously unseen scenes in varying forms (Figure 6). We automatically construct random scenes consisting of: **(1)** multiple forms of terrains (smooth, gravel-like, stairs and sloped), with **(2)** several types of obstacles (walls and low-ceiling obstacles), and four landmarks in each scene. *PlaMo* is given a random scene with navigation instructions to a random sequence of landmarks. Figure 6 illustrates some of these environments, showing that *PlaMo* successfully plans realistic routes. In our experiments, our agent has an 81% chance to fully solve a randomized 4-landmarks scene.

As shown in Figures 1 and 9, *PlaMo* solves this task in varying forms while adhering to the scene constraints. It slows down and starts to crawl before passing under the low ceiling (top left), speeds up during straight lines while carefully slowing during a tight curve, and finally, it dynamically adapts the path when a collision with a dynamically moving object is deemed likely[2].

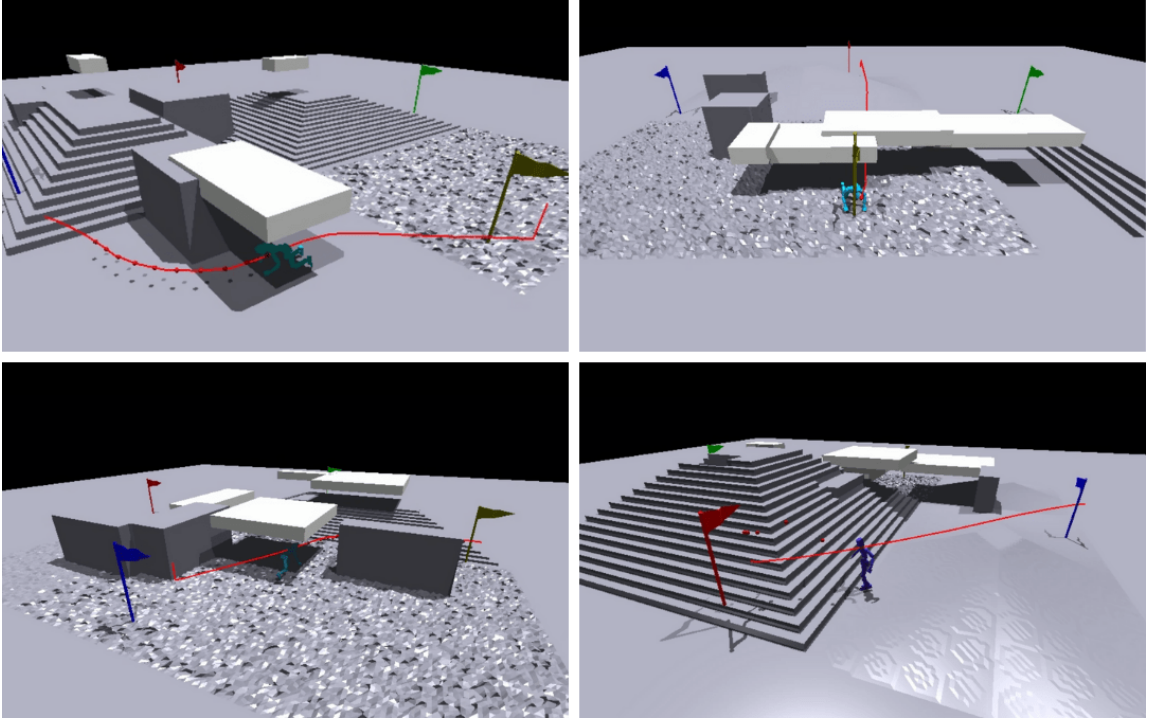[2]The dynamic re-planning is best seen in the supplementary video.

Fig. 6. **Randomized complex scenes** We randomize several terrain types, walls and top obstacles to generate a variety of scenes.

6   Discussion, limitations and future work

We described *PlaMo*, a framework for planning long paths and controlling the motion of humanoid characters in simulated environments for physics-based animation. Given a complex simulated scene and textual instructions, the system efficiently plans a path that a humanoid character can traverse using natural locomotion. The planner takes into account the character's performance envelope in uneven terrain. In addition, it takes into account both static and dynamic obstacles. Then, the trained motion controller successfully tracks these paths, producing complex, scene-aware motions, for example, crawling under a low-hanging ceiling or quickly avoiding an incoming projectile.

   PlaMo can handle various scene complexities, but has some limitations which we outline as future work. **Interaction with objects and multiple agents.** Many interesting scenarios for gaming and simulation would require physical interactions with dynamic objects or other agents. Learning such interactions not only requires specific data but may require extending the human model and potentially the controller training, for example with hands [Pavlakos et al. 2019]. **Dynamic objects.** *PlaMo* handles objects whose motion can be easily predicted using rules. Modeling more complex dynamic objects may require learning-based approaches to model the dynamic scene. **Rich language and scene understanding.** *PlaMo* focuses on planning and control. It opens opportunities for combining it with modern language models and 3D scene understanding. With these expansions in mind, we see *PlaMo* as a stepping stone in a much greater system where non-playable characters (NPCs) are given roles to play, forming a rich simulated virtual world.

Preprint

References

Alfonso Alcántara, Jesús Capitán, Rita Cunha, and Aníbal Ollero. 2021. Optimal trajectory planning for cinematography with multiple unmanned aerial vehicles. *Robotics and Autonomous Systems* 140 (2021), 103778.

German Barquero, Sergio Escalera, and Cristina Palmero. 2023. Belfusion: Latent diffusion for behavior-driven human motion prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2317–2327.

Michael GH Bell. 2009. Hyperstar: A multi-path Astar algorithm for risk averse vehicle navigation. *Transportation Research Part B: Methodological* 43, 1 (2009), 97–107.

Pengzhan Chen, Jiean Pei, Weiqing Lu, and Mingzhen Li. 2022. A deep reinforcement learning based method for real-time path planning and dynamic obstacle avoidance. *Neurocomputing* 497 (2022), 64–75.

František Duchoň, Andrej Babinec, Martin Kajan, Peter Beňo, Martin Florek, Tomáš Fico, and Ladislav Jurišica. 2014. Path planning with modified a star algorithm for a mobile robot. *Procedia engineering* 96 (2014), 59–69.

Chuan Guo, Shihao Zou, Xinxin Zuo, Sen Wang, Wei Ji, Xingyu Li, and Li Cheng. 2022. Generating diverse and natural 3d human motions from text. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5152–5161.

Mohamed Hassan, Yunrong Guo, Tingwu Wang, Michael Black, Sanja Fidler, and Xue Bin Peng. 2023. Synthesizing Physical Character-Scene Interactions. *arXiv preprint arXiv:2302.00883* (2023).

KS Holkar and Laxman M Waghmare. 2010. An overview of model predictive control. *International Journal of control and automation* 3, 4 (2010), 47–63.

Yu-Hsin Hsu and Rung-Hung Gau. 2020. Reinforcement learning-based collision avoidance and optimal trajectory planning in UAV communication networks. *IEEE Transactions on Mobile Computing* 21, 1 (2020), 306–320.

Jordan Juravsky, Yunrong Guo, Sanja Fidler, and Xue Bin Peng. 2022. PADL: Language-Directed Physics-Based Character Control. In *SIGGRAPH Asia 2022 Conference Papers* (Daegu, Republic of Korea) *(SA '22)*. Association for Computing Machinery, New York, NY, USA, Article 19, 9 pages. https://doi.org/10.1145/3550469.3555391

Marcelo Kallmann. 2010. Shortest Paths with Arbitrary Clearance from Navigation Meshes.. In *Symposium on Computer Animation*. 159–168.

Marcelo Kallmann and Mubbasir Kapadia. 2014. Navigation meshes and real-time dynamic planning for virtual worlds. In *ACM SIGGRAPH 2014 Courses*. 1–81.

Korrawe Karunratanakul, Konpat Preechakul, Supasorn Suwajanakorn, and Siyu Tang. 2023. Guided motion diffusion for controllable human motion synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2151–2162.

Sunmin Lee, Sebastian Starke, Yuting Ye, Jungdam Won, and Alexander Winkler. 2023. QuestEnvSim: Environment-Aware Simulated Motion Tracking from Sparse Sensors. *arXiv preprint arXiv:2306.05666* (2023).

Jiaman Li, Jiajun Wu, and C Karen Liu. 2023. Object motion guided human motion synthesis. *ACM Transactions on Graphics (TOG)* 42, 6 (2023), 1–11.

Hung Yu Ling, Fabio Zinno, George Cheng, and Michiel Van De Panne. 2020. Character controllers using motion vaes. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 40–1.

Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. 2015. SMPL: A Skinned Multi-Person Linear Model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)* 34, 6 (Oct. 2015), 248:1–248:16.

Zhengyi Luo, Jinkun Cao, Kris Kitani, Weipeng Xu, et al. 2023a. Perpetual humanoid control for real-time simulated avatars. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 10895–10904.

Zhengyi Luo, Jinkun Cao, Josh Merel, Alexander Winkler, Jing Huang, Kris Kitani, and Weipeng Xu. 2023b. Universal Humanoid Motion Representations for Physics-Based Control. *arXiv preprint arXiv:2310.04582* (2023).

Zhengyi Luo, Shun Iwase, Ye Yuan, and Kris Kitani. 2022. Embodied scene-aware human pose estimation. *Advances in Neural Information Processing Systems* 35 (2022), 6815–6828.

Naureen Mahmood, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll, and Michael J. Black. 2019. AMASS: Archive of Motion Capture as Surface Shapes. In *International Conference on Computer Vision*. 5442–5451.

Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. 2021. Isaac Gym: High Performance GPU Based Physics Simulation For Robot Learning. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Aye Aye Maw, Maxim Tyan, and Jae-Woo Lee. 2020. iADA*: improved anytime path planning and replanning algorithm for autonomous vehicle. *Journal of Intelligent & Robotic Systems* 100 (2020), 1005–1013.

Jeff Michels, Ashutosh Saxena, and Andrew Y Ng. 2005. High speed obstacle avoidance using monocular vision and reinforcement learning. In *Proceedings of the 22nd international conference on Machine learning*. 593–600.

R OpenAI. 2023. GPT-4 technical report. *arXiv* (2023), 2303–08774.

Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed A. A. Osman, Dimitrios Tzionas, and Michael J. Black. 2019. Expressive Body Capture: 3D Hands, Face, and Body from a Single Image. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 10975–10985.

Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel Van de Panne. 2018. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions On Graphics (TOG)* 37, 4 (2018), 1–14.

Xue Bin Peng, Glen Berseth, Kangkang Yin, and Michiel Van De Panne. 2017. DeepLoco: Dynamic Locomotion Skills Using Hierarchical Deep Reinforcement Learning. *ACM Trans. Graph.* 36, 4, Article 41 (July 2017), 13 pages. https://doi.org/10.1145/3072959.3073602

Xue Bin Peng, Yunrong Guo, Lina Halper, Sergey Levine, and Sanja Fidler. 2022. ASE: Large-scale Reusable Adversarial Skill Embeddings for Physically Simulated Characters. *ACM Trans. Graph.* 41, 4, Article 94 (July 2022).

Xue Bin Peng, Ze Ma, Pieter Abbeel, Sergey Levine, and Angjoo Kanazawa. 2021. Amp: Adversarial motion priors for stylized physics-based character control. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–20.

S Joe Qin and Thomas A Badgwell. 1997. An overview of industrial model predictive control technology. In *AIche symposium series*, Vol. 93. New York, NY: American Institute of Chemical Engineers, 1971-c2002., 232–256.

Zhongfei Qing, Zhongang Cai, Zhitao Yang, and Lei Yang. 2023. Story-to-Motion: Synthesizing Infinite and Controllable Character Animation from Long Text. arXiv:2311.07446 [cs.CV]

Davis Rempe, Zhengyi Luo, Xue Bin Peng, Ye Yuan, Kris Kitani, Karsten Kreis, Sanja Fidler, and Or Litany. 2023. Trace and Pace: Controllable Pedestrian Animation via Guided Trajectory Diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 13756–13766.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

Sebastian Starke, He Zhang, Taku Komura, and Jun Saito. 2019. Neural state machine for character-scene interactions. *ACM Trans. Graph.* 38, 6 (2019), 209–1.

Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*.

Chen Tessler, Yoni Kasten, Yunrong Guo, Shie Mannor, Gal Chechik, and Xue Bin Peng. 2023. Calm: Conditional adversarial latent models for directable virtual characters. In *ACM SIGGRAPH 2023 Conference Proceedings*. 1–9.

Wouter G Van Toll, Atlas F Cook IV, and Roland Geraerts. 2012. A navigation mesh for dynamic environments. *Computer Animation and Virtual Worlds* 23, 6 (2012), 535–546.

Tingwu Wang, Yunrong Guo, Maria Shugrina, and Sanja Fidler. 2020a. Unicon: Universal neural controller for physics-based character motion. *arXiv preprint arXiv:2011.15119* (2020).

Wenjie Wang, Qing Tao, Yuting Cao, Xiaohua Wang, and Xu Zhang. 2020b. Robot time-optimal trajectory planning based on improved cuckoo search algorithm. *IEEE access* 8 (2020), 86923–86933.

Yinhuai Wang, Jing Lin, Ailing Zeng, Zhengyi Luo, Jian Zhang, and Lei Zhang. 2023. Physhoi: Physics-based imitation of dynamic human-object interaction. *arXiv preprint arXiv:2312.04393* (2023).

Jungdam Won, Deepak Gopinath, and Jessica Hodgins. 2022. Physics-based character controllers using conditional VAEs. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 1–12.

Zhaoming Xie, Hung Yu Ling, Nam Hee Kim, and Michiel van de Panne. 2020. Allsteps: curriculum-driven learning of stepping stone skills. In *Computer Graphics Forum*, Vol. 39. Wiley Online Library, 213–224.

Junfeng Yao, Chao Lin, Xiaobiao Xie, Andy JuAn Wang, and Chih-Cheng Hung. 2010. Path planning for virtual human motion using improved A* star algorithm. In *2010 Seventh international conference on information technology: new generations*. IEEE, 1154–1158.

Lee Yeong-Ho, Kim Yeong-Jun, Jeong Da-Un, and Weon Ihn-Sik. 2020. Development of an integrated path planning algorithm for autonomous driving of unmanned surface vessel. In *2020 20th International Conference on Control, Automation and Systems (ICCAS)*. IEEE, 27–32.

Ye Yuan and Kris Kitani. 2019. Ego-pose estimation and forecasting as real-time pd control. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 10082–10092.

Ye Yuan and Kris Kitani. 2020. Residual force control for agile human behavior imitation and extended motion synthesis. *Advances in Neural Information Processing Systems* 33 (2020), 21763–21774.

Haotian Zhang, Ye Yuan, Viktor Makoviychuk, Yunrong Guo, Sanja Fidler, Xue Bin Peng, and Kayvon Fatahalian. 2023b. Learning physically simulated tennis skills from broadcast videos. *ACM Transactions On Graphics (TOG)* 42, 4 (2023), 1–14.

Yunbo Zhang, Deepak Gopinath, Yuting Ye, Jessica Hodgins, Greg Turk, and Jungdam Won. 2023a. Simulation and retargeting of complex multi-character interactions. In *ACM SIGGRAPH 2023 Conference Proceedings*. 1–11.

Xunyu Zhong, Jun Tian, Huosheng Hu, and Xiafu Peng. 2020. Hybrid path planning based on safe A* algorithm and adaptive window approach for mobile robot in large-scale dynamic environment. *Journal of Intelligent & Robotic Systems* 99 (2020), 65–77.

Kun Zhou, Qiming Hou, Rui Wang, and Baining Guo. 2008. Real-time kd-tree construction on graphics hardware. *ACM Transactions on Graphics (TOG)* 27, 5 (2008), 1–11.

(a) Crawling up a slope   (b) Crawling on gravel   (c) Crawling down stairs

(d) Crouch-walking up a slope   (e) Crouch-walking on rough terrain   (f) Crouch-walking down stairs

(g) Walking up a slope   (h) Walking on rough terrain   (i) Walking down stairs

(j) Running up a slope   (k) Running on rough terrain   (l) Running down stairs
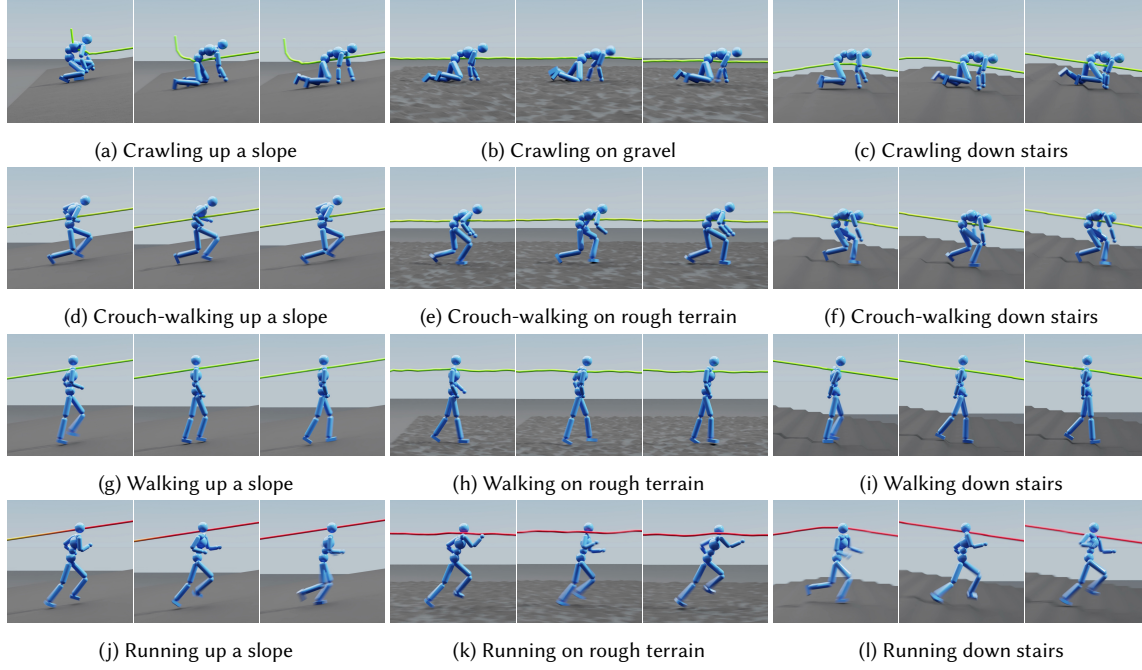
Fig. 7. **Locomotion:** Screen captures of our trained controller when conditioned on various forms of locomotion across varying types of terrains. Crawling is achieved by requesting a head-height of $0.4\,[m]$, crouching at $0.8\,[m]$, walking and running at upright $1.47\,[m]$. The corresponding speeds are $1\,[m/s]$, $2\,[m/s]$, $2\,[m/s]$, and $4\,[m/s]$.

(a) **Slalom test with constant speed (baseline).** The humanoid follows the path poorly, hitting the walls on every turn.

(b) **Slalom test with adaptive speed (ours).** The humanoid follows the path well, never hitting the walls.
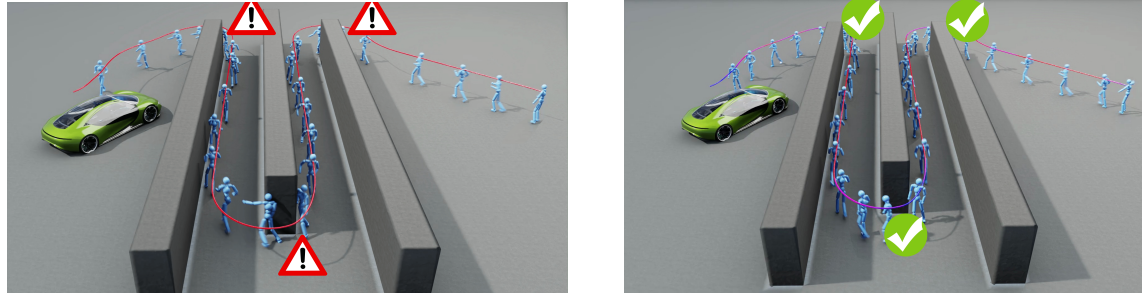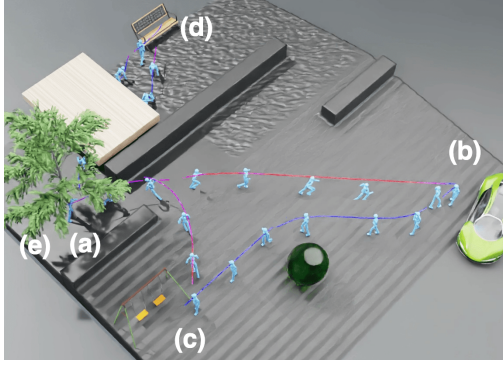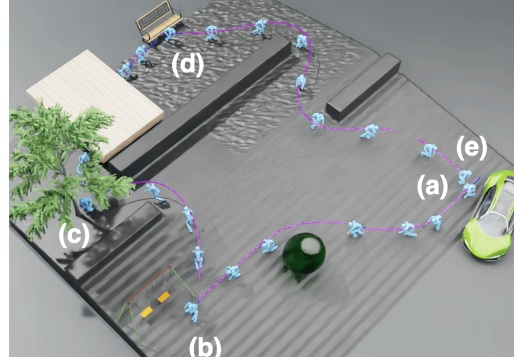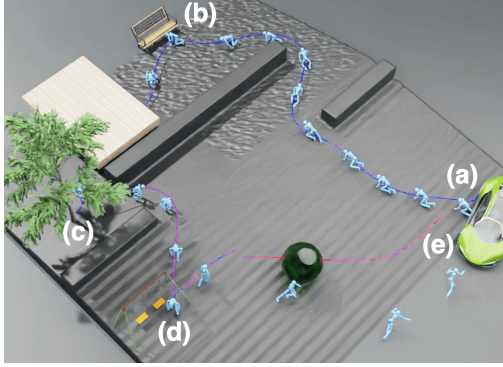
Fig. 8. **Comparison of consistent and adaptive speed profiles on the slalom test.** The colors represent the speed, with red indicating speeds above 3.5m/s and blue below 1m/s.
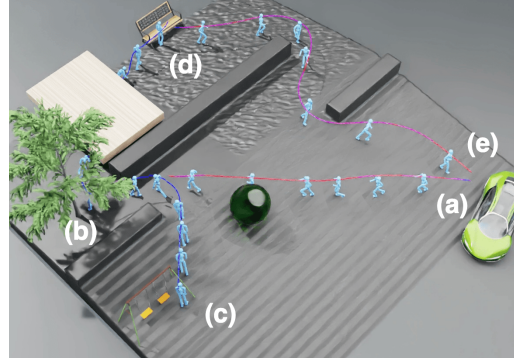
(a) Starting at the tree: "Run to the car" then "walk to the swing" then "run to the bench" then "walk to the tree".

(b) Starting at the car: "Walk crouching to the bench" then "walk crouching to the tree" then "walk crouching to the bench" then "walk crouching to the car".
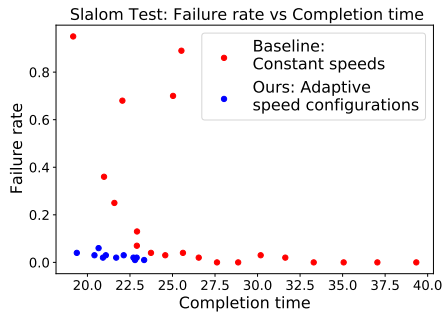

(c) Starting at the car: "Crawl to the bench" then "crawl to the tree" then "walk crouching to the bench" then "sprint to the car"
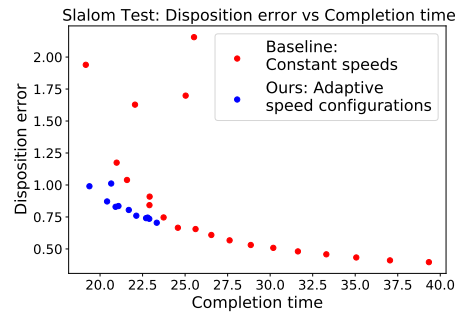

(d) Starting at the car: "Run to the tree" then "walk to the swing" then "walk to the bench" then "run to the car".

Fig. 9. **Diverse solutions:** Our system can solve unseen tasks in diverse forms. Here, we show how the character traverses a random sequence of segments within the scene. Each example showcases a randomly sampled sequence of waypoints to be performed with a randomly sampled motion.

(a) **Failure rate and Completion Time: Pareto curve in the Slalom test**. Testing our speed controller vs. constant speed. Our planner obtains lower failure rate while retaining faster completion.

(b) **Disposition error and Completion Time: Pareto curve in the Slalom test.** Testing our our speed controller vs. constant speed. Our planner obtains lower disposition error while retaining faster completion.

Preprint

A Data curation

To train our motion controller, we consider three datasets. First, the entire AMASS dataset contains various motions, such as dancing, cartwheels, and sitting. Second, we consider an automatic curation method. This uses a large language model (LLM, ChatGPT4 [OpenAI 2023]). We the LLM with the textual descriptions of each movement and ask it to categorize the motions into five classes: ['walking', 'running', 'crouching', 'crawling', and others]. These textual labels are extracted from the HumanML3D dataset [Guo et al. 2022]. This filtering reduced the size of the dataset down to 600 motions. Finally, we also tested a manual curation procedure. For manual curation, we handpick 75 motion captures, roughly 20 from each desired motion type, and create a small, tailor-made dataset.

B Additional results

In Tables 3 - 7 we present the full results for the ablation study given in Table 1. We show the XY and Z errors for every choice of data filtering and path generator speed samples, for every terrain and every motion type.

Table 3. **No data filtering. Naive path generator. Locomotion controller errors.**

Terrain	Crawl		Crouch-walk		Walk		Run	
	XY-err $[m]$	Z-err$[m]$	XY-err $[m]$	Z-err $[m]$	XY-err$[m]$	Z-err $[m]$	XY-err$[m]$	Z-err$[m]$
Flat	1.11	0.20	0.43	0.11	0.35	0.31	0.76	0.36
Slope	1.12	0.20	0.46	0.11	0.38	0.32	0.82	0.36
Rough	1.39	0.19	0.53	0.13	0.44	0.34	1.06	0.40
Stairs	1.44	0.21	0.78	0.16	0.53	0.36	1.19	0.42

Table 4. **No data filtering. Coupled path generator. Locomotion controller errors.**

Terrain	Crawl		Crouch-walk		Walk		Run	
	XY-err $[m]$	Z-err$[m]$	XY-err $[m]$	Z-err $[m]$	XY-err$[m]$	Z-err $[m]$	XY-err$[m]$	Z-err$[m]$
Flat	0.43	0.07	0.39	0.07	0.37	0.31	0.74	0.36
Slope	0.49	0.08	0.42	0.07	0.40	0.32	0.81	0.38
Rough	0.66	0.09	0.49	0.08	0.45	0.34	1.03	0.42
Stairs	0.94	0.13	0.67	0.10	0.53	0.35	1.13	0.43

Table 5. **Naive data filtering. Naive path generator. Locomotion controller errors.**

Terrain	Crawl		Crouch-walk		Walk		Run	
	XY-err $[m]$	Z-err$[m]$	XY-err $[m]$	Z-err $[m]$	XY-err$[m]$	Z-err $[m]$	XY-err$[m]$	Z-err$[m]$
Flat	0.62	0.15	0.26	0.05	0.29	0.27	0.60	0.30
Rough	1.16	0.17	0.37	0.07	0.35	0.28	0.87	0.36
Stairs	1.24	0.19	0.52	0.11	0.43	0.30	1.02	0.37

Table 6. **Naive data filtering. Coupled path generator. Locomotion controller errors.**

	Crawl		Crouch-walk		Walk		Run	
Terrain	XY-err [m]	Z-err[m]	XY-err [m]	Z-err [m]	XY-err[m]	Z-err [m]	XY-err[m]	Z-err[m]
Flat	0.67	0.06	0.25	0.06	0.25	0.23	0.57	0.29
Slope	0.55	0.07	0.26	0.07	0.30	0.23	0.64	0.30
Rough	0.78	0.09	0.32	0.07	0.34	0.26	0.85	0.35
Stairs	0.80	0.13	0.48	0.10	0.42	0.28	1.01	0.36

Table 7. **Curated data filtering. Naive path generator. Locomotion controller errors.**

	Crawl		Crouch-walk		Walk		Run	
Terrain	XY-err [m]	Z-err[m]	XY-err [m]	Z-err [m]	XY-err[m]	Z-err [m]	XY-err[m]	Z-err[m]
Flat	0.29	0.09	0.25	0.04	0.21	0.21	0.49	0.27
Slope	0.34	0.12	0.26	0.04	0.24	0.22	0.57	0.28
Rough	0.50	0.13	0.29	0.05	0.27	0.23	0.78	0.32
Stairs	0.81	0.17	0.43	0.07	0.36	0.26	0.97	0.35

C   Additional technical details for Section 3.1

As a final step, the output format of the path planner (x,y,z,v) is translated into a format suitable for the motion controller. This controller requires (x, y, z) waypoints spaced at fixed time intervals, where a larger waypoint spacing corresponds to a higher speed. The conversion from the spatially defined path to this temporal trajectory format is accomplished through an iterative algorithm supplemented by linear interpolation.

D   Additional experiments for the speed controller

Here, we provide additional results on the slalom test from Figure 8. We record the paths the humanoid traversed with a constant speed, as well as our speed controller output, and compare them with the desired reference path. The constant speed was selected so the two runs had the same completion time. The result is shown in Figure 11. Despite having the same average speeds, in the constant-speed run, the humanoid has a difficult time matching the speed during the steep turns and hits the wall at its corners. This does not happen with the adaptive speed run, which follows the desired path better.

E   Supplementary files

We provide as addition to the paper supplementary videos to demonstrate PlaMo:

(1) Videos of the locomotion controller in terrain X motion type that correspond to Table 2 and Figure 7.
(2) Videos of the humanoid doing the Slalom scene presented in Figure 8.
(3) Videos of randomized route and motion types in a diverse scene, corresponding to Figure 9

We also provide yaml files that correspond to the filtered dataset considered for the ablation study given in Table 1.
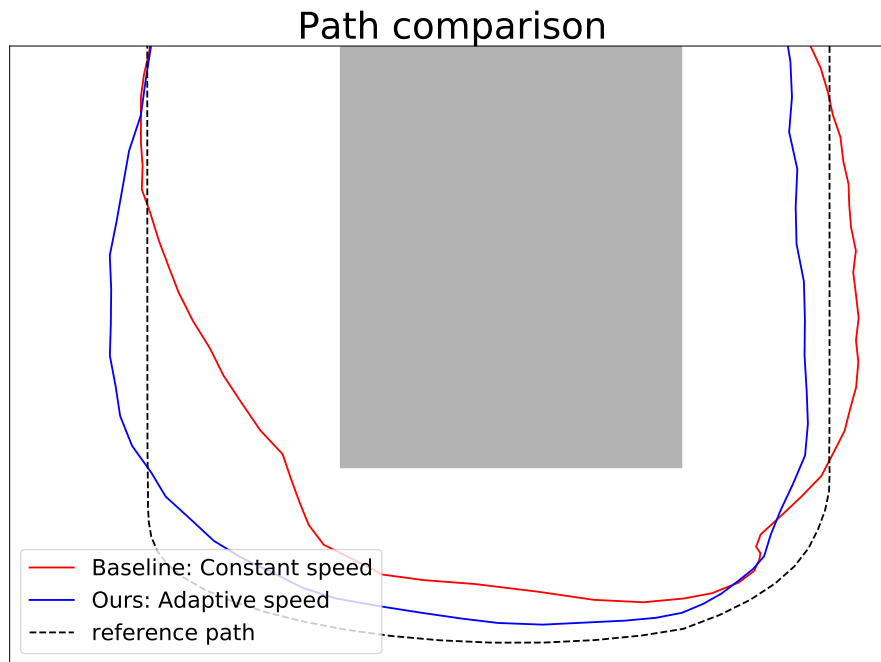
Fig. 11. **Comparison of planned paths** (blue) Our path with adaptive speed. (red) A path with a constant speed that has the same average speed as the adaptive-speed path. (dashed black) the desired reference path. When using a constant speed, the humanoid hits the wall at its corners. Using the adaptive speed generates a fast but smoother path, that is far from the obstacle but still follows follows well the desired path better.