Quadruped walking algorithms
Spline Approach for Trajectory Generation

Team Member: Jan Górski
jagor24@student.sdu.dk

Author: Alan Bejnarowicz
albej24@student.sdu.dk

Project repository: https://github.com/AlanBejnarowicz/AECS

Index Terms—

Qdruped robots, walking algorithms, spline interpolation, trajectory generation, smooth motion planning

I. Introduction

Our interest in a quadruped robot locomotion cames from our previous project. During our bachelor studies we built a quadrupedal robot that was able to walk using a simple predefined gait. However, the robot's movement was jerky and lacked smoothness, mostly because of used motors.

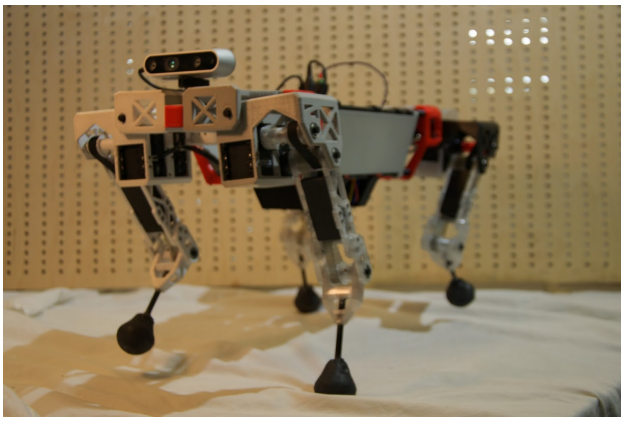

Fig. 1. Our homemade quadruped robot built during bachelor studies.

This project is a nice opportunity to explore more about walking algorithms for quadruped robots. We want to focus on generating smooth trajectories for the robot's legs using spline interpolation techniques.

II. Description of the problem

Quadruped robots are very complex to make. They are complex in a mechanical design as well as in control algorithms. The goal of this project is to implement walking algorithms for a quadruped robot. My focus will be on generating smooth trajectories for the robot's legs using splines. Jan Górski will be implementing the Trajectory Generation using reinforcement learning approach. In the end, we will compare both methods and evaluate. Both algorithm will be tested in a Unitree Mujoco simulation environment with a Unitree GO2 robot model. Both algorithm should be able to walk forward on a flat and obstacle terrain.

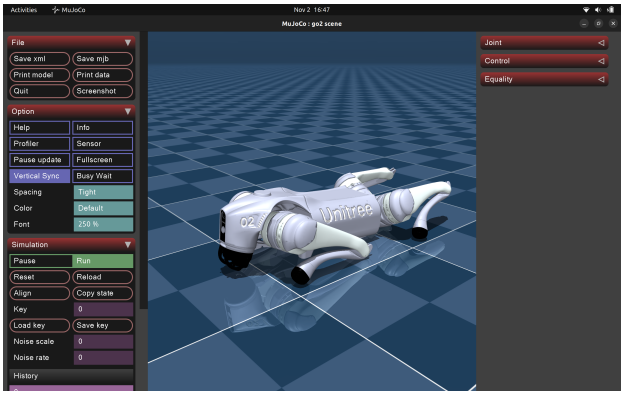

Fig. 2. Unitree GO2 quadruped robot in the Mujoco simulation.

A. Mujoco simulation enviroment

Mujoco is a physics engine that is widely used for simulating robots maintained by DeepMind. Initial paper describing Mujoco [5] was published in 2012. It is known for its speed and accuracy in simulating complex robotic systems.

Unitree Robotics provides a Mujoco simulation environment for their GO2 robot [6]. This environment includes a detailed model of the robot, as well as various terrains and obstacles for testing walking algorithms. Unitree provides exaple of Sim to Real transfer. That makes this project "testable" in real world after the simulation phase.

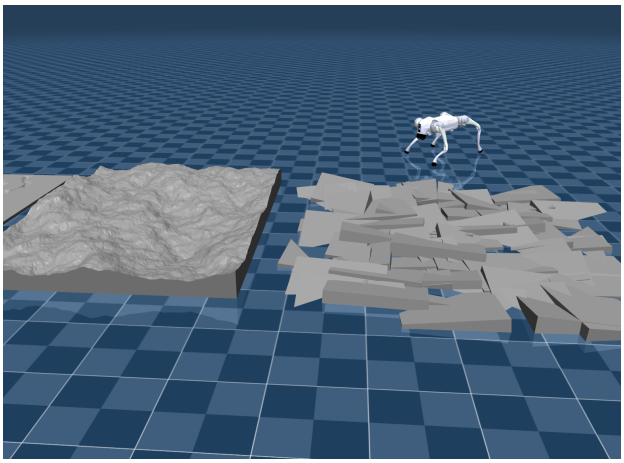

Fig. 3. Unitree GO2 robot and rough terrain in Mujoco simulation.

B. Unitree GO2 Robot

The Unitree GO2 robot [2] is a quadruped robot designed for research and development in robotics. It features a lightweight design, high mobility, and advanced sensors for navigation and obstacle avoidance. The robot is equipped with 12 degrees of freedom, allowing for complex movements and maneuvers. This robot is widely used in research due to its affordability and versatility. Cost for the robot for 1st of December 2025 is 1600 USD.

Unitree GO2 robot weighs 15kg and can output a maximum of 3kW of power. Each motor in a leg can provide up to 45Nm of torque. Each version of robot is eqquipped in 8 core CPU and educational version could be equipped in Nvida Jetson Orin SBC. These parameters make the GO2 robot very capable platform for research in quadruped locomotion. In my approach to control the robot high computing power is not required as the trajectory generation could be precomputed and stored in a look-up table.

III. Problems to be solved

In the begin I will list all the problems that I had to solve in order to implement the spline-based trajectory generation for the quadruped robot:

- Communication the python code with the Mujoco simulation environment.
- Implementing the inverse kinematics for the Unitree GO2 robot to convert desired foot positions into joint angles.
- Developing the spline trajectory.
- Implement phase generation for the walking gait to coordinate leg movements.
- Implement PID controller to balance the robot and maintain stability.

IV. Inverse Kinematics

First major problem I needed to solve is to implementation of inverse kinematics for the Unitree Go2 robot. Very helpful resource was an online article explaining the inverse kinematics for this specific robot [3]. The article provided the dimmensions of the robot's leg segments. Then I using geometric approach I derived the equations to calculate the joint angles based on the desired foot position in the robot's leg coordinate frame.

Implementation of the inverse kinematics is in the file `GO2_IK.py`.

V. Phase and Spline Trajectory Generation

A. Gait phase generator

The gait phase generator is responsible for coordinating the movements of the robot's legs during walking. It defines the possition of each leg in the gait cycle, from given spline trajectory. The gait phase generator uses a time-based approach to determine the phase of each leg, ensuring that the legs move in a coordinated manner. Algorithm uses the following equation to calculate the phase position of each leg 1:

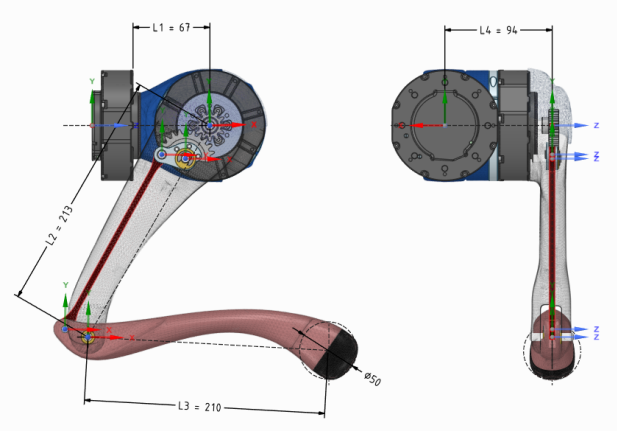

Fig. 4. Unitree GO2 leg dimensions used for inverse kinematics calculations.

$$\text{cycle\_pos} = \left(\frac{t}{T_{\text{cycle}}} + \text{offset} \right) \bmod 1 \tag{1}$$

Where t is the current time, T_{cycle} is the duration of a full gait cycle, and offset is a phase offset for each leg to ensure proper

Current offset sequencing is in Table I. This sequencing is used by multipla animals such as dogs or horses while trotting.

TABLE I
Gait phase offsets for each leg.

Leg	Phase Offset
Front Right	0.0
Rear Left	0.0
Front Left	0.5
Rear Right	0.5

B. Spline Trajectory Generation

The spline trajectory generation module is responsible for creating actual foot trajectories. It uses cubic splines to interpolate between key waypoints, ensuring smooth and continuous motion. The spline generation process involves defining control points for each leg's trajectory and then using these points to create cubic spline functions.

Current approach uses 3 predefined waypoints for each leg's trajectory. The waypoints are defined as follows:

- Start Point: (0, 0, 0) - The initial position of the foot.
- Mid Point: (Step Lenght/2, 0, Step Height) - The highest point of the swing phase.
- End Point: (Step Lenght, 0, 0) - The final position of the foot.

Current version uses these settings for the step parameters:

- Step Lenght = 0.22
- Step Height = 0.12
- Cycle Time = 0.6 — time of full gait cycle
- Duty = 0.6 — portion of the cycle when the foot is on the ground

These settings can be tweeked to achieve different walking styles and speeds. For example increasing the step height could help the robot to walk on uneven terrain.

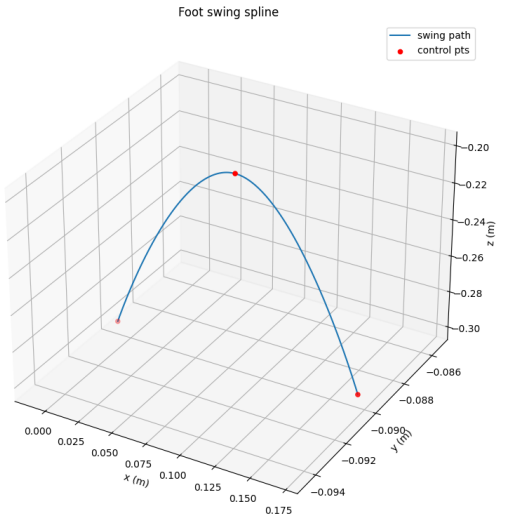

Fig. 5. Cubic spline trajectory for a leg swing motion.

C. Different trajectory

One example to tune the trajectory was to use 4 waypoints to flatten the top of the spline curve. This was done to reduce the acceleration at the top of the swing phase, which could lead to a more stable walking motion. Trajectory is shown in figure 6.

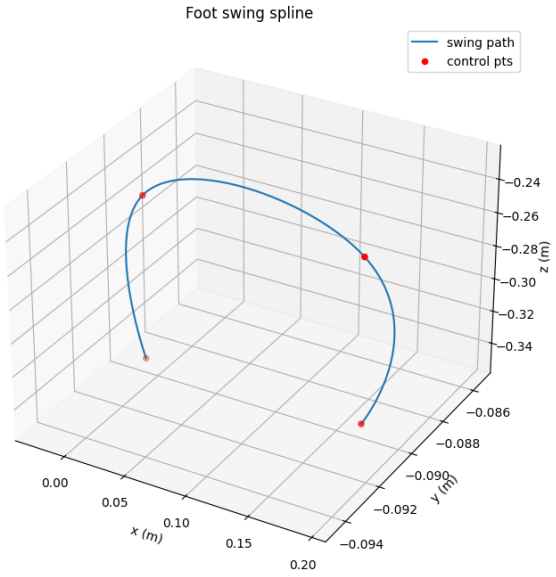

Fig. 6. Fast walking cubic spline trajectory for a leg swing motion.

Additionaly I changed the step parameters to achieve a faster walking speed: Video of the robot walking with this trajectory can be found at https://youtu.be/nRwTH8aU0jQ

D. Spline improvements

For this day I focused on implementing the basic spline trajectory generation and gait phase coordination. However, there are several potential improvements that could be made to enhance the system. The major one would be to implement a system to dynamically adjust the step parameters based on the robot's speed and terrain. This could involve using sensors to detect the ground and adjust the step height and length accordingly. Additionally, implementing more advanced gait patterns, such as walking, trotting, and galloping, could provide more versatility to the robot's movement capabilities. Future work could also explore the use of machine learning techniques to optimize the gait patterns based on feedback from the robot's performance.

VI. PID BALANCING CONTROLLER

To make robot more stable during walking I implemented a PID controller to adjust the body orientation based on IMU data from the simulation. The controller calculates the error between the desired and actual orientation and computes the necessary adjustments witch are then applied to legs Z axis positions. That simple way to help maintain balance during walking and reduce body tilting happend to be very effective.

To balance in both pitch and roll axes I implemented two separate PD controllers. The PD parameters were tuned through trial and error to achieve optimal performance. Whole control loop runs at DT = 0.002s.

VII. CONTROL SYSTEM

Whole control system diagram is shown in Fig. 7. Whole code is about 500 lines of python code and uses simple math. Whole algorithm runs in real-time in the Mujoco simulation environment and easyly could be implemented on a real robot even on a microcontroller with limited computing power.

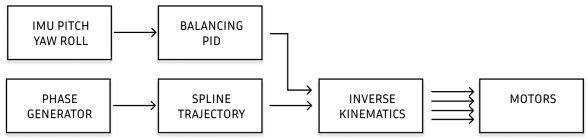

Fig. 7. Whole control system diagram including phase generation, spline trajectory, inverse kinematics, and PID balancing controller.

Phase generation module creates phase offsets for each leg to ensure proper gait timing. Then these phases are used by the spline trajectory generation module to update foot positions in real-time. The inverse kinematics module then converts these foot positions into joint angles for the robot's legs. Finally, the PID balancing controller adjusts the height of each leg to maintain the robot's balance based on its current orientation.

Right now the system is only capable of walking in a straight line. Future should include the ability to turn and change directions and more advanced gaits.

VIII. Results

Algorithm ended up beeing very successful. Robot is able to walk smoothly on a flat terrain. What was more suprising is that robot is also able to walk on an uneven terrain with small obstacles. The PID balancing controller helped a lot with maintaining stability during walking.

Another practical implementation could be robots walking in a theme parks by tweeking the gait parameters it is easy to achieve funny walking styles. Overall, the spline-based trajectory generation approach proved to be effective for controlling the quadruped robot's walking gait in simulation.

Robots like that arleady exists in a Disney theme park, from available information their approach uses reinforcement learning to train the walking gait. Figure 8 shows a bipedal robot. Same approach is implemented by Jan Górski.

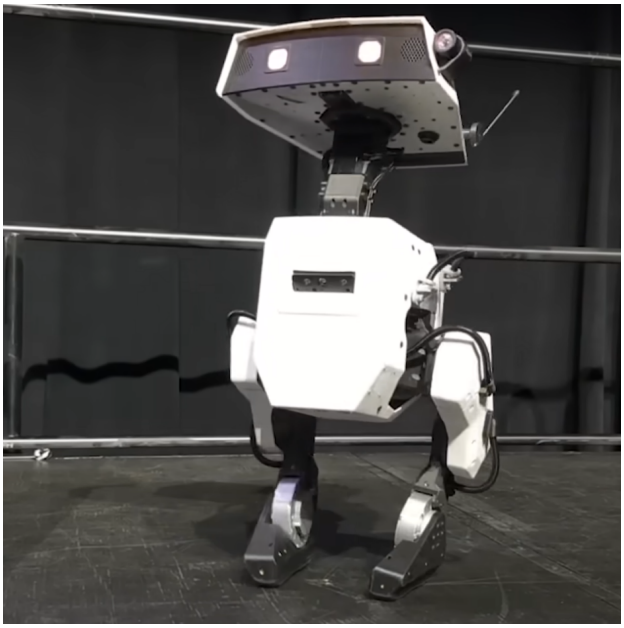

Fig. 8. Disney's Imagineering bipedal robot. [7]

A. Comparision to between Splines and RL algorithms

Both spline-based trajectory generation and reinforcement learning (RL) approaches have their own advantages and disadvantages when it comes to controlling quadruped robot locomotion. Key difference is that spline-based approach is simpler to implement and requires computing power of medium size calculator while RL approach requires a lot of computing power to train the model and then run model in real-time. Despite this difference spline approach can be expanded and achieve similar results to RL approach.

B. Reinforcement Learning [Jan Górski]

This method involves training a neural network to learn optimal walking gaits through trial and error. The robot receives rewards for successful movements and penalties for failures, allowing it to gradually improve its performance over time. To train the RL model Jan used SDU UCloud computing resources. Even with dedicated GPU it took several hours to train the model. Researches from ETH Zürich [8] have demonstrated that RL approach can achieve impressive results in quadruped locomotion, including walking on very challenging terrain. A video demonstrating their results can be found at [9].

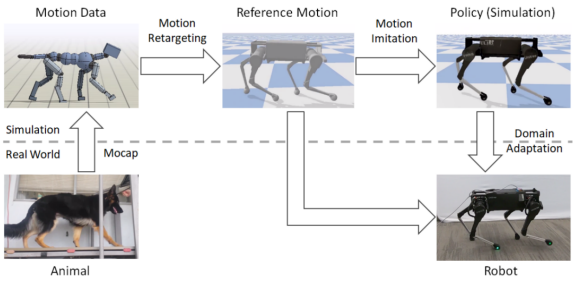

Fig. 9. The motion imitation pipeline graphic from Jan Górski's report.

RL approach has the advantage of being able to adapt to a wide range of terrains and conditions, as the model can learn to generalize from its training data. What RL enables for researches is to prepare training data using motion capture from animals. This way robot can imitate real animal movements.

C. Spline foot trajectory

My approach is based on generating smooth foot trajectories using cubic splines. This method does not require high computing power nor require training. However, witch a proper tuning and additional modules witch modifies the foot trajectory in real-time it can achieve similar results to RL approach. Researchers from Brazil [10] have demonstrated that spline-based trajectory generation can be effective for controlling quadruped robots, including walking on uneven terrain. Great advatange of spline approach is that it is predictable and stable. It is just a math function that could be analyzed and prooved to be stable.

D. Comparison with RL method

As a good metric to compare both methods we used IMU data, to measure robot stability during walking on flat terrain. To do so we recorded Gyroscope and Accelerometer data from both robots during walking. Figures 10 and 11 shows the recorded data. It is visible that in both cases robots achieves similar angular velocities but the accelerometer data shows that spline-based robot has lower acceleration spikes. This suggests that the spline-based robot is more stable during walking, which is a desirable trait for quadruped locomotion.

Another important test we pergormed was a Fourier Transform analysis of the robot's movement during walking. This analysis helps to identify the dominant frequencies in the robot's motion, which can provide insights into its stability and responsiveness. Figures 12 and 13 shows the FFT results for both robots. In this case Spline controller shows

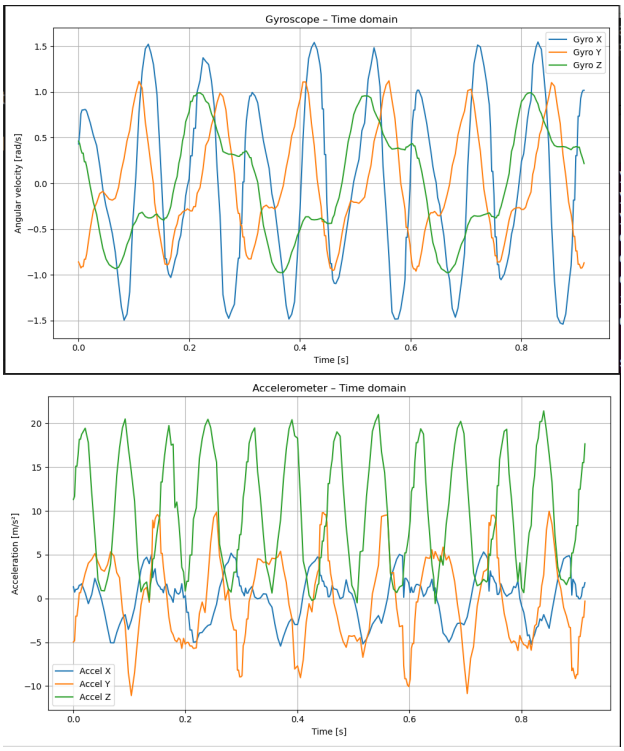

Fig. 10. IMU data from Spline Algorithm. Top: Gyroscope data, Bottom: Accelerometer data.

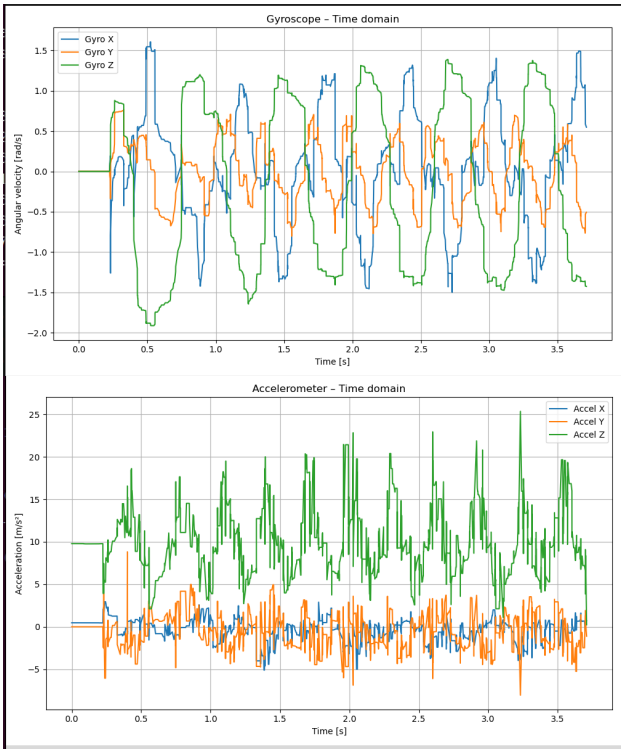

Fig. 11. IMU data from RL Algorithm. Top: Gyroscope data, Bottom: Accelerometer data.

amplitude spikes at higher frequencies where RL controller has most of the energy concentrated at lower frequencies. This suggests that the RL-controlled robot has a smoother and more stable gait, as lower frequency movements are generally associated with better stability. That could be probably solved by tweeking PID parameters of the balancing controller and PD controller for legs. In real life application that would be very important as high frequency vibrations could damage the robot's hardware over time.

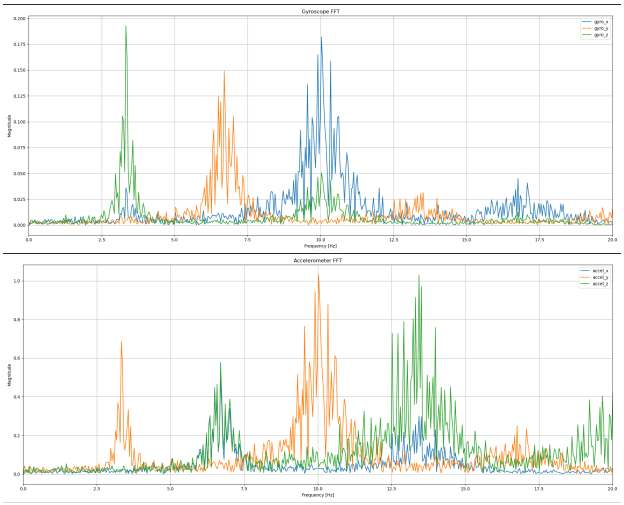

Fig. 12. IMU Fourier analysis data from Spline Algorithm. Top: Gyroscope data, Bottom: Accelerometer data.

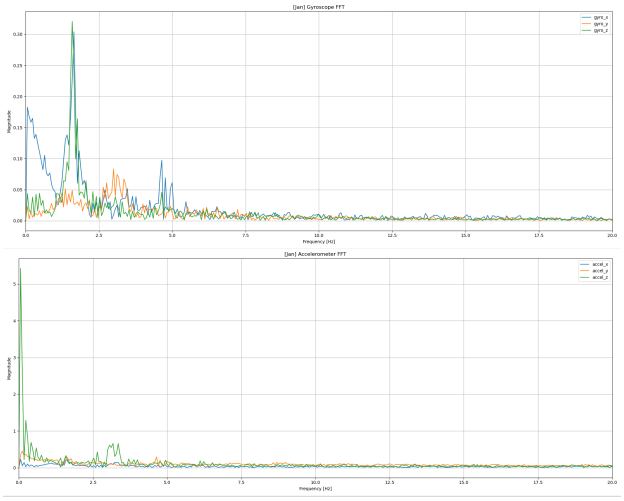

Fig. 13. IMU Fourier analysis data from RL Algorithm. Top: Gyroscope data, Bottom: Accelerometer data.

IX. All YouTube videos links

YouTube videos demonstrating the results can be found here:

- Fast walking: https://youtu.be/nRwTH8aU0jQ
- Heavy terrain: https://youtu.be/4OVWzleJ_GU
- No PID funny setup walking: https://www.youtube.com/watch?v=eI8ZH1-Hwkk

- Flat terrain no PID balancing: https://youtu.be/eI8ZH1-Hwkk

REFERENCES

[1] Unitree Robotics, "unitree_mujoco" GitHub repository. Available: https://github.com/unitreerobotics/unitree_mujoco. Accessed: Dec. 1, 2025.

[2] Unitree Robotics, "Unitree GO2 Robot" Product Page. Available: https://shop.unitree.com/products/unitree-go2?srsltid=AfmBOooo_NzRU6B7Nf0OCNzLnS0lI8Q6H5uuD0nauSsT0dyTkfN_iFYb. Accessed: Dec. 1, 2025.

[3] Unitree Go2 Inverse Kinematics Article. Available: https://observablehq.com/@christophe-yamahata/inverse-kinematics-go2-robot. Accessed: Dec. 1, 2025.

[4] Disney Imagineering Bipedal Robots. Available: https://www.youtube.com/watch?v=-cfIm06tcfA. Accessed: Dec. 1, 2025.

[5] E. Todorov, T. Erez and Y. Tassa, "MuJoCo: A physics engine for model-based control," 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura-Algarve, Portugal, 2012, pp. 5026-5033, doi: 10.1109/IROS.2012.6386109. keywords: Engines;Optimization;Computational modeling;Heuristic algorithms;Dynamics;Mathematical model,

[6] Unitree Robotics, "unitree_mujoco" GitHub repository. https://github.com/unitreerobotics/unitree_mujoco Accessed: Dec. 1, 2025.

[7] Design and Control of a Bipedal Robotic Character https://la.disneyresearch.com/publication/design-and-control-of-a-bipedal-robotic-character/ Accessed: Dec. 1, 2025.

[8] DTC: Deep Tracking Control, ETH Zürich https://www.science.org/doi/10.1126/scirobotics.adh5401

[9] Video of DTC: Deep Tracking Control, ETH Zürich https://www.youtube.com/watch?v=nipH-yl8lR0

[10] G. D. G. Pedro, G. Bermudez, V. S. Medeiros, H. J. d. C. Neto, L. G. D. d. Barros, G. Pessin, M. Becker, G. M. Freitas, and T. Boaventura, "Quadruped Robot Control: An Approach Using Body Planar Motion Control, Legs Impedance Control and Bézier Curves," Sensors, vol. 24, art. no. 3825, 2024. doi: 10.3390/s24123825. [Online]. Available: https://doi.org/10.3390/s24123825