Data Driven Design and Optimization of an Energy-Efficient Jumping Robot

Ali Fuat Sahin
alifuat.sahin@epfl.ch
Student ID: 374463

Jade Therras
jade.therras@epfl.ch
Student ID: 316645

Davide Lisi
davide.lisi@epfl.ch
Student ID: 375197

Antonio Ruiz Senac
antonio.ruizsenac@epfl.ch
Student ID: 327000

Abstract

This project focuses on the data-driven optimization and fabrication of a jumping robot designed to maximize jumping distance while minimizing energy consumption. Using a configurable simulation environment in PyBullet, we investigated the effects of varying key parameters, including link lengths, spring stiffness, and compression, on the robot's performance. Through iterative simulations and optimization algorithms, we identified an optimal set of parameters that achieved the desired balance between jumping distance and energy efficiency. The results were validated through the fabrication of a physical prototype, demonstrating the practical applicability of our optimized design. This study showcases the potential of combining simulation-based optimization with real-world implementation in robotic design.

1 Introduction

1.1 Motivation

In the realm of robotics, efficient and agile locomotion remains a central challenge, particularly for robots intended to navigate complex and varied environments. Jumping robots, with their ability to overcome obstacles and cover significant distances quickly, present a promising solution. However, designing such robots involves navigating the intricate balance between maximizing jump distance and minimizing energy consumption. Traditional design approaches often rely on trial and error, which can be time-consuming and inefficient. This project is motivated by the need for a systematic, data-driven method to optimize the design parameters of jumping robots, thereby enhancing their performance and efficiency.

1.2 Objectives

The primary objective of this project is to optimize the design of a jumping robot to achieve maximum jumping distance with minimal energy expenditure. To this end, we implemented a configurable simulation environment using PyBullet, a robust physics engine for simulating robotic dynamics. Our specific objectives are:

1. **Parameter Exploration**: To systematically investigate the effects of varying key design parameters—link lengths, spring stiffness, and compression—on the robot's jumping performance.
2. **Optimization**: To employ iterative simulation and optimization algorithms to identify the optimal set of parameters that maximize jumping distance while minimizing energy usage.
3. **Fabrication and Validation**: To fabricate a physical prototype based on the optimized design and validate the simulation results through experimental testing.
4. **Performance Analysis**: To analyze and compare the simulated and experimental results, ensuring the practical applicability and reliability of the optimized design.

By achieving these objectives, this project aims to demonstrate the potential of combining simulation-based optimization with real-world implementation in the design of efficient jumping robots. The insights gained from this study could contribute to advancements in robotic locomotion, paving the way for more agile and energy-efficient robots in various applications, from search and rescue missions to planetary exploration.

1.3 Related Work

The design and optimization of jumping robots have been an active area of research, with various approaches being explored to enhance performance and efficiency. Previous studies have demonstrated different strategies for optimizing robotic locomotion, primarily focusing on mechanisms, materials, and control algorithms.

We have investigated different types of jumping robots designed in the literature. Li et al. [2012] presents a bio-inspired jumping robot design and its analysis of jumping performance based on a biological model. Xu et al. [2023] presents a miniature, bio-inspired jumping robot with a Stephenson six-bar mechanism. Lastly, Kovac et al. [2008] presents the development and characterization of a versatile miniature jumping robot with a very simple design. Additionally, Kovač et al. [2010] improves the stability and steerability of the jumping robot design from the previous study.

Our work is closest to Kovac et al. [2008]. Because of its simple design, it is easy to analyze, configure, and upgrade depending on the application.

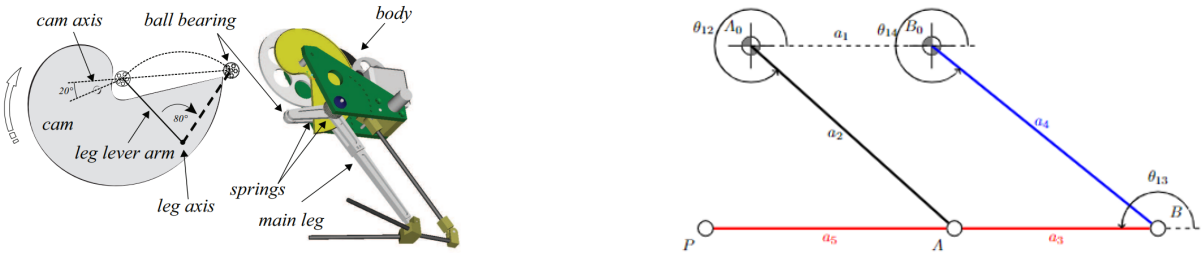
2 Problem Setup

To implement our data-driven approach, we first created a plan to guide the optimization of the jumping robot. We used a process that combined simulation and physical prototyping. We started by understanding the robot's mechanism and building an initial prototype with arbitrary parameters. This prototype bridged the gap between reality and simulation. We then used Principal Component Analysis (PCA) to identify the most influential factors from the data. Afterward, we set up a simulation to optimize the explored parameters using data-driven methods, to maximize the jumping distance and minimize energy consumption. We utilized genetic algorithms to explore the parameter space and Bayesian optimization for fine-tuning the results. Finally, we compared the optimized parameters with those of a second physical prototype to assess how well the simulation results translated to reality.

2.1 Design and Working Principle

As mentioned previously, the design of the robot was influenced by Kovac et al. (2008). The robot features a three-link leg design, which can be compressed using a cam element pushing down on the leg. This cam is powered by an electric motor located on the body, coupled with a series of gears. As the cam pushes the body, it compresses a spring connected to both the backlink A and the body. A diagram illustrating the general structure of the robot can be seen in Figure 1. Furthermore, the mechanism can be observed in Figure 2, which showcases our robot explained in a later section.

Another important step to understanding the mechanism was understanding the movement. A kinematic model was created tracing the speed and position of every link relative to each other, resulting in the link paths that can be seen in figure 1b.



(a) Structure of robot that inspired Kenan 1.0

(b) Kinematic diagram of links.

Figure 1: Jumping phases in CAD model

The robot's mechanism is fairly simple, but calculating the jumping distance becomes complex due to the various factors that affect it. These factors include friction within the gears, springs, cam and joint, as well as the energy stored in flexible parts. Using a data-driven approach involving simulation, we can effectively account for these factors. Additionally, we can test multiple values for our key parameters.

2.2 Mathematical Model

In this problem, we are examining a jumping robot mechanism that consists of five different link lengths, a link angle for the V-shaped link, spring stiffness, the robot's rest angle (when the spring is not compressed), and compression angle.

Our mathematical model includes a kinematic loop where we enforce a constraint that links a_4 must be parallel to a_2 (see Figure 1b). Through our kinematic analysis using Excel, we observed significant distortion in the mechanism's motion when these two links are not parallel.

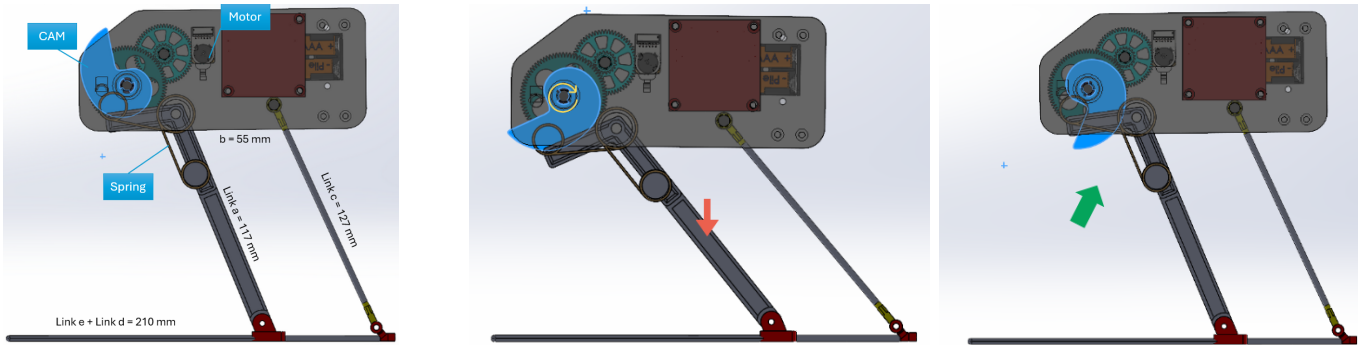
To incorporate this constraint into the model, we parameterized the lengths of each link except link a_0, which is calculated based on the lengths of the other links. The parameterization involves defining a coefficient for each link, which, when multiplied by the length of the force-carrying link (a_2), determines the actual length of the link. This approach helps avoid singular mechanism designs in the optimization process.

Likewise, the compression angle is dependent on the rest angle and is also parameterized by a coefficient, as the compression angle cannot exceed the rest angle.

Our mathematical model accurately captures these constraints and parameterizations, allowing us to simulate and optimize the jumping robot's performance effectively.

3 First Prototype

The model of the first prototype is shown in Figure 2a. The jump of this prototype happens in three stages, as depicted in Figures 2. It is important to mention the following design choices.



(a) Charging phase. (b) Jumping phase. (c) Jumping phase.

Figure 2: Jumping phases in CAD model

- The CAM element, as the one in Kovac et al. [2008], has been introduced to have repeatable jumping conditions, such as angular rest angle and angular compression for the torsional spring.
- Two torsional springs are connected to the body plates and link (a). This is important to have stability during the charging and jumping phases.

3.1 Fabrication

The first prototype, Kenan 1.0, was assembled using a DC motor with reduction, a microcontroller, a motor controller and 3D printed body plates (in PLA), gears, links and CAM element (PETG). Several obstacles were encountered during the fabrication process, such as setting tolerances and PETG weaknesses to extension stress.

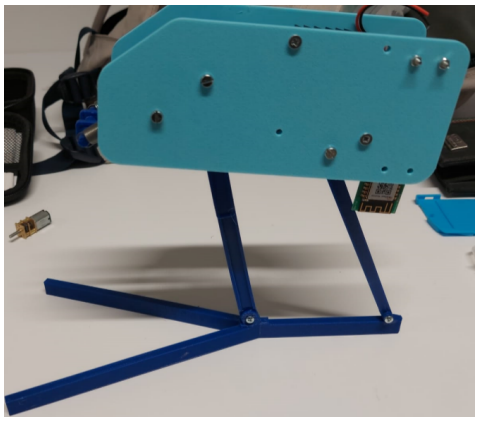


Figure 3: *Kenan 1.0*, first prototype.

3.2 Performance

After completing the prototype, we conducted the first jump attempts. We took five measurements of the jumping distance of the robot's centre of mass into account.

Kenan 1.0 achieved an average jumping distance of 25 cm. The robot's movement during the jump and the jump distance were used to tune the simulation parameters in the next step.

4 Simulation

The simulation environment was implemented using PyBullet, a physics engine widely used for simulating robotic dynamics. PyBullet provides a robust platform for testing and optimizing robotic designs due to its accurate physics modeling and real-time simulation capabilities. Our simulation setup allowed for the adjustable configuration of various robot parameters, including link lengths, spring stiffness, and compression.

In addition to these primary parameters, the simulation includes several hyperparameters that significantly affect the performance and accuracy of the simulation.

These hyperparameters were initially tuned based on the prototype we created. By comparing the simulation results with the actual performance of the prototype, we were able to calibrate these hyperparameters to obtain desirable simulation accuracy before proceeding with the optimization process.

Hyperparameter	Value
Lateral Friction	0.8
Spinning Friction	0.1
Rolling Friction	0.01
Joint Damping	0.001
Coefficient of Restitution	0.8

Table 1: simulation's hyperparameters

In the simulation, the mass and inertia of the links and the bodies are parametrized and changed depending on the thicknesses and lengths of the links. The torque from the torsional spring is not simulated using the soft body simulation given in the PyBullet environment since it reduces the accuracy of the results and increases the time complexity to run the simulation. Therefore, torque from the torsional spring is calculated at each time step and applied to the respective joint in the simulation.

To further decrease the simulation time, we modelled the main body using its mass and inertia parameters instead of loading an STL file for each simulation run. This approach maintained the accuracy of the simulation as long as consistent inertia and mass parameters for the main body were applied.

We implemented the `createMultiBody` function of PyBullet to create a fully reconfigurable simulation. This function allows for the dynamic creation and configuration of robot models, which can be adjusted within a for loop during the optimization process by changing the parameters.

With the final adjustment, the simulation corresponding to Kenan 1.0 achieves a jump of 10 cm height and 25 cm length, with an energy of 0.153J (see Table.3.

5 Parameter Exploration

To explore the impact of different design parameters on the jumping performance of the robot, we conducted a series of simulations varying one parameter at a time while keeping others constant. The key parameters investigated include:

- **Link Lengths**: The lengths of the robot's limbs were varied to determine the optimal configuration for maximum jump height and distance.
- **Spring Stiffness**: The springs' stiffness in the robot's legs was adjusted to study its effect on energy storage and release during jumps.
- **Compression**: The initial compression of the springs was varied to analyze its influence on the jump initiation and overall energy efficiency.

We implemented Principal Component Analysis (PCA) to understand the variance in our parameter space. However, the PCA results indicated that almost all components had equal importance, suggesting that no significant dimensionality reduction was possible. Therefore, we continued to explore all parameters in the optimization process. Variance explained with the principle components can be seen in Figure 4.

Each parameter set was tested multiple times to account for variability and ensure reliable results. The performance metrics recorded included jump height, jump distance, and energy consumption.

Table 2 summarizes specifically which parameters we've changed, as well as their upper and lower bounds.

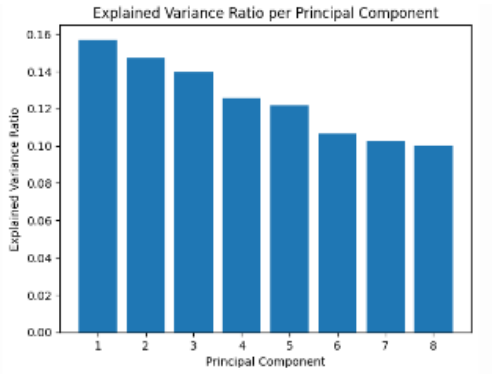


Figure 4: Explained variance of each principal component.

Key Parameter	Lower Bound	Upper Bound
Carrying Link Length	10 [cm]	20 [cm]
Compression Ratio	5 [deg]	30 [deg]
Rest Angle	20 [deg]	70 [deg]
Spring Stiffness	5 [N*mm/deg]	30 [N*mm/deg]
Ground Link Angle	10 [deg]	80 [deg]
Link (c) coefficient	0.8	1.2
Link (d) coefficient	0.5	0.9
Link (e) coefficient	0.6	1.4

Table 2: Key parameters and their bounds

6 Data-driven Optimization

The optimization step has two parts. First, genetic algorithms have been used to explore the parameter space and extract promising robot configurations. Secondly, bayesian optimization is applied to start from the best configurations, exploring single parameter change or the local parameter space for fine-tuning. All code has been made in Python.

6.1 Objective Function

The optimization process focuses on maximizing the robot's jumping distance while considering the energy consumption. To achieve this, we have defined an objective function shown in Eq. (1) where L_{max} represents the maximum length of all the robot's links and D represents the jump distance. Energy consumption is taken into account by selecting the best-performing robots in the genetic algorithm step and then selecting the most energy-efficient ones for the Bayesian optimization step. Energy is calculated using Eq. (2), where k is the spring stiffness and $\delta_{compression}$ is the compression angle.

$$fitness = \frac{D}{L_{max}} \quad (1) \qquad E = \frac{1}{2} k * \delta^2_{compression} \quad (2)$$

6.2 Genetic Algorithm Optimization

Genetic algorithms were used in combination with PyGAD Fawzy Gad. We performed numerous iterations, each consisting of 5000 generations with 10 individuals per generation, while experimenting with different crossover fractions and types and numbers of mutations. The algorithm aimed to optimize the 8 parameters specified in the range outlined in Table.2. The best result from each iteration was saved.

The performance of the best results is shown in Table.3, and the robot's geometry in Figure 5 revealed a jump height of 89.1 cm, which is more than triple the result of the initial robot. The jump was also four times higher.

In terms of parameters, the spring stiffness tended towards the boundaries, increasing from 9.2 N*mm/deg in the initial robot to 29.4 N*mm/deg. This resulted in higher energy. The resting angle became slightly sharper, as did the link angle. The compression ratio increased by a factor of 2. All link lengths tended to become shorter. The length of the feet decreased, while the length of the link of the body in contact with the ground during the jump increased.

6.3 Bayesian Optimization

Bayesian optimization has been implemented using the Scikit-learn Gaussian Process Regressor scikit-learn developers. Two types of fine-tuning are applied: one with a high number of generations (100) and simultaneous tuning of 1 to 2 parameters over a broad range, and another with a low number of generations (10) and tuning of all parameters within a small range around the optimized starting position.

After multiple iterations, the best result (Table. 3) is not significantly different from the one obtained with the genetic algorithm. The jump is slightly longer, higher, and costs a little less energy. The link lengths are very similar. Interestingly, the biggest change is the link angle, which returned to its initial value, and the resting angle is slightly less acute.

6.4 Sim to Real Gap

Even if all 8 parameters have been optimized, some of them can't be changed easily due to material and time constraints. In our case, we were not able to change the spring since it was not available in the lab. Therefore, to validate our findings, it has been decided to only alter the link parameters to achieve the best result.

The final geometry can be observed in Figure 5. Interestingly, although it appears highly similar to the first robot, the simulation results indicate that it jumps about 7 cm longer, 2.5 cm higher, and uses slightly less energy than the first robot.

7 Result

Simulation / Reality	Jump Height	Jump Length	Jump Energy
Kenan 1.0	10 cm	25 cm	0.153J
Kenan 1.0	15 cm	25 cm	$\approx 1.5J$
Best GA simulation	38.6 cm	89.1 cm	0.73J
Best BO simulation	38.9 cm	90.5 cm	0.72J
Kenan 2.0	12.6 cm	32.1 cm	0.133J
Kenan 2.0	12 cm	35 cm	$\approx 1.3J$

Table 3: Robot performance parameters. Precise parameters can be found in the file "Solution.txt" on the git repository.

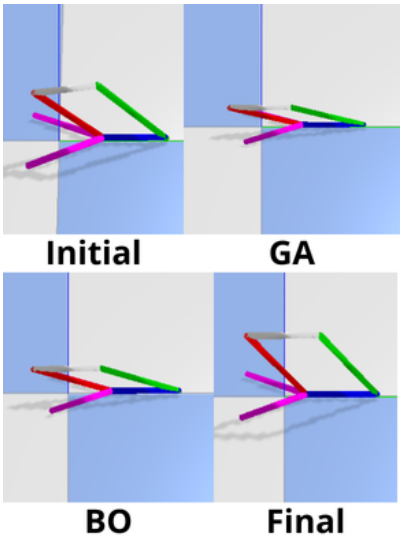


Figure 5: Geometries of the best performance robots.

8 Conclusion

The findings emphasize that the spring stiffness and compression ratio are the most crucial factors. However, optimizing the link length and angle alone can significantly enhance the robot's efficiency.

By adjusting the link length alone, the jump distance has increased by 10 cm while also making the robot more energy-efficient. The transition from simulation to reality is promising and suggests even better results if more parameters are optimized. The best overall outcome leads to jumps 3.5 times longer and 4 times higher.

9 Partnership work and resources

All tasks were completed through collaboration, with regular group meetings aiding progress. For all project-related files, please refer to our GitHub repository at `https://github.com/alifuatsahin/Jumping-Robot-Design`.

References

Ahmed Fawzy Gad. PyGAD - python genetic algorithm! — PyGAD 3.3.1 documentation. URL `https://pygad.readthedocs.io/en/latest/`.

Mirko Kovac, Martin Fuchs, André Guignard, Jean-Christophe Zufferey, and Dario Floreano. A miniature 7g jumping robot. In *2008 IEEE international conference on robotics and automation*, pages 373–378. IEEE, 2008.

Mirko Kovač, Manuel Schlegel, Jean-Christophe Zufferey, and Dario Floreano. Steerable miniature jumping robot. *Autonomous Robots*, 28:295–306, 2010.

Fei Li, Weiting Liu, Xin Fu, Gabriella Bonsignori, Umberto Scarfogliero, Cesare Stefanini, and Paolo Dario. Jumping like an insect: Design and dynamic optimization of a jumping mini robot based on bio-mimetic inspiration. *Mechatronics*, 22(2):167–176, 2012.

The scikit-learn developers. GaussianProcessRegressor. URL `https://scikit-learn/stable/modules/generated/sklearn.gaussian_process.GaussianProcessRegressor.html`.

Yi Xu, Yanzhou Jin, Weitao Zhang, Yunhao Si, Yulai Zhang, Chang Li, and Qing Shi. Design and optimization of a miniature locust-inspired stable jumping robot. *IEEE Robotics and Automation Letters*, 8(8):4673–4680, 2023.