31st International Conference
ENGINEERING MECHANICS 2025
Medlov, Czech Republic, May 12 –14, 2025
Paper **#54**, pp. 181–184, doi: 10.21495/em2025-181

SIMULATION-IN-THE-LOOP OPTIMIZATION OF OBSTACLE-AIDED LOCOMOTION IN SNAKE-LIKE ROBOTS

Shehadeh M. A.[1]

Abstract: *This study investigates the optimization of obstacle-aided locomotion in snake-like robots through a simulation-in-the-loop framework. A dynamic model of the robot is developed in CoppeliaSim to evaluate contact forces interactions during navigation in environments with varying obstacle densities. These interactions are used to optimize the robot's motion trajectory with minimal actuator energy while penalizing contact forces. The robot's motion is parametrized by a set of serpentine gait parameters, optimized using Particle Swarm Optimization to exploit beneficial obstacle contacts. Results reveal a trade-off between obstacle utilization and energy efficiency, demonstrating the potential of obstacle-aware trajectory planning for enhanced locomotion performance.*

Keywords: 3D CoppeliaSim,SerpentineGait,Snake-likeRobot,Obstacle-aided

1. Introduction

Bio-inspired robotics has developed over the past few decades as a challenging topic, drawing inspiration from the fascinating adaptability and efficiency seen in animal locomotion. Snake-like robots alone have attracted considerable attention due to their ability to perform a variety of complex gaits such as lateral undulation (serpentine locomotion), sidewinding, and concertina motion (Bae et al., 2020).

One particular interesting feature of biological snakes is their ability to exploit roughness and objects in the terrain to enhance locomotion, see (Sanfilippo et al., 2017). They use rocks, branches, or narrow passage walls as anchoring points to propel themselves forward efficiently. Emulating this strategy, I investigate a snake-like unmanned ground vehicle (UGV) performing serpentine gait with an aim to exploit obstacle-aided locomotion for optimizing energy efficiency with a trade-off of reduced wear of the robot's body.

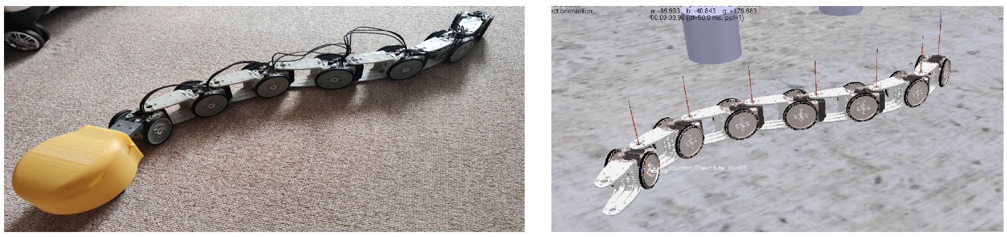

Fig. 1: The physical snake-like robot (left) and its simulation twin model in CoppeliaSim (right)

I simulate a planar robot in CoppeliaSim software, modelled to be the digital twin of a real physical prototype, see Fig. 1. The simulation includes environmental frictional effects and assumes known obstacle positions. To focus on motion planning, perception is simplified by assuming a priori knowledge of obstacle geometry and mass. In practice, this information could be obtained from touch sensors, pressure sensors, or aerial imagery.

[1] Ing. Mhd Ali Shehadeh: Institute of Automation and Computer Science, Brno University of Technology, Technická 2896/2; 616 69, Brno; CZ, Mhd.Ali.Shehadeh@vutbr.cz

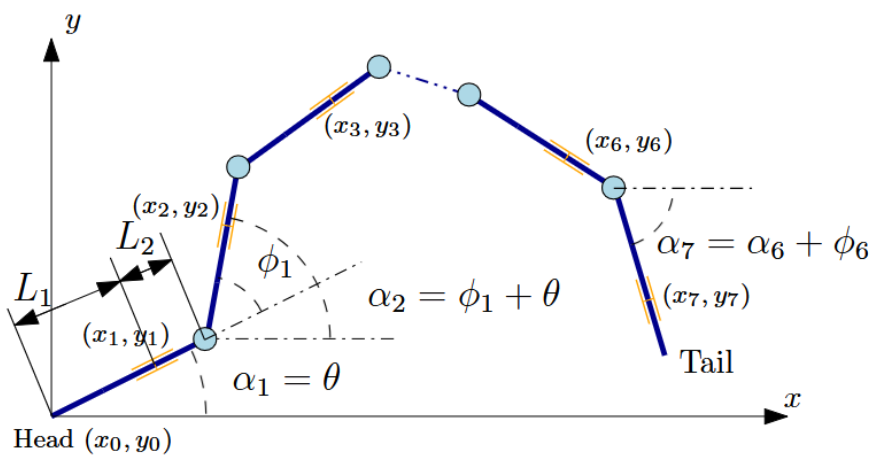

Fig. 2: Schematic of the 7-link snake-like robot, showing the nine generalized coordinates used in the dynamic model

The core objective of this work is to optimize the robot's path by selecting a series of midpoints through which the snake must pass, such that it can use obstacles for propulsion at greater speed while minimizing actuator effort and body corrosion. I frame this as an optimization problem solved by using the metaheuristic algorithm Particle Swarm Optimization (PSO).

2. Mathematical Modeling of a Multi-Link Snake-Like Robot

The mathematical model of a 7-link kinematic chain can be formulated in a similar way to the approach proposed in (Shehadeh et al., 2024), see Fig. 2 in which I use nine generalized coordinates:
$q = (x_0, y_0, \theta, \phi_1, \phi_2, \phi_3, \phi_4, \phi_5, \phi_6)^T$, which resembles the snake-head's position and the six relative angular displacements at each joint. The absolute displacement for each link, designated by α_i, can be simply described as $\alpha_{i+1} = \alpha_i + \phi_i$. The initial global displacement α_1 corresponds solely to the head's orientation θ.

Let (x_i, y_i) denote the coordinates of the wheels' CM, the positions of the wheels can then be expressed as:

$$x_i = x_{i-1} + L_1 \cos(\alpha_i) + L_2 \cos(\alpha_i), \quad i = 1, 2, ..., 7, \tag{1}$$

$$y_i = y_{i-1} + L_1 \sin(\alpha_i) + L_2 \sin(\alpha_i), \quad i = 1, 2, ..., 7. \tag{2}$$

Under the assumption that side wheels enforce a non-holonomic constraint that prevents lateral slipping, I can derive the constraint matrix (Pfaffian matrix) $\mathbf{A}(\mathbf{q})$, where $\mathbf{A}(\mathbf{q}) \cdot \dot{\mathbf{q}} = 0$, (Shehadeh et al., 2024).

To derive the dynamical system of the robot, I apply Lagrange's equation under these constraints:

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}}\right) - \frac{\partial \mathcal{L}}{\partial q} = \mathbf{Q} + \mathbf{F_o}, \tag{3}$$

where $\mathbf{F_o}$ represents external forces applied by obstacles at contact points with the snake, these forces are modelled using a compliant contact model: $F_{oj} = k_p \delta_j + k_d \dot{\delta}_j$. The reduced dynamical system, without applying any external force, can be concisely formulated as:

$$\dot{\mathbf{q}} = \mathbf{G} \cdot \dot{\phi}, \tag{4}$$

where $\dot{\phi} = (\dot{\phi}_1, \dot{\phi}_2, ..., \dot{\phi}_6)^T$ and $\mathbf{G}$ is the control matrix defined by:

$$\mathbf{G} = \begin{bmatrix} -\mathbf{B}^{-1}(q) \cdot \mathbf{C}(q) \\ \mathbf{I_{6\times 6}} \end{bmatrix}, \tag{5}$$

which is derived from the Pfaffian Matrix $\mathbf{A}$. Here, $\mathbf{A}(\mathbf{q})$ is partitioned into $[\mathbf{B}(q) \mid \mathbf{C}(q)]$ where $\mathbf{B}(q)$ relates to the head's pose, and $\mathbf{C}(q)$ to the joint motions.

For locomotion in idealized terrain, assuming absolute traction of the side wheels, the robot can easily follow a serpentine curve, as described by Hůlka et al. (2020): $\phi_i(t) = \mu \cdot \sin(\omega t + (i-1) \cdot \beta) + \gamma$, where ω is a constant angular frequency, β is the phase offset, and γ is a baseline offset.

In reality, side-way slipping may occur when contact surfaces can not endure the applied friction force, so the robot can make advantage of anchors in the environment to propel against. The kinematics of the system are modified to include η_i, an additional offset that accounts for reaction forces from the obstacles:

$$\phi_i(t) = \mu \cdot \sin(\omega t + (i-1) \cdot \beta) + \gamma + \eta_i, \tag{6}$$

The reaction force contributes to forward motion, modifying the velocity equation:

$$v_x = \frac{1}{N} \sum_{i=1}^{N} L_i \dot{\alpha}_i \cos \alpha_i + \sum_{j=1}^{M} F_{oj} \cos \psi_j, \tag{7}$$

where ψ_j is the angle between the direction of the contact force and the robot's forward direction.

3. Simulation Implementation

The digital twin is replicated to match key parameters of the physical model such as mass, inertia, and wheel constraints. Due to the ease of implementing contact detection and sensor feedback in simulation, I use it as an efficient testbed to evaluate obstacle's reaction forces in the x, y, and z directions, then optimize control strategies, see Fig. 1. I use dynamic engine bullet 2.78 with friction factor of the floor equals to 0.5.

4. Control and Optimization Methodology

The goal of this optimization is to minimize energy spent at the joint actuators by utilizing obstacle reaction forces, while ensuring minimal deformation of the robot body. The objective function is defined as:

$$J = \sum_{t=1}^{T} \left(v_x - \zeta \sum_{j=1}^{M} |F_{oj}| \right), \tag{8}$$

where ζ balances forward velocity and contact penalty.

The overall methodology integrates the PSO algorithm implemented in MATLAB with the dynamic simulation of the snake-like robot in CoppeliaSim. The robot's path is discretized in N_p points that the robot will go through. Each individual in the PSO population represents a candidate set of N_p configurations of the snake-like waypoints along with corresponding serpentine parameters that will drive the robot through these waypoints.

For each candidate, MATLAB computes the motion parameters, which are sent to CoppeliaSim via its remote API. The simulation runs, recording contact forces through distributed sensors at obstacle interaction points. Sensor outputs (F_{oj}) are returned to MATLAB, where the fitness of each individual is evaluated. The PSO then updates population, iteratively refining motion parameters. This simulation-in-the-loop framework enables pre-planned optimization of locomotion by leveraging simulated obstacle interactions to refine movement efficiency before deployment.

5. Results and Discussion

To evaluate the proposed methodology, the robot is tested with traversing a 10-meter path over three different scenes, high obstacle density, moderate, and sparse, see Fig. 3. For each scene four levels of obstacle utilization are used: High Utilization: $\zeta = 0.01$, Medium Utilization: $\zeta = 0.3$, Low Utilization: $\zeta = 0.6$, and Obstacle Avoidance: $\zeta = 10^3$. The simulation-in-the-loop run for a thousand iteration and the Total Energy Consumption and Avg. Contact Force of optimal candidates are recorded in Tab. 1. The results reveal a consistent trade-off between energy efficiency and contact-dependence intensity. In sparse environments, obstacle utilization offers limited benefit due to fewer interaction opportunities. In contrast, intermediate and dense environments show clear gains in energy savings.

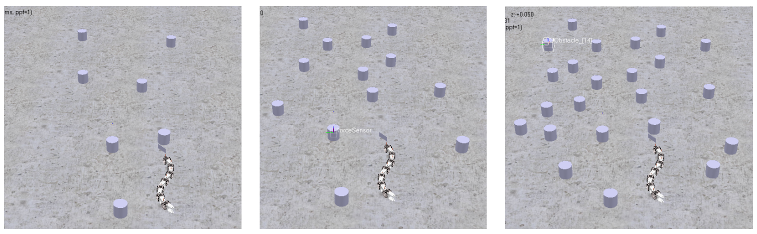

Fig. 3: The three scenes used in this test, obstacles are distributed randomly from the start to the end point, the endpoint is not shown here, since the image illustrates only a quarter of the full scene.

Peak contact forces occur in dense–high-utilization scenarios, but are offset by notable efficiency gains. Medium utilization offers a balanced performance, reducing energy while maintaining moderate contact levels.

6. Conclusion

This study presents a simulation-in-the-loop optimization framework for contact-aided locomotion in snake-like robots. By integrating Particle Swarm Optimization with a dynamic simulation model, I demonstrate that obstacle interactions can be systematically leveraged to reduce actuator energy while maintaining minimal wear to the robot body. Results across varying scenes show that medium to high obstacle utilization significantly enhances performance, particularly in cluttered environments. These findings serve as a first step to unlock the potential of pre-execution optimization to generate adaptive, efficient motion strategies that exploit environmental features rather than avoid them.

Acknowledgments

The author would like to express his sincere gratitude to the Institute of Automation and Computer Science to sponsor this work. Special thanks go to ChatGPT for its invaluable assistance in writing the "Conclusion and abstract" section of this paper, ensuring a clear and effective summary of my work.

References

Bae, J., Kim, M., Song, B., Jin, M., and Yun, D. (2020) Snake robot with driving assistant mechanism. *Applied Sciences*, 10, 21.

Hůlka, T., Matoušek, R., Dobrovský, L., Dosoudilová, M., and Nolle, L. (2020) Optimization of snake-like robot locomotion using ga: Serpenoid design. *MENDEL*, 26, 1, pp. 1–6.

Sanfilippo, F., Azpiazu, J., Marafioti, G., Transeth, A. A., Øyvind Stavdahl, and Liljebäck, P. (2017) Perception-driven obstacle-aided locomotion for snake robots: The state of the art, challenges and possibilities. *Applied Sciences*, 7, 4, pp. 336.

Shehadeh, M. A., Matoušek, R., Hůlka, T., and Holoubek, T. (2024) Geometric algebra modeling of snake-like robot serpentine locomotion: Preliminary study. *MENDEL*. To appear.

Tab. 1: Performance metrics across obstacle densities and utilization strategies

	Energy (J)			Avg. Contact Force (N)		
Utilization	Sparse	Intermediate	Dense	Sparse	Intermediate	Dense
Obstacle Avoidance	160.2	184.8	257.3	0	0	0.3
Low	122.6	126.7	132.6	2.4	3.9	5.9
Medium	116.1	109.5	**115.2**	4.6	6.7	**8.1**
High	**110.3**	**98.4**	124.8	**6.2**	**8.3**	9.9