Slytherdrive

Soft Snake Robot

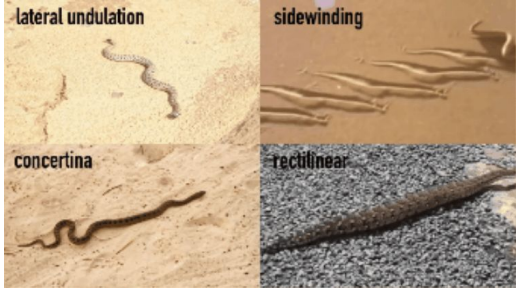
Group 24

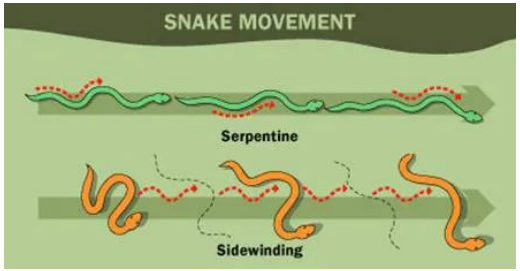
Charles Paxon, Daniel Morales, Ishaan Gupta, Aashrith Beesabathuni, David Soto Gonzalez

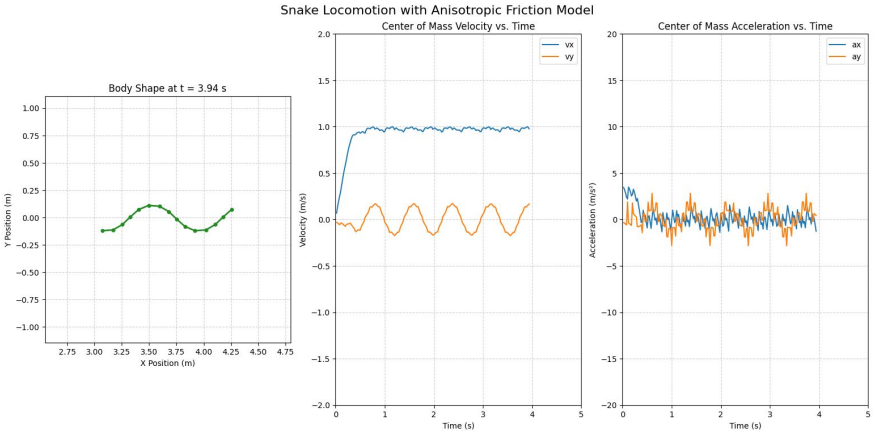
UC Berkeley Engineering

Overview of Slytherdrive

- Model and design a robotic snake that mimics the following natural snake movements:
  - **Serpentine** (wave-like bends push along the surface)
  - **Sidewinding** (body segments lift off the ground and push the rest of the body diagonally)
- Engineering Principles:
  - Rigid and continuum mechanics
  - Fluid dynamics for pneumatic controls
  - Control of a highly redundant robotic system
- Movement goes as follows: *Body Tension → Ground Reaction Force Distribution → Motion*







Dynamics of Snake Movement

How the snake's orientation changes along its length:

$$\boldsymbol{R}'(s) = \boldsymbol{R}(s)[\boldsymbol{u}(s)]$$

Body position of the snake:

$$\boldsymbol{p}'(s) = -\boldsymbol{R}(s)\hat{\boldsymbol{e}}_x$$

Curvature function of the snake's body for ideal "follow-the-leader" motion, as a function of time and path length:

$$\boldsymbol{u}(s) = \begin{bmatrix} 0 \\ 0 \\ \frac{3\pi\theta}{2L}\sin\frac{3\pi(s-v_{\tan}t)}{L} \end{bmatrix} \in \mathbb{R}^3$$

Force equilibrium & moment equilibrium, to determine how much normal force touches the ground and where:

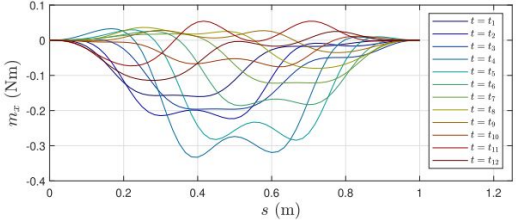
$$\int_0^L \boldsymbol{a}(s)\rho(s)ds$$
$$= \int_0^L -\mu N(s)\hat{\boldsymbol{v}}(s) + N(s)\hat{\boldsymbol{e}}_z - g\rho(s)\hat{\boldsymbol{e}}_z ds$$
$$\int_0^L (\boldsymbol{p}(s)\times\boldsymbol{a}(s))\,\rho(s)ds$$
$$= \int_0^L \boldsymbol{p}(s)\times(-\mu N(s)\hat{\boldsymbol{v}}(s) + N(s)\hat{\boldsymbol{e}}_z - g\rho(s)\hat{\boldsymbol{e}}_z)\,ds$$

Radial ground-penetration penalty & torsional bending energy, which both determine how the body sags under gravity and tension, producing:

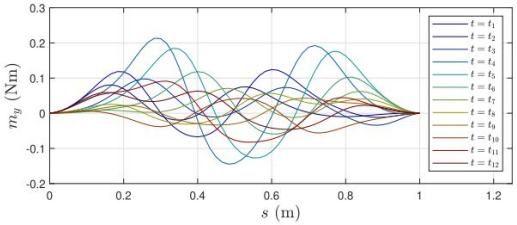
- inflection points for forward motion
- diagonally shifting contacts for sidewinding

$$\epsilon_r = \begin{cases} \frac{1}{2}k_r(\frac{d}{2} - p_z)^2 & \text{if } p_z(s) < \frac{d}{2} \\ 0 & \text{otherwise} \end{cases}$$

$$\epsilon_0 = \frac{1}{2}k_b\,(u_y - \hat{u}_y)^2 + \frac{1}{2}k_t\,(u_x - \hat{u}_x)^2 + \rho g p_z$$

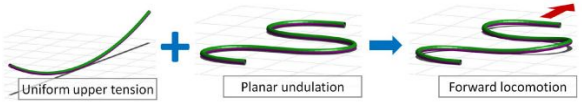


(a)

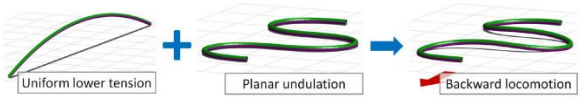


(b)

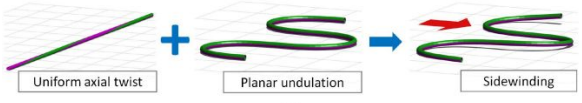
Fig. 6. (a) Torsional moments [x-component of $\boldsymbol{m}(s)$] and (b) vertical bending moments [y-component of $\boldsymbol{m}(s)$] at 12 different time instants in an undulatory cycle of sidewinding locomotion. The time instants were uniformly distributed over an undulatory cycle as $t_i = \frac{i-1}{12}\tau$.



(a)



(b)



(c)

Dynamics of Snake Movement (details)

The scales of actual snakes have anisotropic friction, that is that they have higher friction in the sideways direction than in the forward backward directions which translates a sinusoidal body motion into forward motion. Body compliance within snakes further enhance this phenomenon as it leads to an uneven distribution contact forces between the snake and ground, and also naturally leads to locomotion depending on the orientation of the body compliance. While both body compliance and anisotropic friction can independently lead to snake locomotion, we wished to see if combined both would lead to a smoother and faster snake locomotion.

The dynamics of snake movement can be modeled by treating the body as a continuously deformable rod whose orientation and shape evolve along its length. The rotation matrix **R(s)** describes how each point of the body is orientated, and its derivative **R'(s)=R(s)[u(s)]** links body curvature to internal actuation through the curvature vector **u(s)**. The position of the body centerline follows **p'(s) = -R(s)ex** meaning the snake's shape is defined by integrating orientation along its length. Using the curvature function **u(s)** an ideal "follow the leader" locomotion pattern can be prescribed, allowing different 3D backbone shapes over time, such as planar undulation, vertical bending, or axial twist. As show in Ha (2024), these internal shape changes produce spatial distributions of bending and torsional moments **m(s)**, which in turn create non uniform ground contacts essential for locomotion. The force and moment equilibrium integrals quantify how normal forces, friction, and body tension interact with these internal moments. Ground penetration penalties and torsional bending energy determine how much the body lifts or sags under gravity, creating inflection points for forward motion or diagonally shifting contact patches for sidwinding. The results can be seen in the figures where it can be seen that by combining simple uniform internal tension with a prescribed undulatory curvature can generate forward, backward, and sidewinding motion.

Our project combines these two previously demonstrated modes of snake locomotion to increase speed!

References:
Ha, Junhyoung. **"Robotic Snake Locomotion Exploiting Body Compliance and Uniform Body Tensions."** *IEEE Transactions on Robotics*, accepted for publication, 2024.

H. Chitikena, A. Mohammadi, F. Sanfilippo and M. Poursina, "Anisotropic Friction Skin for Holonomic Snake Robot Mobility*," 2024 IEEE International Conference on Advanced Robotics and Its Social Impacts (ARSO), Hong Kong, 2024,

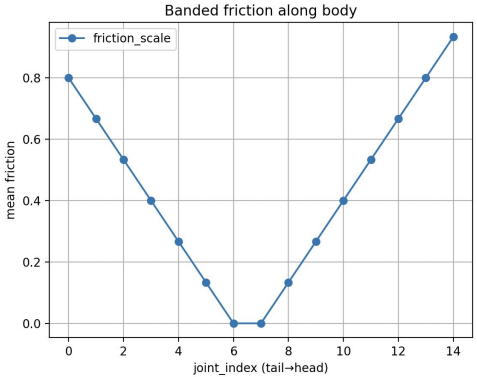
Simulation

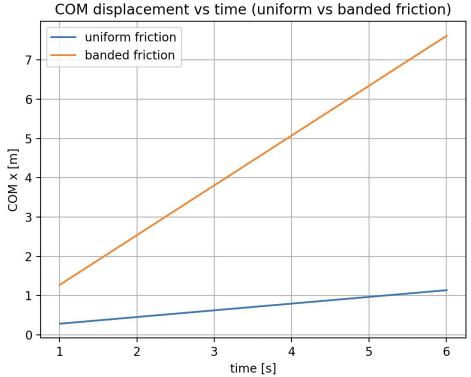
- Simulated using PyBullet Physics engine (16 link, rigid body snake)
- Proving underlying physics principle: **Anisotropic friction and uneven body tension creates movement**

*Anisotropic friction (Varying coefficients of friction laterally vs tangentially)

$$Slipratio = \frac{v_{\text{lat}}}{v_{\text{tang}}} \approx \frac{\mu_{\parallel}}{\mu_{\perp}} \qquad (\text{anisotropic friction})$$

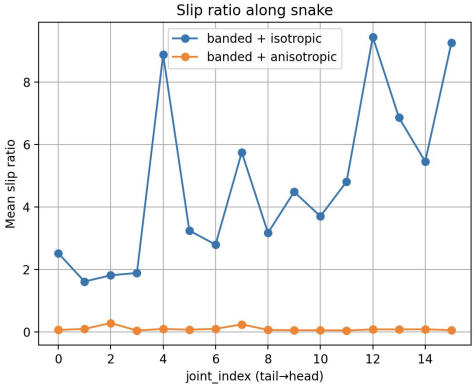
apply_band_lift() used to distribute uneven +Z force across the snake





Effect of banded friction (lower friction in center, higher friction near ends)

apply_anisotropic_friction() used to implement varying coefficients of friction across the surface



Effects of anisotropic friction

Simulation (details)

- PyBullet is a Real Time Physics simulation tool, and easily allow us to simulate robotic motion
- Steps included (snippets of code shown):
  1. Creating geometries and constraining together about the +Z
  2. Propagating a sinusoidal wave through the snake
  3. Creating model for anisotropic friction (our robot equivalent: scales)
  4. Creating model for +Z Body Tension (our robot equivalent: curved middle blocks)

1

```
snake = p.createMultiBody(
    baseMass=mass,
    baseCollisionShapeIndex=col,
    baseVisualShapeIndex=vis,
    basePosition=[0, 0, halfZ + 0.01],
    baseInertialFramePosition=[halfX, 0, 0], #COM, center of the rectangles
    linkMasses=[mass]*n_children,
    linkCollisionShapeIndices=[col]*n_children,
    linkVisualShapeIndices=[vis]*n_children,
    linkPositions=[[2*halfX, 0, 0]]*n_children,  #defines connections, each joint connected across length of rectangle
    linkOrientations=[[0, 0, 0, 1]]*n_children,
    linkInertialFramePositions=[[halfX, 0, 0]]*n_children,
    linkInertialFrameOrientations=[[0, 0, 0, 1]]*n_children,
    linkParentIndices=list(range(n_children)),    # 0->1->2->...
    linkJointTypes=[p.JOINT_REVOLUTE]*n_children,
    linkJointAxis=[[0, 0, 1]]*n_children        # hinge, allowing to only bend around the +Z axis
)
```

2

```
# SIN WAVE PROPOGATION
A = 0.9                          # rad, joint amplitude
omega = 2.0 * math.pi * 3.0 # rad/s
wavelength_links = 4.0       # phase spacing
dt = 1.0/240.0
```

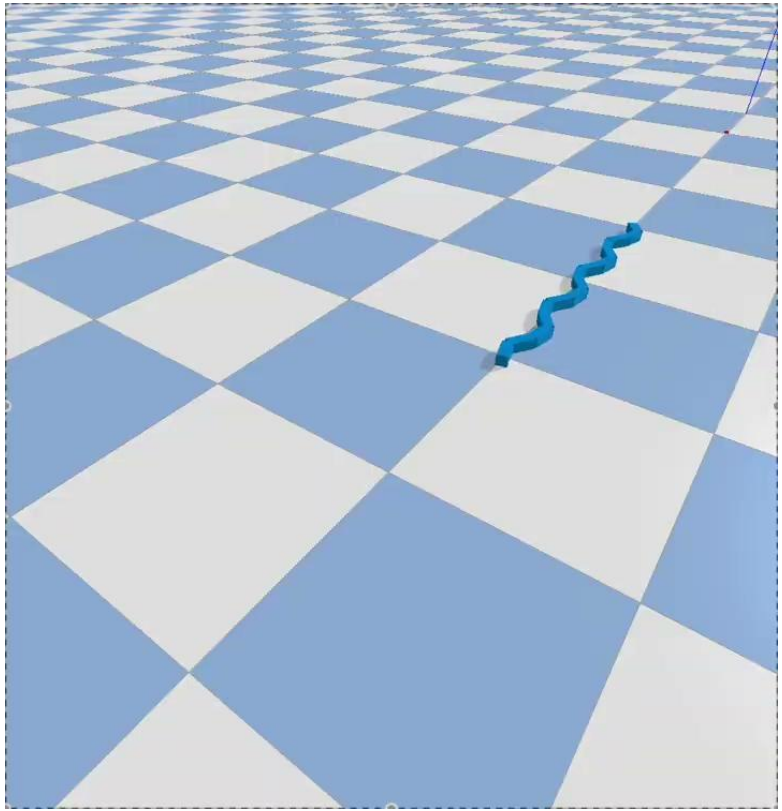
3

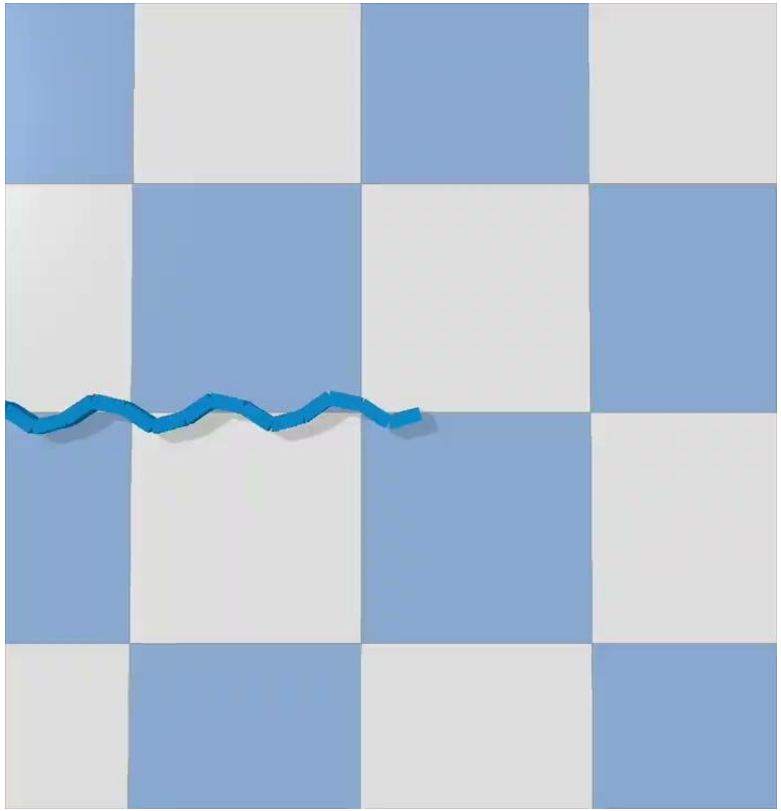
```
def apply_anisotropic_friction(body):
    zhat = (0.0, 0.0, 1.0)
    nJ = p.getNumJoints(body)
    records = []

    for link in range(-1, nJ):
        # ---- index along body: 0..n_links_total-1 ----
        if link == -1:
            idx_body = 0
            pos, orn = p.getBasePositionAndOrientation(body)
            v_lin, _ = p.getBaseVelocity(body)
        else:
            idx_body = link + 1
            ls = p.getLinkState(body, link, computeLinkVelocity=1)
            pos, orn = ls[0], ls[1]
            v_lin = ls[6]

        t_hat = normalize(quat_to_world_x_axis(orn))
        z_cross_t = cross(zhat, t_hat)
        n_hat = normalize((z_cross_t[0], z_cross_t[1], 0.0))

        v_par = v_lin[0]*t_hat[0] + v_lin[1]*t_hat[1] + v_lin[2]*t_hat[2]
        v_per = v_lin[0]*n_hat[0] + v_lin[1]*n_hat[1] + v_lin[2]*n_hat[2]

        s_par = 0.0 if abs(v_par) < vel_eps else (1.0 if v_par > 0 else -1.0)
        s_per = 0.0 if abs(v_per) < vel_eps else (1.0 if v_per > 0 else -1.0)


        mu_t = (mu_f if v_par > 0 else mu_b) if s_par != 0 else 0.0

        # normal force: deterministic profile for stability; log raw contact separately
        N_contact = estimate_normal_load(body, link)
        base_N = mass * g
        N_profile = base_N * friction_scale_for_index(idx_body, n_links_total, mid_gain=2.0, end_gain=1.0)
        N = N_profile
```

4

```
def apply_band_lift(body, Fz_mid=1.0):
    """
    Approximate an internal band:
    - middle links get +Fz force
    - end links get -Fz force
    with sum of all forces ~ 0.
    """
    nJ = p.getNumJoints(body)
    indices = [-1] + list(range(nJ))   # base + links
    n = len(indices)

    # choose which links are "middle" vs "ends"
    mid_ids = []
    end_ids = []
    for k, link in enumerate(indices):
        s = k / (n - 1)  # 0 at tail, 1 at head
        if 0.3 <= s <= 0.7:
            mid_ids.append(link)
        else:
            end_ids.append(link)

    n_mid = len(mid_ids)
    n_end = len(end_ids)
    if n_mid == 0 or n_end == 0:
        return

    # total upward force on middle
    total_up = Fz_mid * n_mid
    # choose downward per end so total is zero
    Fz_end = -total_up / n_end

    for link in indices:
        if link in mid_ids:
            Fz = Fz_mid
            if link == -1:
                pos, _ = p.getBasePositionAndOrientation(body)
            else:
                pos = p.getLinkState(body, link)[0]
        else:
            Fz = Fz_end
```

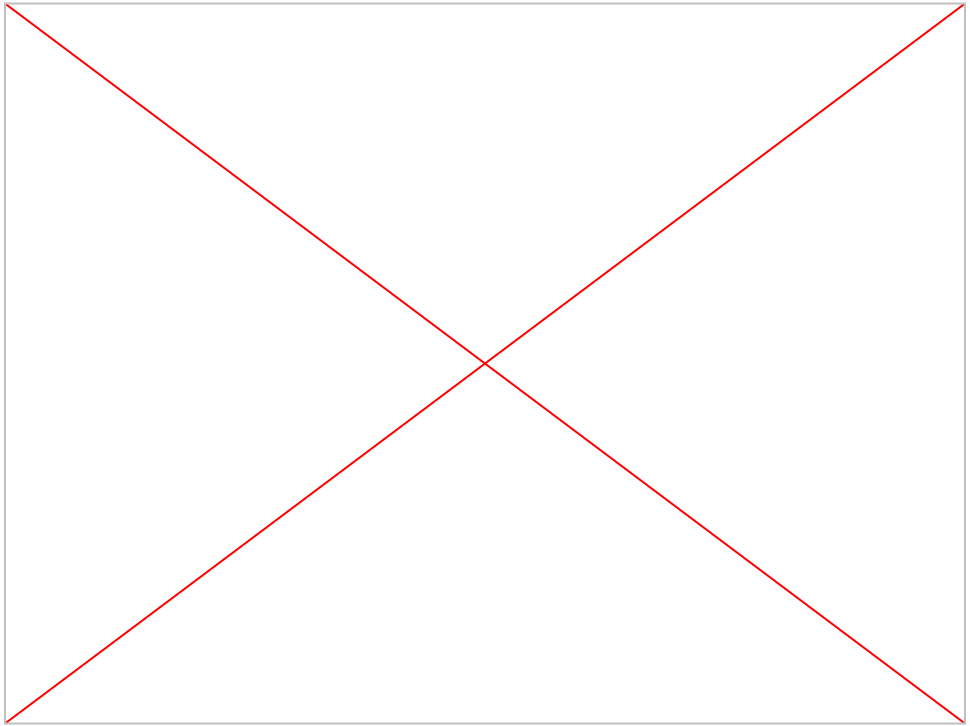
Simulation Video





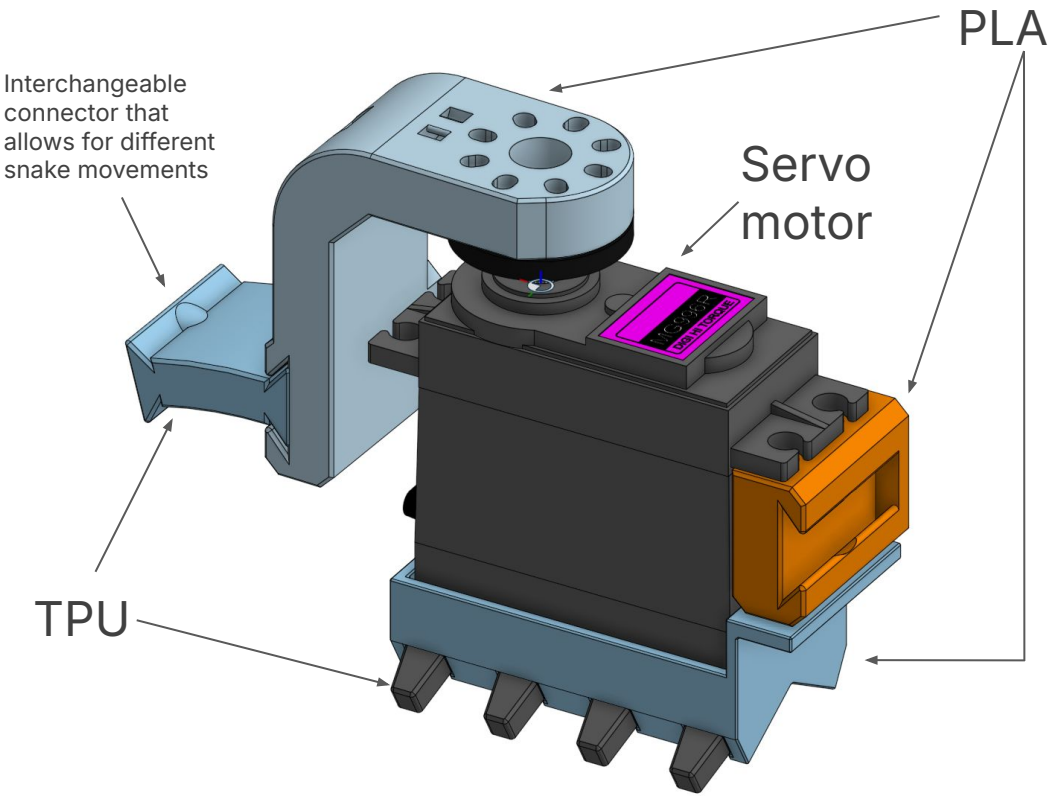
*Demonstration of final sim, proving our 2 methods of locomotion work

Simulation Video (details)

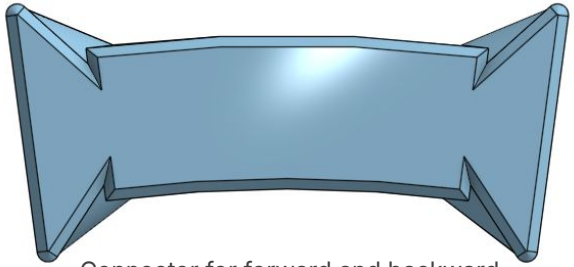


Early test, showcasing snake geometry in the physics simulator (Step 1)

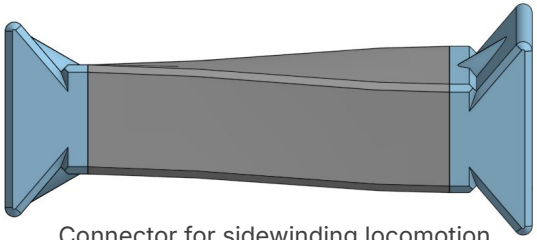
Hardware



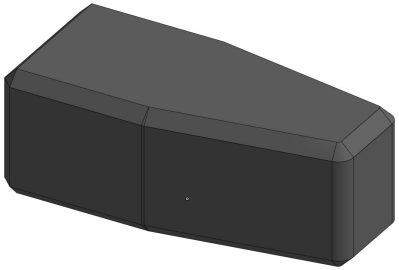
1 of 15 identical segments that connect together to make the snake



Connector for forward and backward locomotion - 165° bend along y-axis



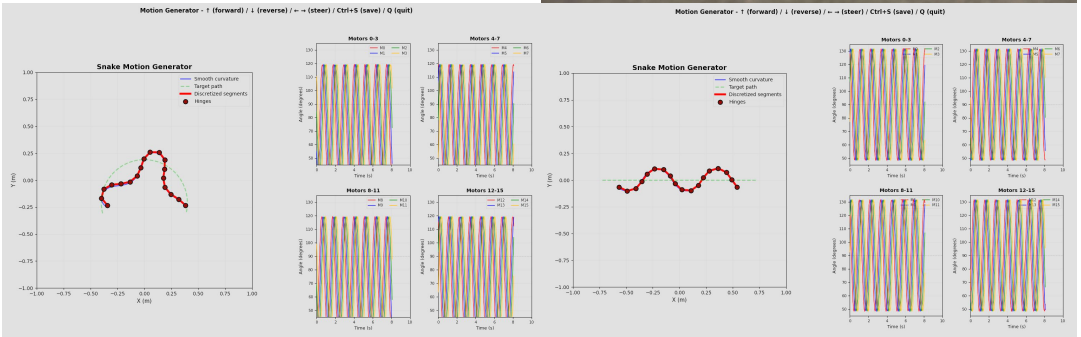
Connector for sidewinding locomotion - 6° rotation around x-axis



Scale for anisotropic friction

Hardware and Controls (Details)

The entire snake is made up of 15 segments of individual servos and 3D printed parts. The 3D printed parts consist of the arm, base, the clamp, the scales, and the connector. The connector has dovetail ends and a notch on both sides for secure connection to both the arm and the clamp while the robotic snake is moving. The connector is made of TPU filament to mimic the rubber bands motion that was shown in the paper to allow flexibility for a similar movement of a snake's.
We designed two types of connectors, one for forward and backward locomotion with a 165° bend along the y-axis, and the other is for sidewinding locomotion with a 6° bend along the x-axis, and both are interchangeable due to the dovetails.
The scales are also made of TPU filament to allow anisotropic friction, mimicking when snakes use their scales to move along a surface. The shape of the scale that would be the most efficient for the locomotion of a snake is the hexagonal shape in the reverse order configuration.
Electronics-wise, the snake is divided into 4 "worm" segments with 4 servos each, with each "worm" having a cheap ESP32-C3 microcontroller. The 4 ESP32-C3s connect wirelessly over ESP-NOW to a control ESP32, that is connected over USB to a laptop running a Python-based control software. The software displays the snake profile and joint angle traces and allows it to curve based on keyboard inputs.



Paper used as inspiration for the uneven body tension:
Ha, Junhyunyoung. "Robotic Snake Locomotion Exploiting Body Compliance and Uniform Body Tensions." *IEEE Transactions on Robotics*, 2023.
https://ieeexplore.ieee.org/document/10195887
Paper used as inspiration for the scales:
Gao, Xingda, Tong Gao, and Xiaobo Tan. "Bioinspired 3D-Printed Snakeskins Enable Effective Serpentine Locomotion of a Soft Robotic Snake." *Soft Robotics*, vol. 10, no. 3, 2023
https://www.liebertpub.com/doi/full/10.1089/soro.2022.0051#tab-citations

Hardware Video