ME 133A Final Project

Sara Razavi

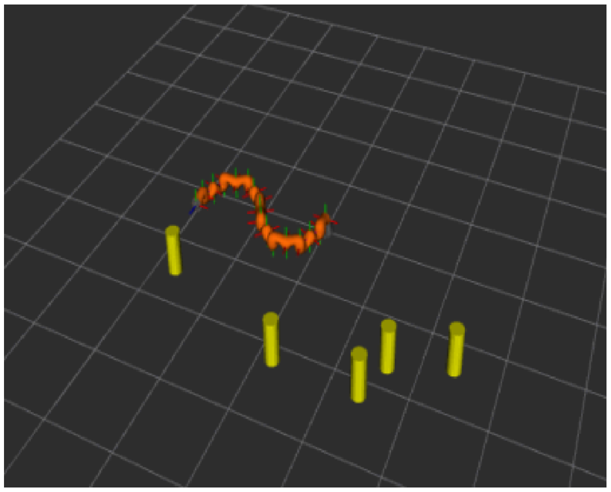
Part I: Introduction

This project implements a biologically inspired movement system for a snake robot while avoiding obstacles (fence posts). The primary goal is to simulate biologically inspired locomotion, specifically the lateral undulation (sinusoidal pattern) observed in real snakes. The robot autonomously adjusts its trajectory to avoid obstacles, ensuring smooth and efficient motion. To achieve this, we used the snake bot urdf as well as the fencedemo.py.

The robot is designed and simulated in ROS2 with a URDF model, and sinusoidal joint motions drive its movement. The requirements for this setup are:

- Precise control of joint angles to mimic realistic snake motion
- Real-time collision avoidance
- Dynamic adjustment of the trajectory to maintain progress

Figure 1: Snake Robot



Part II: Design Overview

The snake robot has 16 degrees of freedom (DOFs) and 8 pairs of joints. Each segment contributes to the lateral undulation, where opposing joint pairs move in sinusoidal patterns to mimic biological locomotion. Some key features of this include:

- Workspace: The robot operates ROS2. The snake's movement is governed by the position and orientation of its segments while keeping the base fixed. This allows for better control stability and simplified kinematics.
- Dimensionality: The task involves both spatial and temporal dimensions.
    - Spatial: The robot has 24 DOFs, corresponding to its joints. Each joint can be controlled independently, resulting in a high-dimensional control space.
    - Temporal: The movement is time-dependent, with joint angles being computed based on the current time t.
- Task Coordinates: Joint angles for each of the 24 joints, and position coordinates for the tip of the snake robot in 3D space.

- Constraints: The design assumes redundancy in motion, allowing flexibility to optimize objects by avoiding obstacles.
- Kinematics: Dynamic adjustments are made in response to obstacles, leveraging Jacobians to compute velocity and positional changes.

Part III: Task

- Main Task: The snake robot performs lateral undulation to move in a sinusoidal trajectory. This is the core biologically inspired movement that allows the robot to move. (generate sinusoidal joint angles based on time, which ensures smooth undulation)
- Secondary Task: The snake robot avoids collisions with obstacles. This involves dynamically adjusting its trajectory based on obstacle proximity to ensure smooth navigation (applying repulsive forces to avoid collisions).
- Coordination & Optimization:
    - Coordination: The system coordinates between the movement generation (sinusoidal pattern) and obstacle avoidance mechanisms. Each system operates independently but must integrate its output to achieve the overall objective.
    - Optimization: The optimization criteria is minimizing the distance to obstacles, while also maintaining the desired sinusoidal motion. This is achieved by adjusting joint angles based on repulsive forces calculated from obstacle proximity.
- Achievability of Tasks: The tasks are generally achievable under normal conditions, given that:
    - The robot has a sufficient range of motion in its joints
    - The obstacle detection system works according to what it was intended to
    - The repulsive forces are applied to adjust movement without causing instability

Part IV: Implementation

The implementation is structured around generating sinusoidal joint angles and dynamically adjusting them based on obstacle feedback.

- Pseudo-Code Overview:

```
Initialize SnakeRobotTrajectory(node)
    Set trajectory parameters (num_joints, amplitude,
frequency, etc.)
    Create subscriptions for fence positions
    Create publishers for condition numbers

Function compute_joint_angles(t):
    Initialize angles array with zeros
    For each joint index i from 1 to num_joints/2:
        Set angle_a = 0 (fixed for head)
        Calculate angle_b using sinusoidal function
        Store angles in the array
    Return angles

Function detect_and_avoid_obstacles(ptip):
    Initialize repulsion vector with zeros
    For each obstacle in fence_positions:
        Calculate distance from ptip to obstacle
        If distance < obstacle_threshold:
            Calculate repulsion vector away from obstacle
            Scale repulsion based on distance
            Apply localized bending effect on joints
    Return repulsion vector

Function evaluate(t, dt):
    Call compute_joint_angles(t) to get joint angles
    Perform forward kinematics to get ptip, R, Jv, Jw
    Call detect_and_avoid_obstacles(ptip) to get repulsion
vector
    Calculate desired velocity for movement
    Compute joint velocity using pseudoinverse of Jacobian
    Combine primary and secondary tasks (movement and
repulsion)
    Update joint angles and clip them within limits
```

- Generating Trajectories
    - The robot generates trajectories by calculating joint angles based on a sinusoidal function. This is done in the ‘`compute_joint_angles()`’ method.
    - Alternating pairs of joints have mirrored phases to create the undulation pattern
    - Joints Na (N = 1 … 8) in the snake urdf were responsible for twisting, so the angle for those joints was set to zero
- Solving Inverse Kinematics:
    - Inverse Kinematics: The robot uses forward kinematics (‘`fkin`’) to calculate the position of the tip based on current joint angles. It does not explicitly solve inverse kinematics since this operates primarily in joint space.
    - Jacobian: The Jacobian matrix is used to relate joint velocities to end-effector velocities.

- Combination of Tasks:
    - The primary task is to maintain sinusoidal motion while the secondary task is to avoid obstacles. The robot combines these objectives by adjusting joint velocities based on detected obstacles.
    - The primary and secondary tasks are combined in the `'evaluate'` method.
- Implementation Details:
    - The robot uses RVIZ for visualization through markers published on the `'/visualization_marker_array'` topic
    - The kinematic chain is defined using a URDF file that describes the robot's structure
    - The snake robot trajectory is evaluated in real-time, dynamically updating joint angles and repulsion vectors to avoid obstacles while preserving smooth sinusoidal motion.

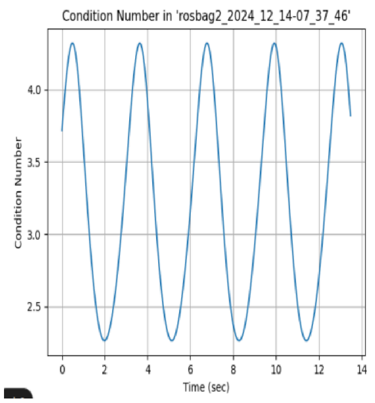
Part V: Features

- Singularities & Workspace Limits: The Jacobian condition number is monitored and published in real time to detect singularities. Adjustments are made to ensure stability
- Obstacle Avoidance: A key feature is the dynamic repulsion vector, calculated based on proximity to obstacles. This vector changes the primary velocity to prevent collisions while allowing continued motion.
- Visualization: rviz illustrates the robot's trajectory and obstacle positions. The cylindrical posts provide clear visual feedback on the robot's performance.

Part VI: Analysis

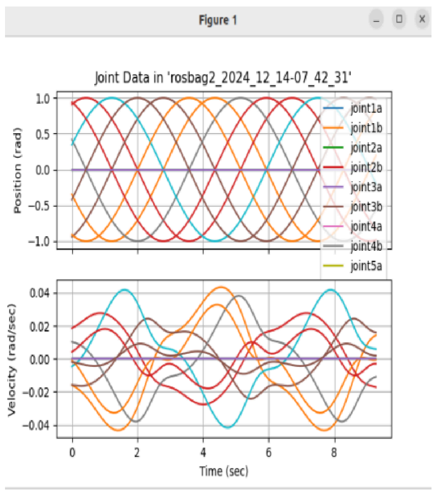
As illustrated in the graph, the periodic nature of the condition number graph suggests a consistent variation in the robot's configuration or joint angles, which is most likely due to the sinusoidal motion of the snake robot's joints. Hence, the condition number's behavior aligns with the snake robot's sinusoidal wave motion.

Figure 2: Condition Number Graph



From the joint data analysis, the joint positions exhibit smooth, sinusoidal patterns over time, which is consistent with the lateral undulating motion of the snake robot. The range of motion also shows how the joints have well-defined and consistent oscillation amplitude. Also, the joint velocity graph, follows smooth sinusoidal paths as well, except there are different phase differences for different joints, which means that there was a coordinated and sequential motion pattern. The magnitudes of the velocities are fairly small [-0.04 rad, 0.04 rad], which indicates that the robot moves at a controlled speed (there are no rapid or unstable joint motions).

Figure 3: Joint Position & Velocity Graph



Part VI: Conclusion

This project has provided a valuable opportunity in the field of robots, as it allowed me to design and implement a biologically inspired locomotion system. Through the development of a snake robot capable of such motion, I gained hands-on experience with various critical concepts, like kinematics, trajectory generation, and real-time obstacle avoidance. For more details about this project, please click on this link. This is my final project video. Moving forward, I am eager to continue refining this project. I plan to explore further enhancements, like potentially implementing machine learning techniques to enable the robot to learn from its environment and improve its navigation capabilities over time.