Part 3: Experimental Validation and Analysis

Experimental Setup: Geometric Scaling Analysis

Objective The primary objective of this experiment was to empirically validate the relationship between link length and locomotion performance. By systematically varying the geometric scale of the four-bar linkage, we aimed to observe the trade-offs between stride length, jump height, and kinematic stability.

Design Variables We evaluated three distinct robot configurations. The "Small" design served as the baseline, with subsequent "Medium" and "Large" iterations created by adding a uniform length increment (+1 cm and +2 cm, respectively) to every link in the kinematic chain.

Configuration ID Link A (Base) Link B (Crank) Link C (Coupler) Link D (Leg) Scaling Factor

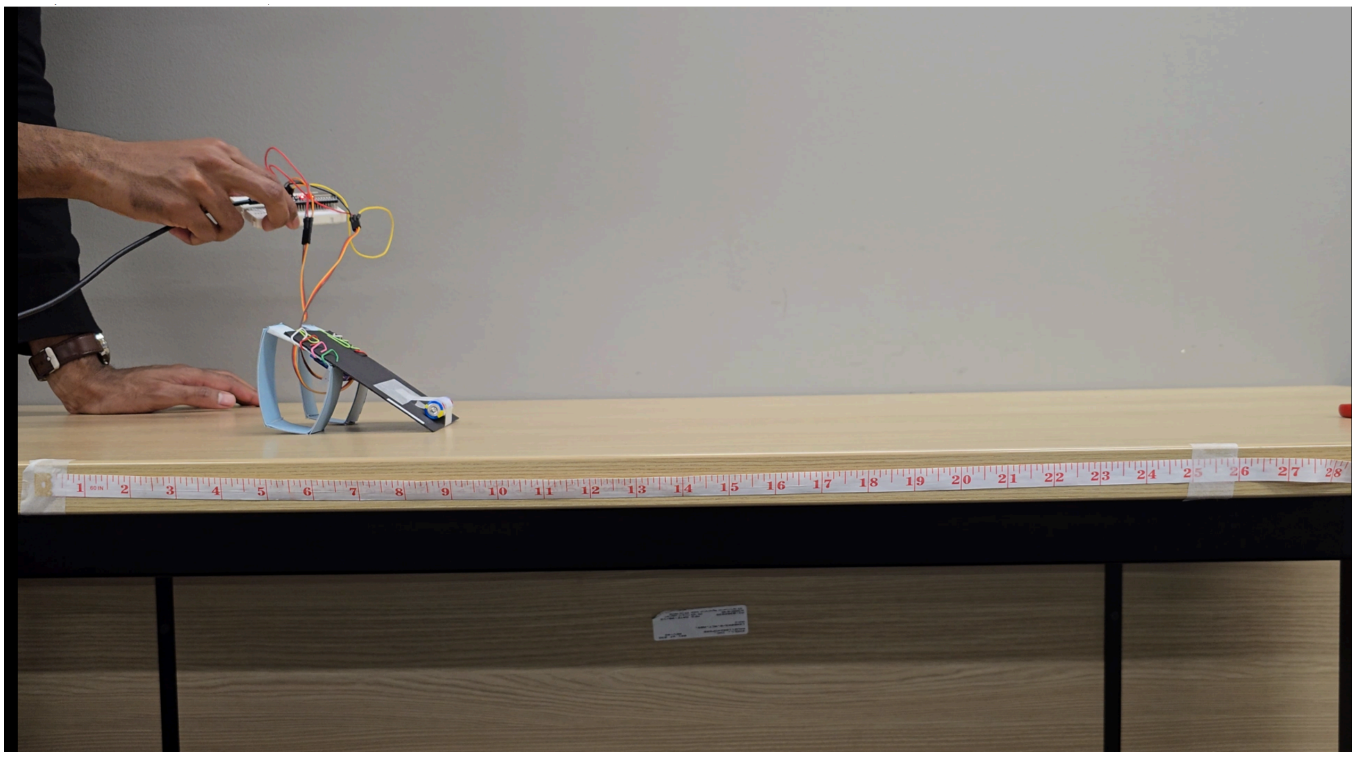
1. Small (Baseline) 4 cm, 4 cm, 2 cm, 6 cm,
2. Medium 5 cm, 5 cm, 3 cm, 7 cm,
3. Large 6 cm, 6 cm, 4 cm, 8 cm,

Methodology & Data Acquisition To ensure consistent data capture across all three configurations, the following protocol was utilized:Execution: The robot was placed on a flat, high-friction surface and executed a pre-programmed open-loop hopping sequence (Bang-Bang control).

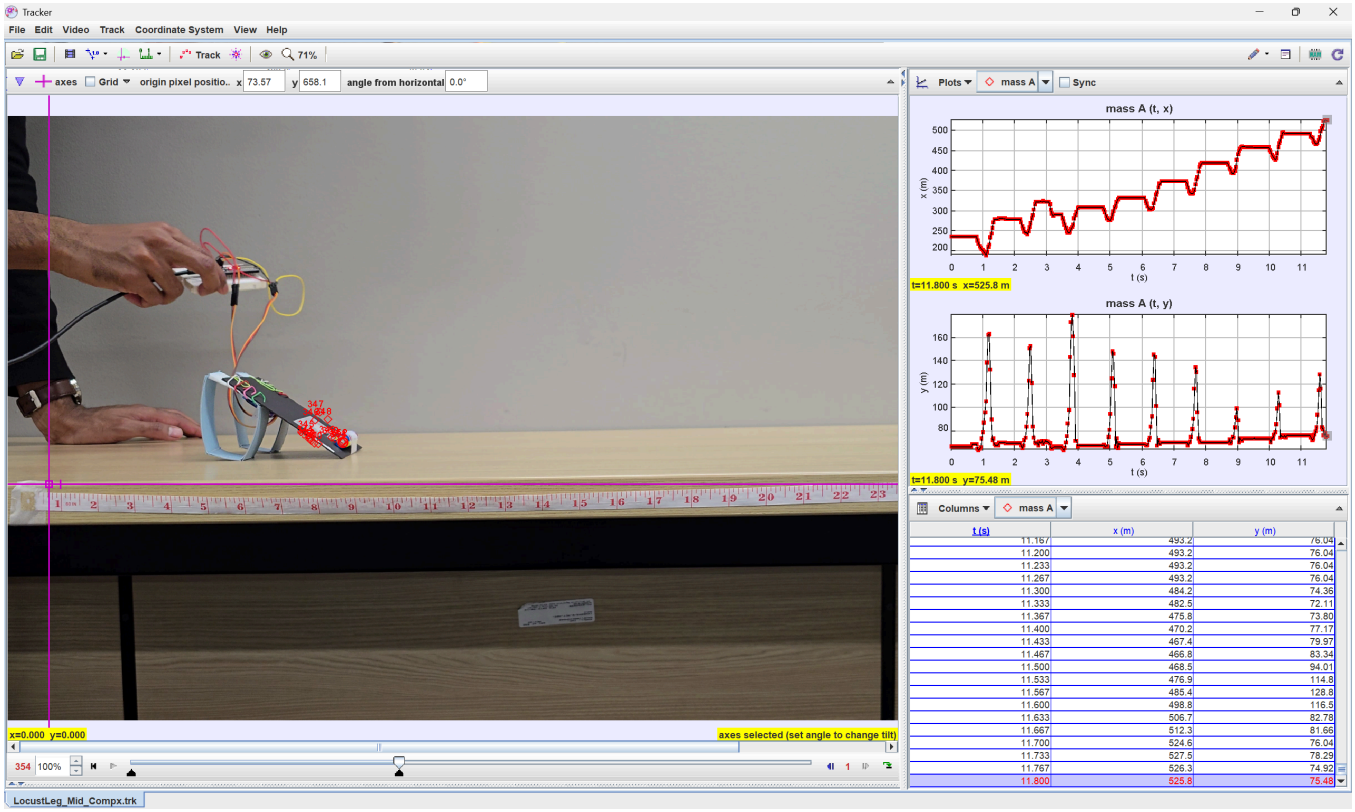
Video Recording: Each trial was recorded using a stationary high-definition camera positioned orthogonal to the robot's plane of motion to minimize perspective distortion and parallax error.

Tracker Analysis: The raw video footage was processed using Tracker Video Analysis software. Virtual markers were manually tracked frame-by-frame on the robot’s "Nose" (front body) and "Foot" to extract precise kinematic data.4.4 Performance MetricsUsing the coordinate data exported from Tracker, we calculated the following key performance indicators (KPIs):Longitudinal Displacement: The continuous position $x(t)$ of the robot over time.Total Distance Traveled: The net forward distance achieved after a fixed duration of hopping ($x_{final} - x_{initial}$).Maximum Jump Height: The peak vertical displacement (y_{max}) achieved during the flight phase.

```
from IPython.display import Image, display
display(Image(filename='experimentsetup.png', width=800, height=600))
```



```
from IPython.display import Image, display
display(Image(filename='trackersetup.png', width=800, height=600))
```



Data Visualization & Performance Analysis

Overview This script generates publication-quality visualizations to evaluate and compare the kinematic performance of three distinct robot leg configurations: Small, Medium, and

Large.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.ticker as ticker

# --- 1. VISUAL STYLE CONFIGURATION ---
# Use a high-contrast style optimized for papers and slides
sns.set_theme(style="white", context="talk")

# Global Matplotlib Parameters for Typography & Layout
plt.rcParams.update({
    'font.family': 'sans-serif',
    'font.sans-serif': ['Arial', 'Helvetica', 'DejaVu Sans'],
    'font.size': 16,
    'axes.labelsize': 16,
    'axes.labelweight': 'bold',
    'axes.titlesize': 18,
    'axes.titleweight': 'bold',
    'xtick.labelsize': 14,
    'ytick.labelsize': 14,
    'legend.fontsize': 14,
    'lines.linewidth': 4,      # Thicker lines for visibility
    'axes.spines.top': False,  # Minimalist look
    'axes.spines.right': False
})

# Consistent Color Palette (Blue, Orange, Green)
color_map = {
    'Small Leg': '#1f77b4',   # Muted Blue
    'Medium Leg': '#ff7f0e',  # Safety Orange
    'Large Leg': '#2ca02c'    # Cooked Asparagus Green
}
```

Data Loading & Processing Pipeline:

This segment handles data ingestion. It defines a robust processing function process_leg_data that normalizes the raw telemetry:Sorting: Ensures time-series data is ordered.Normalization: Zeros the starting X-position so all robots start at $x = 0$.Metric Extraction: Calculates the exact vertical jump height ($Y_{max} - Y_{start}$).

```
# --- 2. DATA INGESTION ---
def process_leg_data(df):
    """
    Standardizes raw telemetry data.
    Returns: Processed DataFrame, Max Jump Height (float)
    """
    # 1. Sort by time to prevent plotting errors
    df = df.sort_values('t')

    # 2. Calculate relative displacement (Start at X=0)
    df['displacement_x'] = df['x'] - df['x'].iloc[0]
```

```
    # 3. Calculate max vertical jump height
    initial_y = df['y'].iloc[0]
    max_y = df['y'].max()
    jump_height = max_y - initial_y

    return df, jump_height

# Load Data from CSVs
try:
    large_leg_df = pd.read_csv('RobotParamets - LargeLeg.csv')
    medium_leg_df = pd.read_csv('RobotParamets - MediumLeg.csv')
    small_leg_df = pd.read_csv('RobotParamets - SmallLeg.csv')

    # Process Datasets
    large_leg_df, h_large = process_leg_data(large_leg_df)
    medium_leg_df, h_medium = process_leg_data(medium_leg_df)
    small_leg_df, h_small = process_leg_data(small_leg_df)

except FileNotFoundError:
    print("Error: Data files not found. Check directory.")
```

This segment generates the final visual output. It creates a single figure with two subplots side-by-side:

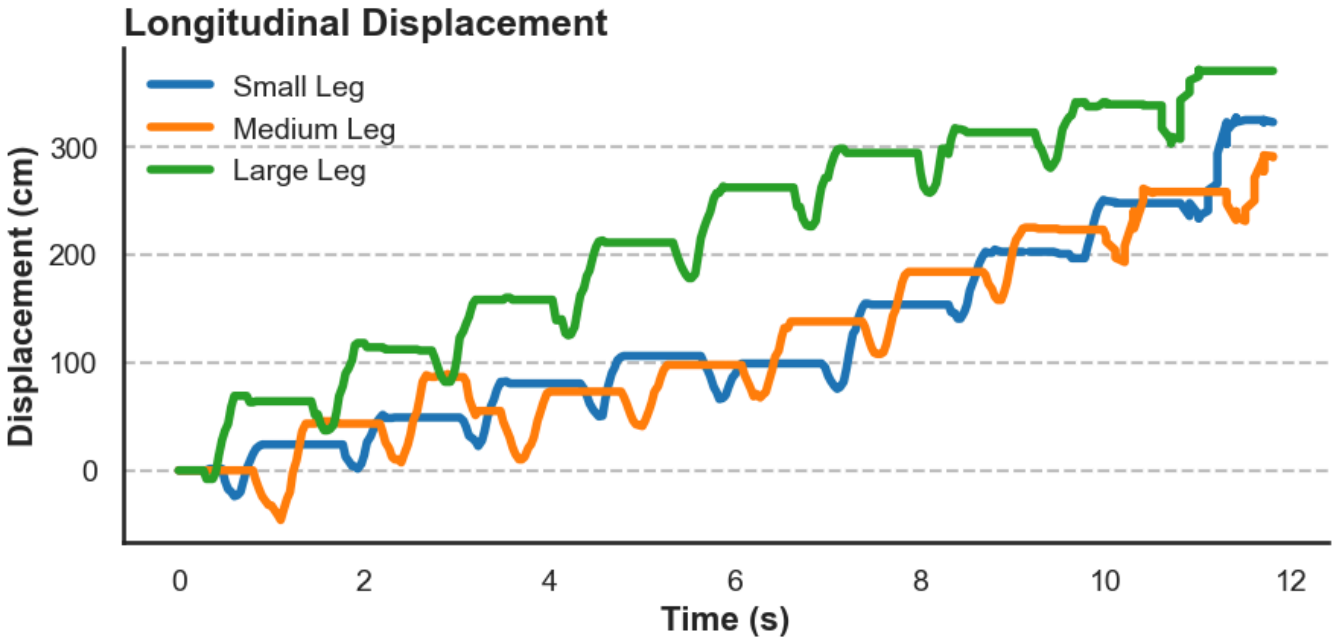
Left Panel: The displacement trajectory line chart.

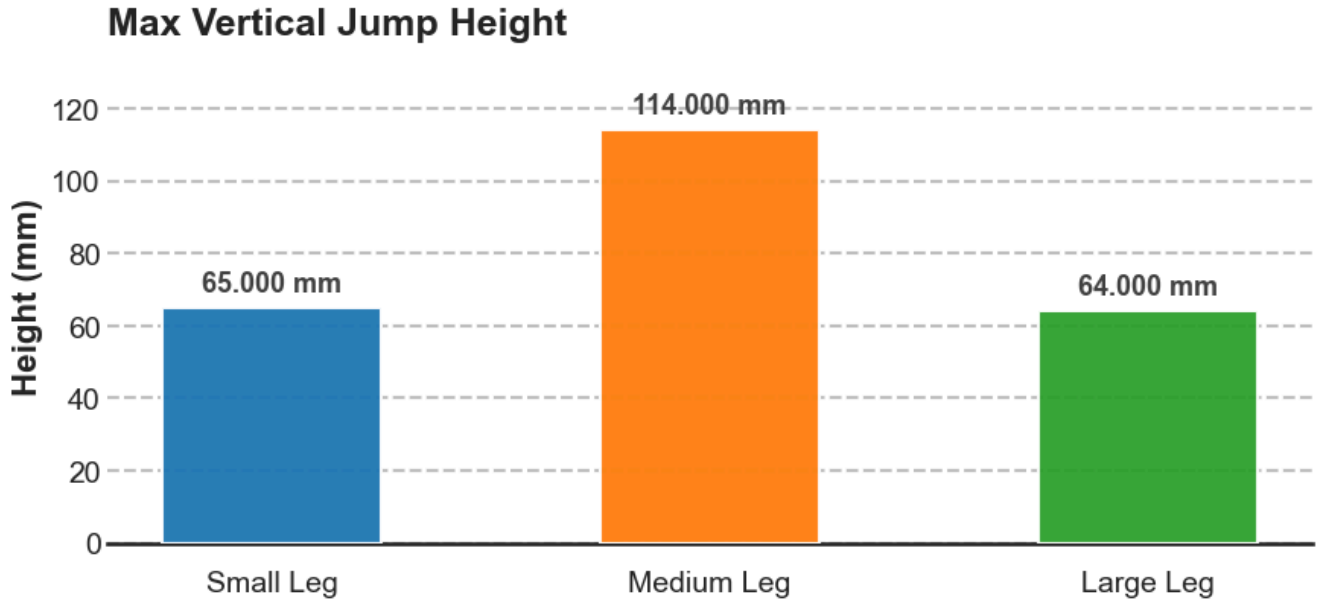
Right Panel: The jump height bar chart. This layout is cleaner than having separate files and makes comparison easier.

```
# --- 3. UNIFIED VISUALIZATION DASHBOARD ---
# Reduced figsize (10x10) to fit comfortably in a notebook cell
fig, axes = plt.subplots(2, 1, figsize=(10, 10))

# --- TOP PANEL: Displacement vs Time (Line Chart) ---
ax1 = axes[0]
ax1.plot(small_leg_df['t'], small_leg_df['displacement_x'],
         label='Small Leg', color=color_map['Small Leg'])
ax1.plot(medium_leg_df['t'], medium_leg_df['displacement_x'],
         label='Medium Leg', color=color_map['Medium Leg'])
ax1.plot(large_leg_df['t'], large_leg_df['displacement_x'],
         label='Large Leg', color=color_map['Large Leg'])

ax1.set_title('Longitudinal Displacement', loc='left')
ax1.set_xlabel('Time (s)')
ax1.set_ylabel('Displacement (cm)')
ax1.yaxis.grid(True, linestyle='--', alpha=0.5, color='grey')
ax1.legend(frameon=False, loc='upper left')

# --- BOTTOM PANEL: Max Jump Height (Bar Chart) ---
ax2 = axes[1]
legs = ['Small Leg', 'Medium Leg', 'Large Leg']
heights = [h_small, h_medium, h_large]
bar_colors = [color_map[leg] for leg in legs]
```

```python
# Draw Bars (width adjusted to 0.5 for cleaner look)
bars = ax2.bar(legs, heights, color=bar_colors, alpha=0.95, width=0.5, zorder=3)

# Dynamic Y-Axis limits for labels
y_max = max(heights)
ax2.set_ylim(0, y_max * 1.2) # Slightly more headroom for labels

# Add Data Labels on top of bars
for bar in bars:
    height = bar.get_height()
    ax2.annotate(f'{height:.3f} mm',
                 xy=(bar.get_x() + bar.get_width() / 2, height),
                 xytext=(0, 5),
                 textcoords="offset points",
                 ha='center', va='bottom',
                 fontsize=13, fontweight='bold', color='#444444')

ax2.set_title('Max Vertical Jump Height', loc='left')
ax2.set_ylabel('Height (mm)')
ax2.yaxis.grid(True, linestyle='--', alpha=0.5, color='grey', zorder=0)
sns.despine(ax=ax2, left=True)
ax2.tick_params(axis='y', length=0)

# --- SAVE & DISPLAY ---
plt.tight_layout(pad=3.0) # Balanced spacing
plt.savefig('Robot_Performance_Dashboard_Compact.png', dpi=300, bbox_inches='tight')
plt.show()

print("Compact dashboard generated successfully.")
```

```
Compact dashboard generated successfully.
```





Conclusion & Discussion

The experimental evaluation of the geometric scaling (Small, Medium, Large) reveals a distinct trade-off between longitudinal speed and vertical capability.

Longitudinal Displacement(Stride Length) As observed in Graph 1, the total distance traveled exhibited a positive linear correlation with link length.

The Result: The **Large Configuration** (Link A=6cm, Link D=8cm) achieved the greatest total displacement, significantly outperforming the Small (Baseline) design.Physical Interpretation: Increasing the length of the output leg (Link D) increases the effective stride length. For a constant angular sweep of the servo, the foot tip traces a longer arc,

propelling the robot further forward per hop. This confirms that for pure distance, larger link geometry provides a kinematic advantage.

Vertical Jump Height(Torque Limitations)As observed in Graph 2, the vertical performance followed a non-linear trend, identifying the **Medium Configuration** (Link A=5cm, Link D=7cm) as the optimal design for jumping.

The Result: While the Large legs provided a higher static standing height, they failed to achieve the maximum ballistic jump height. The Medium legs outperformed both the Small (limited by reach) and the Large (limited by power).

Physical Interpretation: This result illustrates the critical limit of the actuator's torque capacity (0.15 Nm).Small Legs: The motors have ample torque, but the short links limit the vertical range of motion.Large Legs: The lever arm is too long. The torque required to lift the body explosively exceeds the motor's stall limits, resulting in a slower "push" and a lower jump.

Medium Legs: This configuration represents the "**Sweet Spot**," balancing the kinematic benefit of longer legs with the dynamic limitations of the servo motors.5.3 Final Design RecommendationWhile the Large configuration offers the best forward velocity, the **Medium Configuration provides the most robust overall performance, maximizing obstacle clearance (jump height) while maintaining competitive forward travel**. Future iterations should utilize the Medium geometry unless higher-torque motors are upgraded to support the Large linkage.

Part 4: Foldable Robotics Manufacturing Workflow

All dimensions are in CENTIMETERS (cm)

Workflow Overview

1. Read EXACT DXF geometry (body outline and hinge slot shapes)
2. Build five-layer laminate structure (rigid-adhesive-flexible-adhesive-rigid)
3. Generate manufacturing files for laser cutting

1. Setup and Imports

```
# Install required packages if needed
!pip install shapely matplotlib ezdxf numpy --break-system-packages -q
```