Testudog: Autonomous Quadruped Robot for Unstructured Terrain Navigation

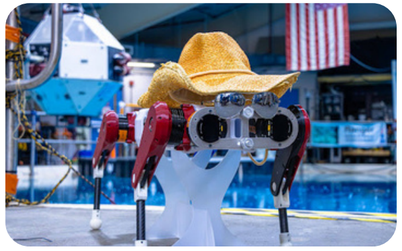
Arjun Mudda, Arian Moradi, Saiarun Jayanthi, and Krish Sridhar

Background

Quadruped robots provide an opportunity for traversing complex terrain not accessible to traditional robots. However, robust autonomous navigation, especially through variable environments, remains a challenge. Enabling walking requires fundamentally different design and control decisions compared to traditional robots [3].

Proposed Solution

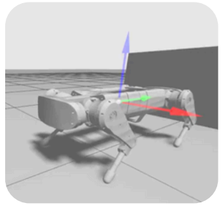
Testudog is a quadruped robot platform that seeks to provide a low-cost and fully autonomous navigation solution.



References

[1] Focchi, Michele, et al. "Torque-control based compliant actuation of a quadruped robot." 2012 12th IEEE international workshop on advanced motion control (AMC). IEEE, 2012.
[2] Spong, Mark W., et al. Robot Modeling and Control. John Wiley & Sons, 2005.
[3] Hardarson, Freyr. Stability analysis and synthesis of statically balanced walking for quadruped robots. Diss. Maskinkonstruktion, 2002.

Control Systems Architecture



Hardware Considerations

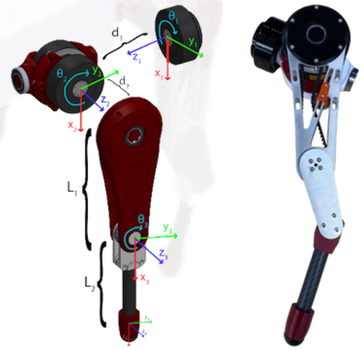
Testudog features an array of sensors aimed to support mission critical functions such as walking and obstacle avoidance. In relation to these goals we feature an IMU, LIDAR, Stereo Camera, and our motors contain driver boards which we use to calculate the torque exerted.

Kinematics and Trajectory Generation C++ and Python Library

Forward Kinematics

A - Length of common normal (zi to zi+1 along xi)
α - Angle about common normal (zi to zi+1 about xi)
d - Offset to common normal (xi-1 to xi along zi)
θ - Angle about z (xi-1 to xi about zi)

Link	A	α°	d	θ
1	0	-90	d_1	θ_1
2	L_1	0	d_2	θ_2
3	L_2	0	0	θ_3



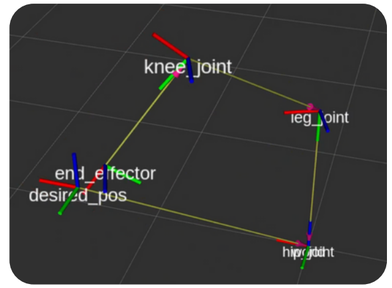
Levenberg Maquardt Iterative Inverse Kinematics Implementation:

$\lambda_{init} = 0.3$
$\alpha = 0.01$

while $error_{curr}$ not within ε and *iter threshold*:
$\theta_{prev} = \theta_{new}$
$\theta_{new} = \theta_{curr} + \alpha (J_v^T J_v + \lambda I_{3x3})^{-1} * J_v^T (foot_{des} - foot_{curr})$ [a] [b]
$error_{prev} = error_{curr}$
$error_{curr} = ½ ||(FK(\theta_{new}) - foot_{des})||^2$
if($error_{curr} < error_{prev}$) :
  λ *= 0.8
else:
  $error_{curr} = error_{prev}$
  $\theta_{new} = \theta_{prev}$
  λ *= 2.0

[a] J_v refers to Velocity Jacobian (3x3 Matrix)
[b] foot is a 3x1 vector describing the end effector x,y,z coordinate

Rviz2 Kinematics Simulation



Overview of Jacobian Utilization in IK Algorithm

Quaternion Jacobian(7x3): Fast convergence orientation-wise. Struggles with both orientation and position.
Analytical Jacobian(6x3): Slow and numerically unstable.
Velocity Jacobian(3x3): ***Currently implemented Jacobian*** *– Computationally fast, numerically stable (when paired with a damping matrix), and rapid convergence.*