Technical Report: Control and Design of an Open-Source Two-Degree-of-Freedom Hopping Robot

1st Weiyi Tang
Electrical and Systems Engineering
University of Pennsylvania
Philadelphia, PA

2nd Sonia F. Roberts
Electrical and Systems Engineering
University of Pennsylvania
Philadelphia, PA

3rd Daniel E. Koditschek
Electrical and Systems Engineering
University of Pennsylvania
Philadelphia, PA

***Abstract*—Using mechanical design inspired by the Ghost Minitaur and the open-source motor controller hardware from the Stanford Doggo, we built an open-source two-degree-of-freedom hopping robot. The robot hops using a Raibert-inspired reactive controller on the leg length and velocity. This technical report documents the project and provides a guide to others interested in building similar research robots.**

***Index Terms*—reactive, legged, robot, open-source, odrive**

I. Introduction

A. Motivation

The Ghost Minitaur[1] [1] is a medium-sized legged robot platform which has been used to perform a variety of robotics research [2]–[7], much of which makes use of the actuator transparency [8] conferred by the direct-drive (no gearbox) motors and high-bandwidth motor controllers. This robot's morphology and "compliant" direct-drive architecture both warrant further study, but Minitaur is not open-source and is no longer being sold by Ghost. Researchers therefore need alternative open-source platforms that they can build and modify for their own projects.

B. Background and related work

The Stanford Doggo [9] is a fully open-source platform, down to the open-source Teensy microcontroller and ODrive motor controllers. The result is a highly agile and accessible robot which can be programmed using a variety of methods, including open-loop trajectory control and impedance control. The extensive documentation[2] provided enables other researchers to adapt the control system to different morphologies.

The final piece of the puzzle was the reactive jumping controller. The leg emulates a soft spring (low proportional and derivative gain) during the first half of stance, while it compresses, and then switches to emulating a stiff spring (high gains) when the leg length velocity changes sign, and the leg starts to extend. This simple controller dates back to Marc Raibert's seminal work on legged locomotion [10] and is still used to underpin robust legged robot locomotion to this day [3], [4].

This work was supported by NSF NRI-2.0 grant 1734355.

[1]Ghost Robotics, 3401 Grays Ferry Ave, Philadelphia, PA 19146

[2]Stanford Doggo Github page

C. Contribution

We describe a fully direct-drive hopping robot using the motor controllers and control architecture of the Stanford Doggo, the basic morphology of the Ghost Minitaur, and reactive control reminiscent of Raibert.

II. Materials and construction

A. Materials list

No.	Item	Details
2	Motor	T-Motor U8 PRO KV100
1	Motor controller	ODrive v3.6
2	Encoder	AS5047D
1	Microcontroller	Teensy 3.5
2	Encoder mount	Custom 3D-printed part
2	Motor mount	Custom waterjet-cut part
1	ODrive mount	Custom 3D-printed part
1	Chassis	Custom waterjet-cut parts
2	Leg	Prototype Minitaur legs

TABLE I: Summary table of parts list.

You can find a bill of materials here: Hopping Robot BOM

1) Motor: (2) T-Motor U8 PRO KV100.

These motors are the same ones used in Minitaur, and are in the same line as the motors used in the Doggo. Due to the large gap radius, these motors are able to produce a large amount of torque without the use of a gearbox [11].

2) Motor controller: (1) ODrive v3.6.

This motor controller is a DC motor controller which could command two motors at the same time. It is open-source motor controller with clear ducomentation and useful support forum. And it is much more high-performance and accurate than previous hobbyist DC motor controller.

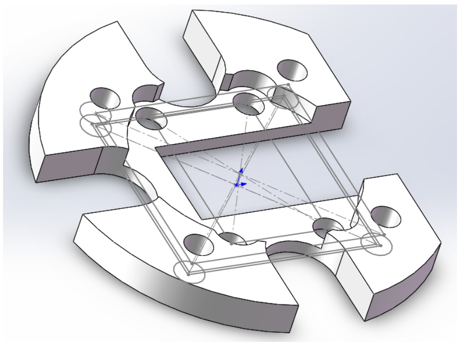
3) Motor encoder: (2) AS5047D.

It is High Speed Magnetic Rotary Position Sensor with independent output interfaces. It is easy to use, because of no programmer needed. The resolution of the incremental ABI interface is programmable with a maximum resolution of 2000 steps / 500 pulses per revolution in decimal mode and 2048 steps / 512 pules per revolution in binary mode.

4) Microcontroller: (1) Teensy 3.5.

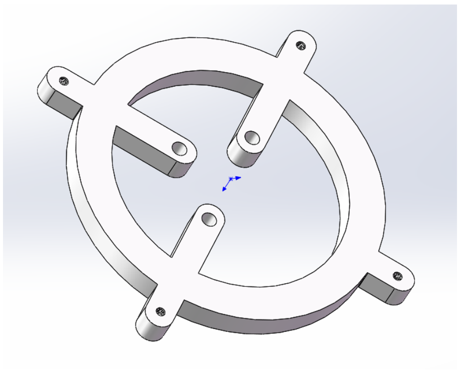
Teensy 3.5 is selected because it is a low-cost, Arduino-compatible microcontroller that is common among the open-source community. It has high bandwidth communication capability and mutiple PWM ports.

5) Encoder mount: (2) Custom 3D-printed part.



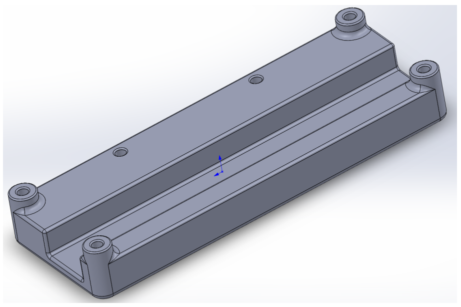
6) Motor mount: (2) Custom waterjet-cut part.

This part was waterjet-cut from 1/4" aluminum.



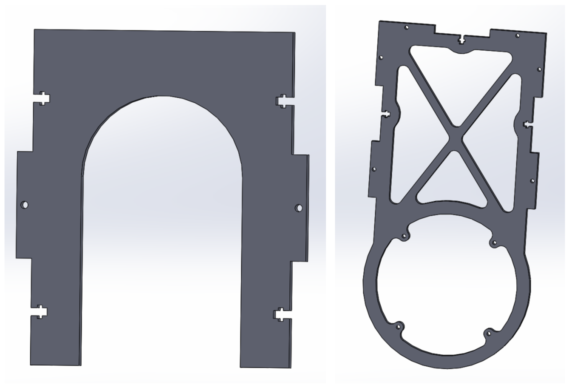
7) ODrive mount: (2) Custom 3D-printed part.

This part is should not be withstanding strong forces and can be printed at minimum infill.



8) Chassis: (2) Custom waterjet-cut parts.

The robot's body consists of two sets of two identical pieces of waterjet-cut 1/4" aluminum. The pieces are connected by M3 bolts and square nuts.



9) Legs: 2: Prototype legs.

We used prototype Minitaur legs that were developed before Ghost was incorporated as a company. The design of the legs is similar to the Doggo legs, though, and researchers interested in replicating this project can use the Doggo leg design as a starting point. The only important difference is that our legs mount directly to the motors, and the Doggo legs mount to a drive shaft.

B. Mechanical design

There are few steps to put all the hardware things together:

1) : Utilize 3D printer to make a encoder mount to attach AS5047D to each motor;

2) : The defult voltage of AS5047D is 5V, but we need 3.3V mode, so we need to modify the default circuit by moving the resistor with 0 value in 'R2' to 'R1' and changing 'JP1'connector correspondingly. More details can be checked in Fig. 1;

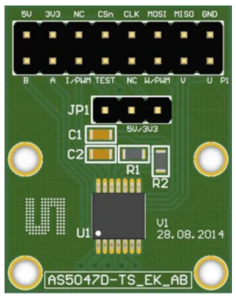

5V 3V3 NC CSn CLK MOSI MISO GND
B A I/PWM TEST NC W/PWM V U P1
JP1
5V/3V3
C1
C2
R1
R2
U1
V1
28.08.2014
AS5047D-TS_EK_AB

Fig. 1: Diagram of AS5047D-TS_EK_AB encoder

3) : Connecte AS5047D to ODrive with five wires. To be more precise, pin 'A', 'B', 'Z', '3.3V', 'GND' of ODrive in J4 and AS5047D should be connected correspondingly. More details can be referred in Fig. 2;

4) Utilize 3D printer to make a ODrive mount to fix ODrive:

5) Utilized water-jet cutting to make two motor mounts to fix motors:

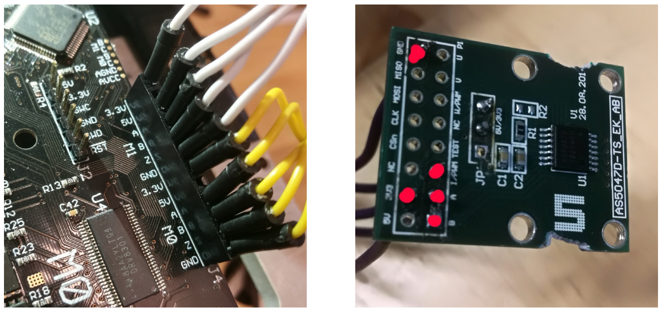


Fig. 2: Connection between Odrive and encoder (Red dots in right picture are the ports needed to be connected)

6) Attach Minituar leg to motors to give us this hopping robot: You can see the structure of the entire machine leg in Fig. 3 below. L_1 and L_2 are Upper and lower leg length, and γ is the separation.

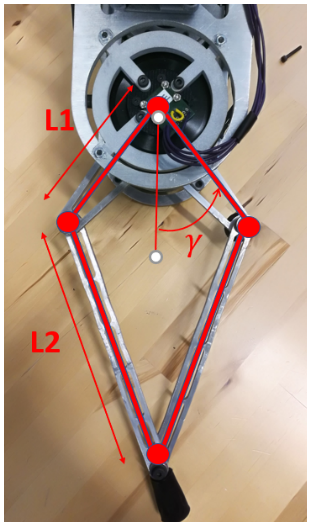


Fig. 3: Diagram of the Hopper Leg

III. Control

In order to use position control mode, we first need to initialize motor correctly. Encoder Count Per Revolution(CPR) and motor pole pairs are two important parameters and should be modified for different motor and encoder according to ODrive document. In our hopper, we set them to be 2000 and 21 respectively. Please refer to motor and encoder datasheet to set these two parameters if you want to use different motor or encoder. Also, you can set any other parameters you want, like configuration current limit and velocity limit. Setting appropriate parameters is important which will affect initialization process. If initialization doesn't work well (e.g. After hearing a beep, the motor does not turn slowly in one direction for a few seconds, then back in the other direction.), try to increase the max voltage value of resistance calibration. We set it to be 4 for our hopper.

Stanford Doggo [9] utilized two motors facing the same direction to control each leg by using gearbox, while our hopper utilized two motors facing the opposite direction. We therefore need to adjust the kinematics functions for the leg. Here are two angles which are related to this issue, alpha and beta. As you can see in Fig 4. and Fig. 5 (Fig 4. for Doggo and Fig 5. for our hopper), Both of them are measured from the downward vertical, and increase in the counterclockwise direction. The way Doggo computes gamma can be seen in equation (1). Here we applied a parameter called axis_direction and set it to be -1 to change the direction. To be more precise, we multiply axis_direction to every paramter related to axis zero in Firmware/MotorControl/Controller.cpp to change the direction as we want. Modified equation can be seen in equation (2). We now only care about coupled control, so we only need to adjust this part. The diagram of angles of our hopper can be seen in Fig. 5.

$$\begin{aligned} \alpha &= measured\,rad + \frac{\pi}{2} \\ \beta &= measured\,rad - \frac{\pi}{2} \\ \gamma &= \frac{\alpha - \beta}{2} \end{aligned} \tag{1}$$

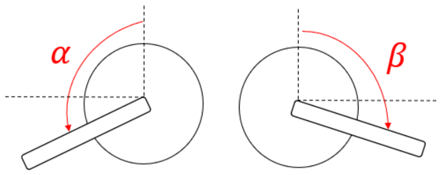


Fig. 4: Doggo mleg angle

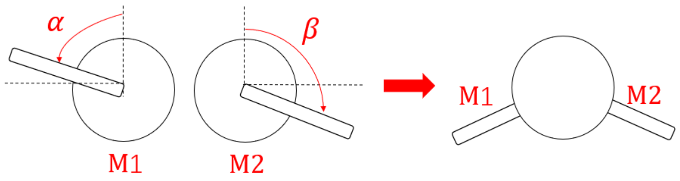


Fig. 5: Our hopper leg angles

$$\begin{aligned} \alpha &= -measured\,rad + \frac{\pi}{2} \\ \beta &= measured\,rad - \frac{\pi}{2} \\ \gamma &= \frac{\alpha - \beta}{2} \end{aligned} \tag{2}$$

As for control part overview, Teensy 3.5 will compute leg trajectories and send leg position commands to the ODrive motor controllers at 100Hz. The motor controllers run field-oriented-control motor commutation at 10KHz to control the torque applied by the T-Motor with positional feedback from axially mounted magnetic encoders with 2000 count per revolution (CPR).

Stanford Doggo [9] applied ChibiOS to help them control different gaits more expediently. ChibiOS is a compact and fast real-time operating system supporting multiple architectures including Arduino. They created three main ChibiOS thread to help control different gaits, control thread, serial thread and USB serial thread. Control thread executes PID and controls the motor. Serial thread reads any incoming serial messages from ODrive. USB serial thread reads any incoming serial message from the computer. ChibiOS thread will be called one by one in cycle, so they use time as a criterion to control robot to jump. But we adjust this part by using our own method.

Here we can apply PD control to allow hopper's leg to perform like a spring. Our jump method can be divided into two phases: Hopper in the air and hopper touches the ground. When hopper is in the air, we set a hard gain for it to jump. When it touches the ground, a soft gain is picked to allow hopper to compress. When the extension length decreases to the min value or the direction of velocity of hopper changes, a hard gain is picked to give enough torque for it to jump, then it comes back to phase one. We set the same position for the motor for both phases.

The code can be referred in Jumpcode.py.

IV. ACKNOWLEDGMENTS

This work was supported by NSF NRI-2.0 grant #1734355.

REFERENCES

[1] G. Kenneally, A. De, and D. E. Koditschek, "Design principles for a family of direct-drive legged robots," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 900–907, 2016.

[2] T. T. Topping, V. Vasilopoulos, A. De, and D. E. Koditschek, "Composition of templates for transitional pedipulation behaviors." ISRR, 2019.

[3] S. F. Roberts and D. E. Koditschek, "Mitigating energy loss in a robot hopping on a physically emulated dissipative substrate," *Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

[4] A. De and D. E. Koditschek, "Vertical hopper compositions for preflexive and feedback-stabilized quadrupedal bounding, pacing, pronking, and trotting," *The International Journal of Robotics Research*, vol. 37, no. 7, pp. 743–778, 2018.

[5] V. Vasilopoulos, T. T. Topping, W. Vega-Brown, N. Roy, and D. E. Koditschek, "Sensor-based reactive execution of symbolic rearrangement plans by a legged mobile manipulator," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3298–3305.

[6] F. Qian, D. Jerolmack, N. Lancaster, G. Nikolich, P. Reverdy, S. Roberts, T. Shipley, R. S. Van Pelt, T. M. Zobeck, and D. E. Koditschek, "Ground robotic measurement of aeolian processes," *Aeolian research*, vol. 27, pp. 1–11, 2017.

[7] D. J. Blackman, J. V. Nicholson, C. Ordonez, B. D. Miller, and J. E. Clark, "Gait development on minitaur, a direct drive quadrupedal robot," in *Unmanned Systems Technology XVIII*, vol. 9837. International Society for Optics and Photonics, 2016, p. 98370I.

[8] G. Kenneally, W.-H. Chen, and D. Koditschek, "Actuator transparency and the energetic cost of proprioception." ISER, 2018.

[9] N. Kau, A. Schultz, N. Ferrante, and P. Slade, "Stanford doggo: An open-source, quasi-direct-drive quadruped," *arXiv preprint arXiv:1905.04254*, 2019.

[10] M. H. Raibert, *Legged robots that balance*. MIT press, 1986.

[11] S. Seok, A. Wang, D. Otten, and S. Kim, "Actuator design for high force proprioceptive control in fast legged locomotion," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 1970–1975.