Bioinspired Mobile Robot - Spring 2019 MAE207 Project

Henry Chang, Charlie Hsiao, Philip Hsieh, Yujie Wang and Nicholas Gravish*
Department of Mechanical and Aerospace Engineering, University of California San Diego

I. INTRODUCTION

This is the project for MAE207 Bioinspired Robotics. Given that lots of modern robots are using no-gear drive system to operate responsively and efficiently. We are trying to study the design and control of a direct and quasi direct-drive legged robot with the following kit components, Quanum brushless DC motors, ODrive BLDC controllers, Capacitive rotary encoders, 24V 500W power supply and leg assemblies (bearings, legs, hardware) We will use forward, inverse kinematics [1] and develop Python code to control our legged robot.

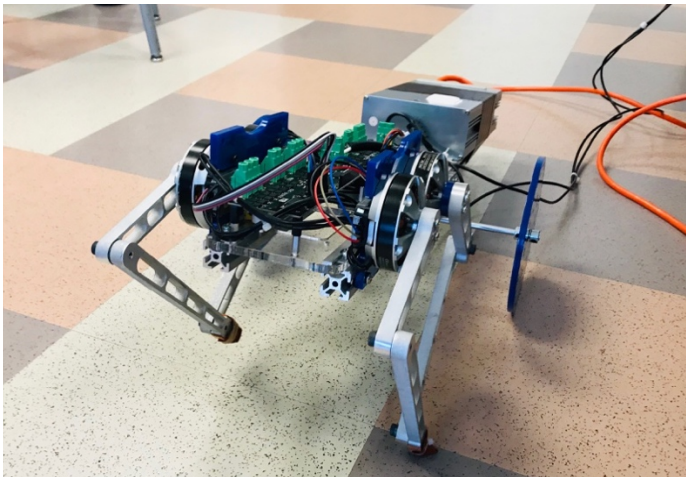

Fig. 1. Overview of our robot

II. FUNCTIONALITY OF ROBOT

We are aiming at three functions of our robot, moving forward, backward, and doing push up. Besides the above functions, we could also manually move the end effector to input a desire trajectory. For our project, we set the moving pace to be more similar to walking than running.

III. STUDY & IMPLEMENT

By generating the inverse kinematics of the robot, we can convert our objective leg trajectory into motor angle space. Then, output the desired gait. Robot's legs are two five-bar linkage and each leg consists of two shorter and two longer links. The shorter leg length is $l_1 = 0.09\,m$, the longer leg length is $l_2 = 0.16\,m$, and the base link has length $w = 0.09\,m$, which is also the distance between the two motors.

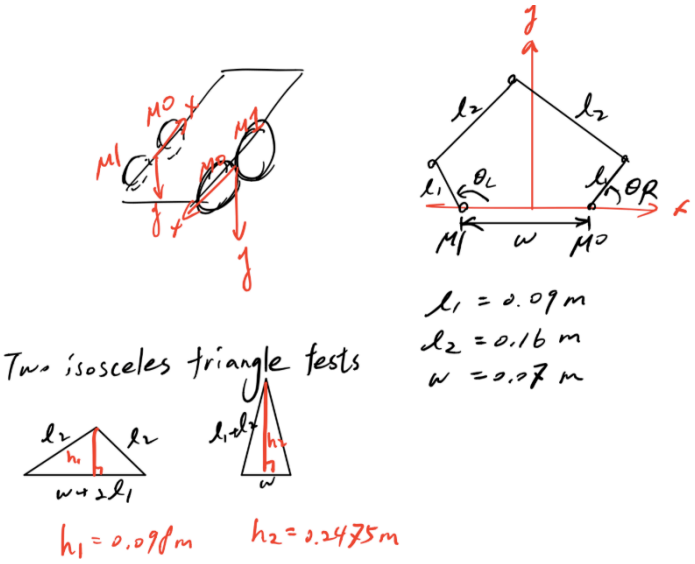

Fig. 2. Geometry of each leg (right), Configuration of the robot (top left) and Special position of leg (bottom left)

The leg trajectory is a combination of straight-line stance phase and parabolic function swing phase. There are three critical points to determine the trajectory, stance phase lift point, stance phase touch point and the highest point from the ground of swing phase (according to our coordinate, it should be the lowest point of the parabola).

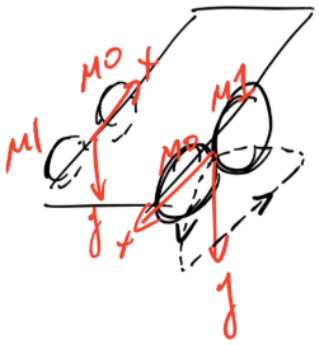

Fig. 3. The leg trajectory (Noting that the sequence of the motors on both legs are opposite, which means the coordinates are also in opposite directions)

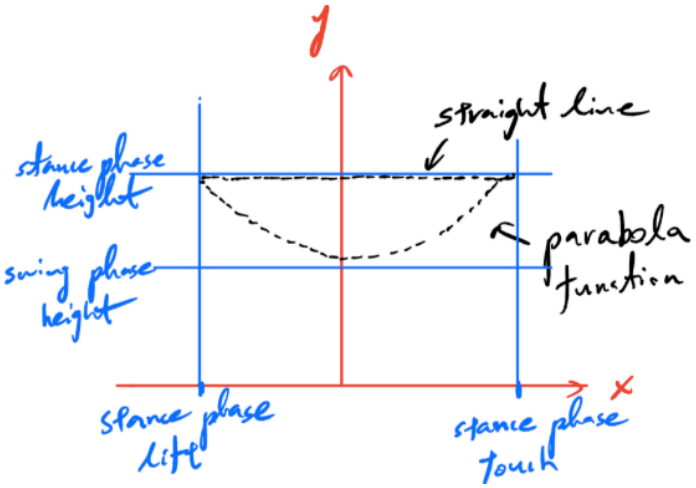

Fig. 4. Details of the trajectory

For the forward moving gait, we set following geometry parameters and proper step size to generate the trajectory. Then, we input three critical points to calculate the second order polynomial.

$$x_{\text{stance\_phase\_touch}} = 0.10\ m$$
$$x_{stance\_phase\ lift} = -0.10\ m$$
$$y_{stance\_phase\_height} = 0.18\ m$$
$$y_{\text{swing\_phase\_height}} = 0.10\ m$$
$$Swing\ step\ size = 0.002\ m$$
$$Stance\ step\ size = 0.004\ m$$

Finally, input the trajectory (cartesian space) to inverse kinematics to get the motor rotation angle (joint space) and convert the angle into encoder positions (2π rad = 8192 encoder units). Following shows the code we used to derive inverse kinematics. [2]

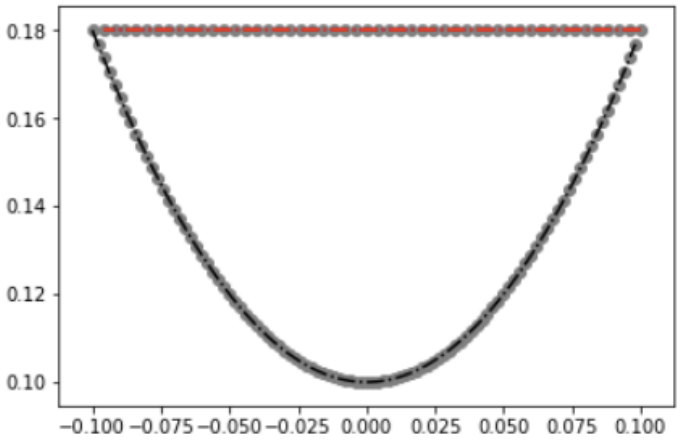

Fig. 5. Calculated trajectory

```
l1 = 0.09;    l2 = 0.16;     w = 0.07;

def IK_5_link(x, y, l1 = l1, l2 = l2, w = w):
    def leg_wide(var):
        return np.linalg.norm([var[0], \
                               var[1] - np.pi])
    def x_constraint_equation(var):
        return l1**2 - l2**2 + (x - w/2)**2 \
            + y**2 - 2*l1*(y*np.sin(var[0]) \
            + (x - w/2)*np.cos(var[0]))
    def y_constraint_equation(var):
        return l1**2 - l2**2 + (x + w/2)**2 \
            + y**2 - 2*l1*(y*np.sin(var[1]) \
            + (x + w/2)*np.cos(var[1]))

    res = minimize(leg_wide,          \
                   (0.1, 9*np.pi/10), \
                   method="SLSQP",    \
                   constraints =      \
({"type":"eq","fun": x_constraint_equation},
{"type":"eq","fun": y_constraint_equation}))

    return (res, np.linalg.norm \
           ([x_constraint_equation(res.x), \
             y_constraint_equation(res.x)]))
```

Next, input encoder position and use [3] `controller.pos_setpoint` command to control the motors. The end effector of both legs would show the

trajectory in the Fig. 5. The last step is to make sure the robot walks smoothly by tuning the parameter of critical points and the phase difference of both legs.

Stance phase period (T_st)	1.5 sec
Swing phase period (T_sw)	1.5 sec
Phase difference between legs	0.3 sec

Form 1. Final settings for demo (unit: sec)

IV. CHALLENGES & ACCONPLISHMENT

There are several challenges show up during the designing and controlling processes. We briefly separate them into hardware and software part.

For the hardware part, the auxiliary wheels play a critical role in walking difficulty. The desirable wheel diameter for moving forward is around 15 cm. If the diameter of the wheel is smaller, the robot will tilt forward and backward frequently while moving. This is because when the leg touches the ground, the vertical force will generate a moment cause the robot to tilt backward. After the leg lifts up, the robot would tilt forward to the original position. However, if we use auxiliary wheels with a diameter of around 15 cm, the robot becomes much stable than we use smaller auxiliary wheels. The reason is that friction force between the larger wheels and the ground can generate larger moment to balance the robot and thus make the robot walk stably. Also, if the robot is at a bad angle of attack, its legs might not able to push itself to move easily. This result shows that the balance of the robot is obviously important and highly related to the robot's auxiliary wheels.

For the software part, we spent lots of time on deriving forward and inverse kinematics. The five-bar-linkage leg has a complicated kinematic model and it is a time-consuming process to calculate the Jacobian matrix. On the other hand, we learned that the code calculating the motor position and the code controlling the motor should be separated in different for loop, because the calculating process would cause some delay and makes the controlling commands fail to run at our desired time.

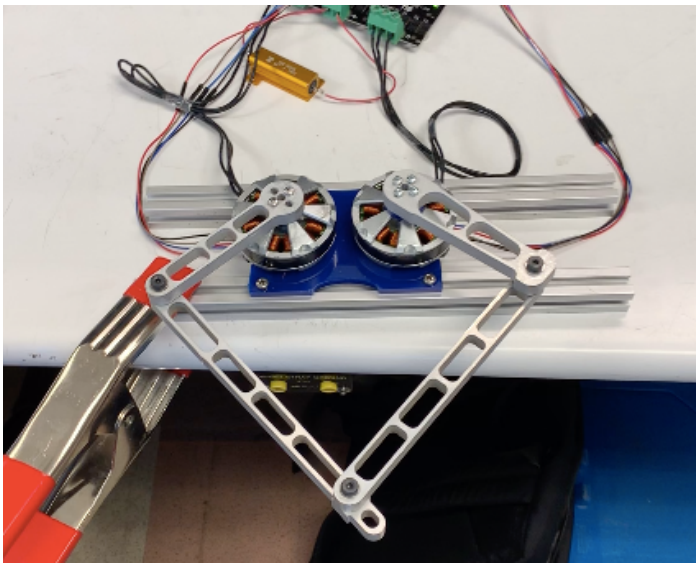

Fig. 6. Five-bar-linkage leg

To sum up, it is an amazing experience to working on this project with my teammates, Professor Nicholas Gravish and T.A. Ming-song Jiang. Programming the Python codes and applying to the motor control is rather challenging but practical.

V. ACKNOWLEDGEMENT

This project is supported by MAE207 Bioinspired Robotics course instructor Professor Nicholas Gravish and T.A. Ming-song Jiang. They provided our team with plenty useful instruction and ideas on coding, software problems and hardware issues.

VI. REFERENCE

[1] P. S. Basu, K. Farhang, 1994, *Kinematic Analysis and Design of Two-Input, Five-Bar Mechanisms Driven by Relatively Small Cranks,* Dpt. of Mechanical Engineering, Southern Illinois University, Carbondale, IL 62901

[2] MAE207 course platform

[3] ODrive GitHub