# Hexapod Robot

Yichun Gao, Chengyuan Li, Yihan Li
gyc1999, lichengyuan, yianlee@berkeley.edu

## Abstract

This project focuses on the development of a hexapod robot. We used the gait algorithms to build the system for basic motion control, and then developed different capabilities for the robot including object tracking, bluetooth control and obstacle avoidance. The basic motions include moving straight, rotation and keeping balance, which are developed by modeling and feedback control. Based on that, the robot can either track a human automatically or be controlled by humans to move around while avoiding obstacles. For different capabilities mentioned above, comprehensive sensors and multitasking are used to achieve corresponding goals.

## 1. Materials

### 1.1 Hardwares

#### 1.1.1 Development Board

The main requirement of the development board is to execute the program for controlling our robot. It means that it needs decent processing speed, enough memory, a mature robot control system and sufficient library resources. Based on these considerations, the Raspberry Pi 3b+ is our best choice because it also offers enough GPIO pins and USB ports for input and output signals.

Fig.1 Raspberry Pi 3b+

For wireless control, bluetooth is used for message transmission. Since the Raspberry Pi 3b+ supports classic bluetooth protocol, an Arduino UNO R3 (Fig.2) is used to control a HC-05 bluetooth serial pass-through module to transfer the movement of a joystick.

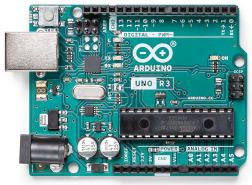

Fig.2 Arduino UNO R3

#### 1.1.2 Sensors

Our raspberry pi board carries three kinds of sensors: camera (OV5647 Camera Module, Fig.3), IMU sensor and ultrasonic sensor (HC-SR04, Fig.3).

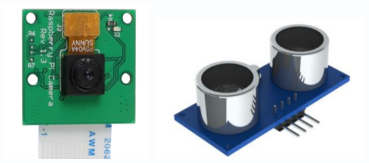

Fig.3 Camera (left) and ultrasonic sensor (right)

#### 1.1.2 Wireless Communication Module

A HC-05 bluetooth module (Fig.4) is used for wireless data transmission because it allows classical bluetooth protocol, which is faster than the speed of BLE.

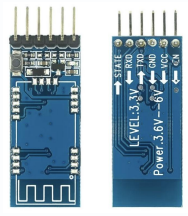

Fig.4 HC-05 bluetooth module

## 1.2 Software

We installed Raspberry OS for our project, and we use python as our programming language because the command in python is brief and we can use python threading to deal with the multitasking problem.

# 2. Preliminaries

## 2.1 Movement Control

The movement control problem of a hexapod robot is to move the tip of a leg to a 3D coordinate. The goal is to solve the angle of the three motors on a leg when given the 3D coordinate of the tip of a leg (Fig.5). Assume α, β and γ are the angles of the motors, and $L_1$ and $L_2$ are the lengths of the two parts of a leg:

$$x = [L_1 cos\beta + L_2 cos(\beta + \gamma)]cos\alpha$$
$$y = [L_1 cos\beta + L_2 cos(\beta + \gamma)]sin\alpha$$
$$z = [L_1 sin\beta + L_2 sin(\beta + \gamma)]$$

the solution of α, β and γ is as follows:

Let $K_1 = \frac{x}{cos\alpha}$,

$$\alpha = tan^{-1}(\frac{y}{x})$$
$$\gamma = cos^{-1}(\frac{K_1^2 + z^2 - K_1^2 - K_2^2}{2L_1L_2})$$
$$\beta = sin^{-1}(\frac{z}{\sqrt{(L_1 + L_2cos\gamma)^2 + (L_2sin\gamma)^2}}) - tan^{-1}\frac{L_2sin\gamma}{L1 + L2cos\gamma}$$

By using these equations, the angle of each motor can be calculated and assigned.

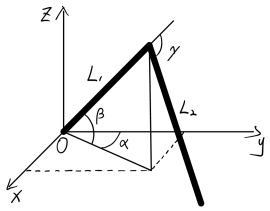


Fig.5 Hexapod robot kinematic model

With this model, we can control the tip of each leg to move to the desired position, which allows us to run gait algorithms.

## 2.2 Gait Algorithms

The gait algorithm is to let a robot know where it should place its legs exactly in each step. It should allow a robot to go straight and rotate on a 2D plane. The algorithm is designed based on the decomposition of velocity and movement discretization. To be specific, the velocity data in a control command includes linear velocity, x and y, and rotational velocity, a. The coordinate transformation can be realized by the following matrix:

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} cos(a) & sin(a) \\ -sin(a) & cos(a) \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} + \begin{bmatrix} x \\ y \end{bmatrix}$$

Denoting $p_0 = [x_0, y_0]^T$, $p_1 = [x_1, y_1]^T$, $v = [x, y]^T$ and A(a) as the rotation matrix, we have $p_1 = A(a) * p_0 + v$, which is shown in Fig.6. Since we can only apply a discrete kinematic system in a program, the equation is actually $p_i = (A(a/steps) * p_0 + v/steps)$ and $p_n = \Sigma p_i$. The $p_i$ is the coordinate for the tip of a leg in each step.

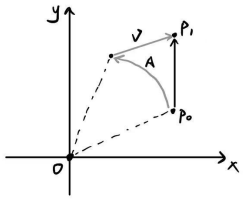


Fig.6 Composition of linear velocity and rotational velocity

Next, a hexapod robot should know which leg it should move. We have two algorithms for this part. The first one is to move 3 legs forward in air and leave the other 3 legs on the ground in each step (Fig.7), so the legs are divided into 2 groups with different colors. Their movements in one cycle are shown on the right side of Fig.7 . The F is the cycle time and the L is the least distance that the tip of a leg will move in one step. The less the F is or the bigger the L is, the faster the robot can move.

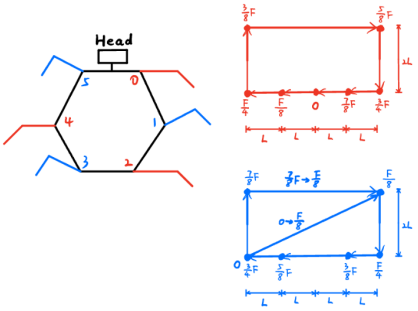


Fig.7 3 by 3 gait algorithm

The second one is to move 1 leg forward in air and leave the other 5 legs on the ground in each step (Fig.8), which allows the robot to move more stably.

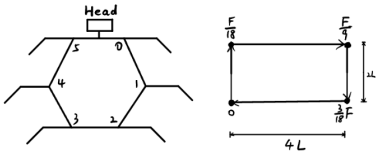


Fig.8 1 by 1 gait algorithm

In both algorithms, the legs in air move forward while the legs on the ground move backward in order to push the body forward.

## 2.3 Balancing

The balancing problem is to let the robot keep balance. To be specific, it means the baseboard of the robot should always be parallel to the horizontal plane. To solve this problem, the robot should use the data from a gyroscope. The data is filtered by a Kalman filter to reduce noise. Next, PID is used to do a negative feedback control. Whenever the baseboard is leaning towards one direction, the PID will require the robot to raise the side in that direction and lower the side in the other direction.

## 2.4 Finite State Machine

Our robot can be controlled by humans wirelessly or track a specific object (e.g. human face) by itself. The overall finite state machine is shown in Fig.9. Initially, it is in the connecting state. After connecting with the HC-05 module, it will be controlled by a joystick through bluetooth. If the connection is lost, the robot will go back to the connecting state. If the controller pushes the Button A, it will start detecting a specific object, and it will keep tracking the object autonomously until it becomes close enough to the object or it cannot find it. In this project, the object is set as humans' face.

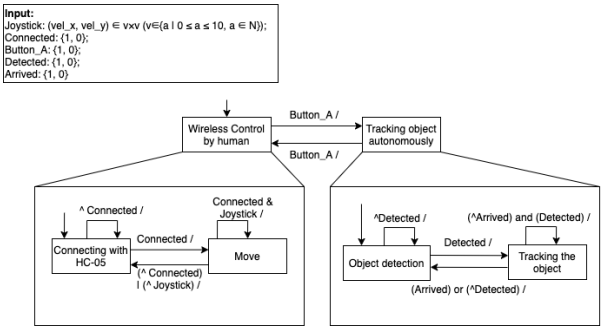


Fig.9 Finite state machine

The program of object tracking progress and the FSM of obstacle avoidance will be specified in the following parts. Since the robot should always avoid collision, the obstacle avoidance can be run by a thread.

# 3. Capabilities

## 3.1 Object tracking

With the basic motion control and the camera, the hexapod robot can identify the position of the object, turn its direction and follow the object, which is the human face in our project. To do that, we also need to detect the face with the help of opencv. The hexapod robot has a server, which runs on the Raspberry Pi OS, and a client, which can be launched on our own computer. The client will send commands to the server and control the robot, it can also show the data sent from the server such as video stream data. Fig.10 shows the interface of the client receiving video stream data transmitted from the server.

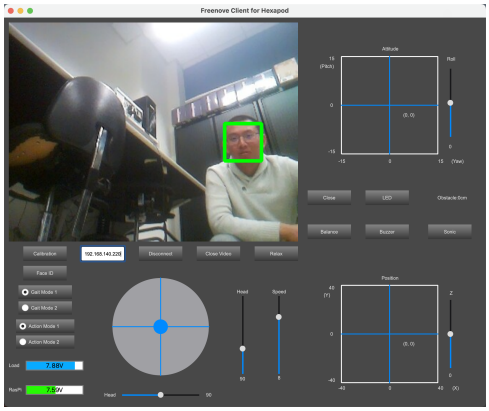


Fig.10 The client interface

Based on the motion control, we can easily send commands to the robot and make it move in any direction and turn both sides.The logic of face tracking is shown in Fig.11. We start a thread on the client side to keep detecting the position of the human face and send commands to control the server. If the face is on the left side of the video window, the robot needs to turn left and point right to the human face. Then if it is on the right side, as shown in Fig 10, we need to turn to the other side. Otherwise, the robot can simply move forward and approach our target.

However, the actual implementation doesn't show the exact result we want. We find that the latency of video transmission is high, and the image captured from the video stream is often delayed for a few seconds. That's why in the video the robot would rotate for a few seconds, its movements are based on the image it detected seconds before. We believed this is because the Raspberry Pi 3B+ doesn't have enough power to send real-time video stream data and tried to fix this problem by only using the server side to reduce the cost of transmitting. Unfortunately, it didn't solve the problem either, so the transmitting cost is not the major problem, Raspberry Pi 3B+ with 1G memory just couldn't deal with real-time video. So the video shows that this capability has a little delay.

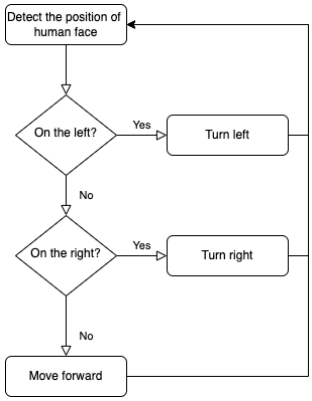


Fig.11 Object tracking process

## 3.2 Bluetooth Control

In order to allow humans to control the robot to move around, we use a joystick for the input of direction and velocity, and then transfer the data to the robot through bluetooth.

The joystick consists of two rheostats. Both of them provide a number ranging from 0 to 1023, which will be mapped to -10 to 10 as the velocity along x and y axes through a map function. The direction can be calculated in 2 ways. The first method is to let the robot always faces forward, so the direction of the robot is calculated by:

$$\alpha = tan^{-1}(\frac{v_y}{v_x})$$

where α is the angle between the direction and x axis and $v_x$ and $v_y$ are the velocity along the x and y axes. The other method is to use the α as the angular velocity, which allows the robot to change the direction it will face. These two ways can be shifted by the switch in the joystick.

The data transmission is based on the classic bluetooth transmission protocol because the transmission speed is faster than BLE. Therefore, we use HC-05 bluetooth module and the rfcomm package in the Raspberry Pi system for receiving messages. The numbers in the messages are extracted by using the re package in Python. To make the robot respond to each command more quickly, a thread is created to let the robot move towards a direction for a short time after receiving a message.

## 3.3 Obstacle Avoidance

With the basic motion control, we can use python threading to do obstacle avoidance. We use thread1 to process the recovery logic when the robot encounters an obstacle, and thread2 to keep receiving the data from the ultrasonic sensor, and thread3 to send the command to go forward.

The basic logic of obstacle avoidance is shown in Fig.12 as a finite state machine. The three threads start at the same time. Initially the robot goes forward, and if the data from the ultrasonic sensor shows that there is an obstacle less than 10 cm from the robot on the front, it will first go backward for a const period of time and then turn around to find a direction that has no obstacles, and this recovery process is done by thread1.

So far, with multithreading, the robot can finish the task of obstacle avoidance well.

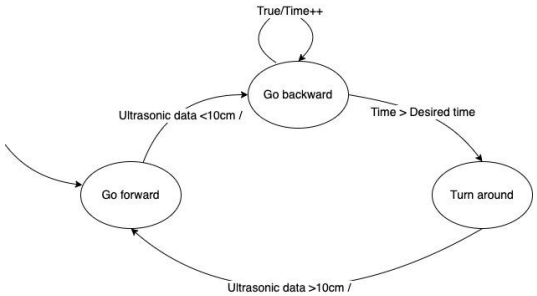


Fig.12 Finite State Machine for Obstacle Avoidance

# 4. Course topics

Table.1 Related course topics in each capability

| | Sensors & actuators | FSM | Multitasking |
|---|---|---|---|
| Object tracking | The server uses the camera to capture video data. | The movement of the robot is based on the position of the face, which changes between turn left, turn fight and move forward. | The client launches multiple threads to detect faces and send commands to the server. |
| Bluetooth control | It uses a joystick as an input device to control the motors on the robot to move around. | The movement of the robot is based on the input value of the velocities along x and y axes. It can move towards a direction with or without rotation. | It launches a main function and a thread. The main function gets the data from the messages by bluetooth. The thread controls the robot to move towards a direction for a short time after receiving a message. |
| Obstacle avoidance | It uses the ultrasonic sensor to detect obstacles. | The basic logic can be shown as a FSM, which describes the task transformation of the robot. | It uses three threads to let the robot move forward, receive ultrasonic data, and do the recovery after encountering obstacles. |

# 5. Challenges

In the first capability of object tracking, we found that the Raspberry Pi 3B+ couldn't handle real-time video data efficiently and bring latency to face detection. We have tried our best to reduce the latency in different ways but the problem still remains. This capability would definitely be more fluent with the help of the GSI. Besides, somehow the voltage provided by the battery is unstable, that's why sometimes the robot in the video moves really slow and even falls on the ground.

In the capability of obstacle avoidance, we didn't encounter much trouble without the GSI. The only thing is that it may be more convenient for us to get one or two more ultrasonic sensors to detect obstacles in other directions from the GSI.

In addition, the bluetooth connection is not stable. Although the HC-05 is paired and connected with the Raspberry Pi board, it often disconnects by itself and sometimes receives nothing. Sometimes it doesn't allow reconnection, and the reasons are various. It may be because a connection port cannot be released, or because of some errors from the package bluez. Looking for possible solutions on the Internet is time-consuming. In most cases, the only way to solve the problem is to reboot the system.

# 6. Future

Based on what we have done so far, we can extend this project in several aspects. First, for the capability of face detection, we can use deep learning models specifically designed for embedded systems, such as MobileNet, which raises the accuracy of face detection. Second, one ultrasonic sensor is not enough for perfect obstacle detection, we can use more sensors to detect obstacles in all directions. Finally, we would like to redesign the gait algorithm to make the robot move in four legs, leave the two front legs carrying stuff like boxes and give the robot the capability of transportation.

# Acknowledgement

Though the strike has influenced this course, we still want to give our appreciation to the professor and GSIs. This project can be done without their effort through this whole semester. Alvin discussed the project idea with us, Shishir also helped us in machine learning algorithms on embedded system and Xiaoman gave a lot of support on hardware.

# Reference

[1] Seide, K., Faschingbauer, M., Wenzl, M.E., Weinrich, N. and Juergens, C. (2004), A hexapod robot external fixator for computer assisted fracture reduction and deformity correction. Int. J. Med. Robotics Comput. Assist. Surg., 1: 64-69. https://doi.org/10.1002/rcs.6