ECE452C_Fall-2018
Project-02_Team-7

Team member:

Yunjuan Wang

Yinbin Ma

Qiteng Wu

Video Link: https://www.youtube.com/watch?v=8YHw5gn9G5k

Task allocation:

1. Image capture: Yinbin Ma
2. Image analysis and processing: Qiteng Wu
3. Robot moving: Yunjuan Wang
4. Debug the code and do the experiment: Yunjuan Wang, Yinbin Ma, Qiteng Wu

Difficulties:

1. It takes significant trial and testing to find out the proper thresholds for robot moving in one iteration.
2. We tried to let one wheel move forward and let the other wheel move backward but failed. Thus, we make the assumption that the coordinates remain the same when the robot rotate.
3. During the test, our robot sometimes will not stop and cannot detect light, and we find the reason is that the count of wheel spindle doesn't work at times, this hard ware problem bothers us for a long time.
4. Finding a suitable angle of camera plays a crucial role in this experiment, otherwise it cannot capture the right images to detect brightness.

Steps of tuning the sensors:

Since only our right infrared sensor works well and there is something wrong with our left infrared sensor, we only detect if there is obstacle in front of our right infrared sensor.

Code & Comments:

1. Configure Environment

utils.py

Function: Robot configuration for the convenience of writing code calls later.

```
1.  import RPi.GPIO as GPIO
2.  import picamera as pc
3.  from AlphaBot import AlphaBot
4.  import numpy as np
5.
6.  Ab = AlphaBot()
7.  Ab.stop()
8.
9.  S1 = 27
10. S2 = 22
11. DL = 19
12.
```

```
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(S1, GPIO.OUT)
GPIO.setup(S2, GPIO.OUT)
GPIO.setup(DL, GPIO.IN, GPIO.PUD_UP)

PS1 = GPIO.PWM(S1, 50)
PS1.start(50)
PS2 = GPIO.PWM(S2, 50)
PS2.start(50)

def set_servo1(angle): # vertical
    PS1.ChangeDutyCycle(12.5 - 10.0 * float(angle) / 180)

def set_servo2(angle): # horizontal
    PS2.ChangeDutyCycle(12.5 - 10.0 * float(angle) / 180)
```

2. Image capture:

capture.py

Function: Capture a specific number *n* of images. Return these images.

```
from utils import pc, set_servo1, set_servo2, np
import time
weight = np.array([0.2126, 0.7152, 0.0722])

def init():
    set_servo2(0)
    time.sleep(0.5)

def capture(n):
    r = pc.PiResolution(480, 320).pad()
    imgs = np.zeros([n, 480*320], np.uint8)
    c = pc.PiCamera(resolution=r)
    try:
        for i in range(n):
            set_servo2(180/(n-1)*i)
            time.sleep(0.5)
            tmp = np.zeros(480*320*3, np.uint8)
            c.capture(tmp, 'rgb')
            tmp = np.uint8(np.dot(tmp.reshape([-1, 3]), weight))
            imgs[i, :] = tmp
    finally:
```

```
        c.close()
    return list(imgs)
```

3. Image analysis and processing:

light.py

Function: Calculate the sum of pixel value in every image. Return the angle that the sum of pixel value in that image is maximum.

```
from utils import np

def Light(imgs):
    maxNum = 0
    indexMax = 0
    for index, img in enumerate(imgs):
        count = np.sum(img)
        if count > maxNum:
            maxNum = count
            indexMax = index

    return indexMax*(180/(len(imgs)-1))
```

4. Robot moving:

lightmove.py

Function: If the robot turns left, right wheel moves and left wheel stops. If the robot turns right, left wheel moves and right wheel stops. Record the distance of the corresponding wheel when robot rotates based on the sensor of our speeding test module. Given a specific angle, we can calculate the distance that the corresponding wheel has to move. Then the robot goes straightforward in a specific distance and stop. The robot will repeat from capturing images to adjust moving direction until it detects obstacle (we assume the light source is an obstacle) and stop. We use infrared sensor to detect obstacles. Since only our right infrared sensor works well and there is something wrong with our left infrared sensor, we only detect if there is obstacle in front of our right infrared sensor. We print out the coordinates when the robot stops.

```
from __future__ import print_function
import math
import time
import sys
from utils import Ab, GPIO

cntl = 7
cntr = 8
EncL = 0
EncR = 0
DL = 19

def updateEncoderL(channel):
    global EncL
    EncL += 1

def eprint(*args, **kwargs):
    print(*args, file=sys.stderr, **kwargs)

def updateEncoderR(channel):
    global EncR
    EncR += 1

GPIO.setup(cntr, GPIO.IN)
GPIO.setup(cntl, GPIO.IN)
GPIO.add_event_detect(cntr, GPIO.BOTH, updateEncoderR)
GPIO.add_event_detect(cntl, GPIO.BOTH, updateEncoderL)

def detect_obst():  # detect light source as obstacles.
    DL_status = GPIO.input(DL)
    return DL_status == 1
def followlightmove(angle):
    global EncL, EncR
    ini_encl = EncL
    ini_encr = EncR
    speed = 30

    print("Starting.")

    time.sleep(0.5)
    d = 5  # the distance between two wheel
    threshold = 3 * math.pi
    k = 0.2

```

```
    if angle > 90:
        Ab.setMotor(0, speed)
        count = 0
        while True:
            distance_l = float(EncL - ini_encl) / 40 * 2 * math.pi * 1.5
            if count % 200 == 0:
                eprint("l:", distance_l)
            if distance_l >= d * abs(angle - 90) * 2 * math.pi / 360:
                break
            count += 1
        Ab.stop()

    else:
        Ab.setMotor(-speed, 0)
        count = 0
        while True:
            distance_r = float(EncR - ini_encr) / 40 * 2 * math.pi * 1.5
            if count % 200 == 0:
                eprint("r:", distance_r)
            if distance_r >= d * abs(angle - 90) * 2 * math.pi / 360:
                break
            count += 1
        Ab.stop()

    af_encl = EncL
    af_encr = EncR

    Ab.setMotor(-speed, speed+2)

    count = 0
    straight_distance = 0  # distance that the robot move straightforward.

    while detect_obst():
        straight_distance = float(EncL - af_encl + EncR - af_encr) / 2 / 40 * 2 * math.pi *
    1
        if count % 100 == 0:
            eprint("straight:", straight_distance)
        count += 1
        if straight_distance > threshold:
            break

    Ab.stop()

    y = math.cos(angle * 2 * math.pi / 360) * straight_distance
```

```
88.     x = -math.sin(angle * 2 * math.pi / 360) * straight_distance / 1.5
89.     return x, y
```

Test:

app.py

```
1.  import capture
2.  import light
3.  import lightmove
4.  import time
5.  import math
6.  import numpy as np
7.
8.  x, y = 0, 0  # initialize the original coordinates.
9.  n = 7
10. # try:
11. while lightmove.detect_obst():
12.     capture.init()
13.     img = capture.capture(7)
14.     angle = light.Light(img)
15.     print angle
16.     time.sleep(1)
17.     dx, dy = lightmove.followlightmove(angle)
18.
19.     x += dx  # update coordinates.
20.     y += dy
21.
22.     print "angle: {} degree\nx: {}\ny: {}".format(angle, dx, dy)
23. #
24. # except KeyboardInterrupt:
25. #     lightmove.Ab.stop()
26.
27. angle = math.atan2(y, x)
28. # calculate SE(2) based on angle and moving distance in x-axis and y-axis
29. se2 = [math.cos(angle), -math.sin(angle), x,
30.        math.sin(angle), math.cos(angle), y,
31.        0, 0, 1]
32.
33. se2 = np.array(se2).reshape([3, 3])
34.
35. print "SE(2) for flashing light:\n", se2
```