# Program Analysis for Obstacle Avoidance

## 1. File Path

The program corresponding to this lesson is stored in: **/home/pi/MasterPi/Functions/VisualPatrol.py**

## 2. Performance

After the game starts, the distance between MasterPi and measured object will be shown on the live camera feed. When this distance is less than or equal to 30cm, the car will turn to the left; when the distance is greater than 30cm, the car will keep moving forwards.

## 3. Program Analysis

---

**Note: Before modifying the program, it is necessary to back up the original file. Only after that, proceed with the modifications. Directly modifying the source code files is strictly prohibited to avoid any errors that could lead to the robot malfunctioning and becoming irreparable!!!**

---

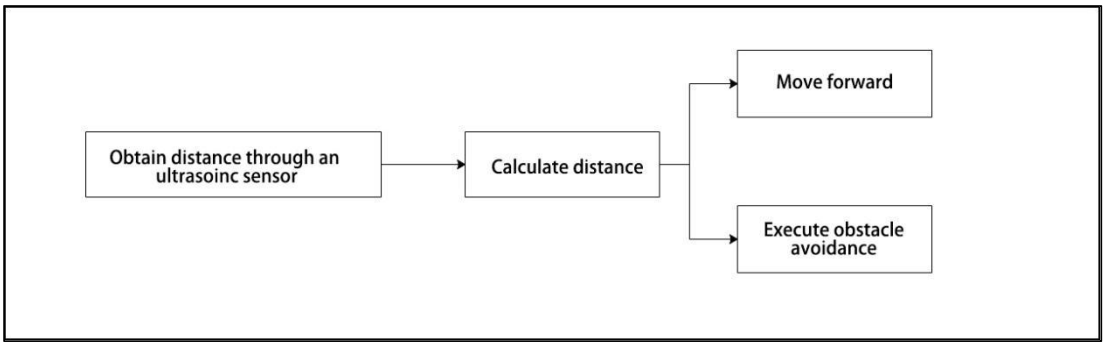
### 3.1 Import Parameter Module

| Import module | Function |
| --- | --- |
| import sys | Importing Python sys module is used for getting access to the relevant system |

| | |
|---|---|
| | functionalities and variables. |
| import cv2 | Importing OpenCV library is used for functionalities relayed to the image processing and computer vision. |
| import time | Importing Python time module is used for time-related functionalities,such as delay operations. |
| import math | Importing Python math function is used for mathematics operations and functions. |
| import HiwonderSDK.Board as Board | Importing board library is used for controlling sensor. |
| import numpy as np | Importing numpy library and renaming it as “np” for performing array and matrix operations |
| from HiwonderSDK.Misc as Misc | Importing Misc module is used for processing therectangular data identified. |
| import threading | Provide an environment for multi-thread running |
| import Camera | Importing Camera library for the use of camera. |
| PID | Import the PID class from the armpi-pro module. This is used to implement PID control algorithm. |

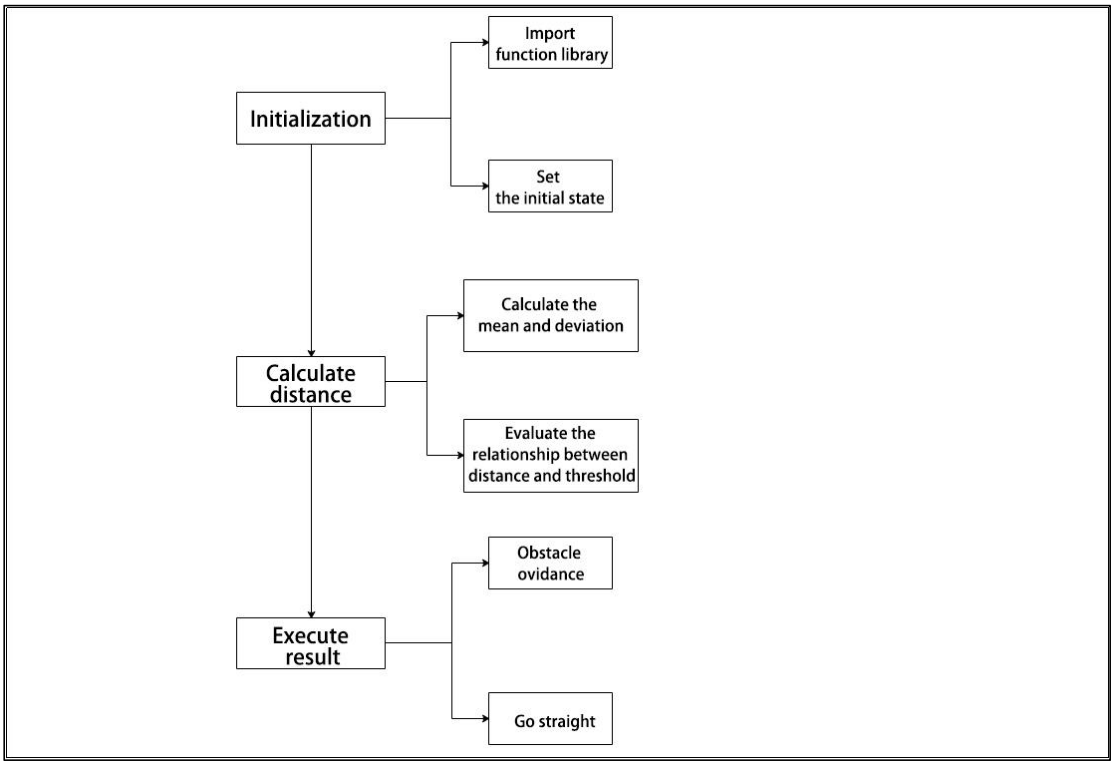
| | |
|---|---|
| from ArmIK.Transform import * | Used for functions related to the transformation of the robotic arm's pose. |
| from ArmIK.ArmMoveIK import * | Used for functions regarding to inverse kinematics solution and control. |
| import yaml_handle | Contain some functionalities or tools related to handling YAML format file. |
| import signal | Used for receiving and processing signals |

## 3.2 Program Logic




The distance between MasterPi and obstacle is obtained through ultrasonic module, then perform an action based on the calculated distance between the robot and the module. When the distance is less than a certain set value, the car will execute obstacle avoidance; when it is greater than another set value, the robot will continue moving forward.

## 3.3 Program Logic and Corresponding Code Analysis




From the above flow diagram, it is primarily used for initialization, distance calculation and result execution. The following content are edited based on this program flow diagram.

### 3.3.1Initialization

- **Import function library**

You need to first import the function library during the initialization.Regarding the content imported, please refer to “3.1 Import Parameter Module”.

```
import sys
sys.path.append('/home/pi/MasterPi')
import cv2
import time
import signal
import Camera
import numpy as np
import pandas as pd
import HiwonderSDK.Sonar as Sonar
import HiwonderSDK.Board as Board
from ArmIK.Transform import *
from ArmIK.ArmMoveIK import *
import HiwonderSDK.mecanum as mecanum
```

- **Set the initial status**

After the initialization is complete, it is necessary to set the initial state. This includes configuring the initial position for the servo following, the threshold of the obstacle avoidance, etc.

```
39  # 初始位置
40  def initMove():
41      chassis.set_velocity(0,0,0)
42      Board.setPWMServoPulse(1, servo1, 300)
43      AK.setPitchRangeMoving((0, 6, 18), 0,-90, 90, 1500)
```

```
99  # 设置避障速度
100 def setSpeed(args):
101     global speed
102     speed = int(args[0])
103     return (True, ())
104
105 # 设置避障阈值
106 def setThreshold(args):
107     global Threshold
108     Threshold = args[0]
109     return (True, (Threshold,))
110
111 # 获取当前避障阈值
112 def getThreshold(args):
113     global Threshold
114     return (True, (Threshold,))
115
```

## 3.3.2 Calculate Distance

- **Calculate the mean and standard deviation**

The data.copy() and data_.std() functions from the pandas library are used. With these two functions, we can calculate the mean and standard deviation of the imported values which correspond to the distance between the robot and obstacles.

```
135     dist = HWSONAR.getDistance() / 10.0
136
137     distance_data.append(dist)
138     data = pd.DataFrame(distance_data)
139     data_ = data.copy()
140     u = data_.mean()  # 计算均值
141     std = data_.std()  # 计算标准差
142
143     data_c = data[np.abs(data - u) <= std]
144     distance = data_c.mean()[0]
145
146     if len(distance_data) == 5:
147         distance_data.remove(distance_data[0])
```

- **Determine the relationship between distance and threshold**

During the judgement relationship, the most commonly used statement is if statement.

In the relationship determination, the commonly used statement is the "if" statement. Here, it is used to check whether the current distance meets the threshold set during the initialization process. Based on the result, different actions can be executed subsequently.

In this context, the value "30.0" inside the "Threshold" parameter corresponds to the threshold of 30 cm.

```
27  Threshold = 30.0
```

```
149     if __isRunning:
150         if speed != old_speed:   # 同样的速度值只设置一次
151             old_speed = speed
152             chassis.set_velocity(speed,90,0)
153 
154         if distance <= Threshold:   # 检测是否达到距离阈值
155             if turn:
156                 turn = False
157                 forward = True
158                 stopMotor = True
159                 chassis.set_velocity(0,90,-0.5)
160                 time.sleep(0.5)
161 
162         else:
163             if forward:
164                 turn = True
165                 forward = False
166                 stopMotor = True
167                 chassis.set_velocity(speed,90,0)
```

## 3.3.2 Control Execution and Obstacle Avoidance

After completing the distance detection and determination, the next step is to execute actions based on the judgment result. This program is calling the functions from the library file "**HiwonderSDK.mecanum**" to perform the actions.

```
def set_velocity(self, velocity, direction, angular_rate, fake=False):
    """
    Use polar coordinates to control moving
    motor1 v1|  ↑  |v2 motor2
             |     |
    motor3 v3|     |v4 motor4
    :param velocity: mm/s
    :param direction: Moving direction 0~360deg, 180deg<--- ↑ ---> 0deg
    :param angular_rate:  The speed at which the chassis rotates
    :param fake:
    :return:
    """
```

```
    if __isRunning:
        if speed != old_speed:   # 同样的速度值只设置一次
            old_speed = speed
            chassis.set_velocity(speed,90,0)

        if distance <= Threshold:   # 检测是否达到距离阈值
            if turn:
                turn = False
                forward = True
                stopMotor = True
                chassis.set_velocity(0,90,-0.5)
                time.sleep(0.5)

        else:
            if forward:
                turn = True
                forward = False
                stopMotor = True
                chassis.set_velocity(speed,90,0)
```

Based on the different distances, there are two corresponding actions: turning and execution. The execution is demonstrated using the function chassis.set_velocity(speed, 90, 0).

The first parameter "speed" represents the linear speed value.

The second parameter "90" denotes the direction angle, which can be adjusted within the range of 0 to 180 degrees.

The third parameter "0" represents the angular rate, controlling the speed at which the chassis rotates.