基于距离差的方向跟踪

李琦　浙江省诸暨牌头中学
谢作如　浙江省温州中学

相关学科：数学、技术、物理

当前，在很多领域中都需要用到方向跟踪，如拍摄某一运动物体时需要将摄像机实时对准被摄物体；在自动驾驶系统中需要确定前方障碍物的方向；在自动化武器的火控系统中也需要确定目标的方向；等等。

方向的确定有很多种方法，其中有一种方法就是通过距离差，根据平面几何原理计算出被测物体与观察者正前方视线的水平夹角。如图1所示，角θ就是测物体与观察者正前方视线的水平夹角。

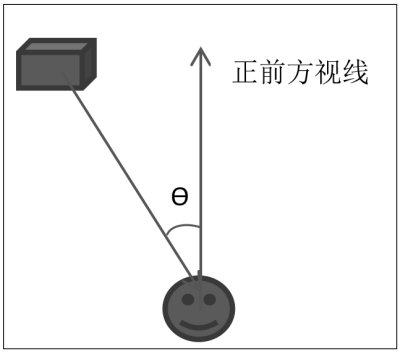

图1

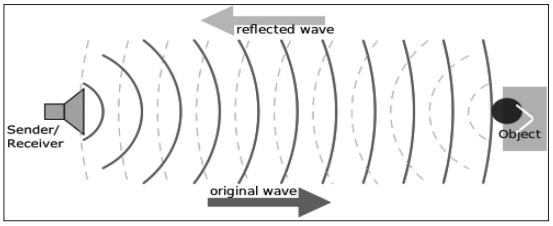

图2

距离测量工具：超声波测距传感器

人可以听到的声波频率范围大约是20Hz到20kHz，低于20Hz的声波称为次声波，超过20kHz的声音称为超声波，超声波遇到障碍物时容易被障碍物反射回来。超声波测距传感器就是通过发射40kHz超声波同时接收被测物体反射的超声波，并根据发射与接收的时间间隔来计算出被测物体的距离，如图2所示。

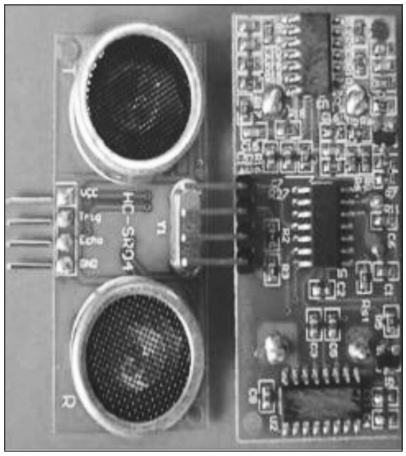

图3

假设t0时刻超声波测距传感器向被测物体发出超声波，t1时刻接收到被物体反射的超声波，则时间间隔Δt=t1−t0；设声音在空气中传播速度为v，则可以计算出被测物体的距离d为：d=(Δt*v)/2。

市面上所售的超声波测距传感器一般有4个针脚，分别为VCC、GND、TRIG、ECHO，图3所示的是HC−SR04超声波测距模块。

使用时先将trig和echo端口都置低，然后向trig端发送持续时间至少10us的高电平脉冲，模块会自动向外发送8个40kHz的超声波方波。接着捕捉来自echo端的高电平持续时间，这个时间间隔就是超声波在空气中运行的时间。

根据HC−SR04的测距原理，我们可以写出如下Arduino超声波测距函数：

```
float ultrasonic_distance(int trig,int echo){ //定义端口号
    float distance;
    long pulseIntime;
    pinMode(trig,OUTPUT);
    //设置trig口为输出模式
    pinMode(echo,INPUT);
    //设置echo口为输入模式
    digitalWrite(trig, LOW);
    //将trig设置为低电平
```

```
delayMicroseconds(2);
    //保证trig口稳定为低电平
digitalWrite(trig,HIGH);
    //将trig设置为高电平
delayMicroseconds(10);
    //并使高电平持续10微秒
digitalWrite(trig,LOW);
    //10微秒后，trig置回低电平
    pulseIntime=pulseIn(echo,HIGH,30000);//检测echo出现的高电平时间，
    //如果30000微秒内没有出现高
    //电平则返回0，
    //考虑声音来回以及HC—SR0
    //的测量最大距离约4.5米，那
    //么30000微秒时间如果没有出
    //现echo信号，则说明超过了测
    //量距离或者测量失败了。
    if(pulseIntime==0){
//如果超过测量距离或者测量失败
    distance=-1;  //则定义距离为-1
    {else{          //否则，计算公式如下：
    //测量距离（毫米=持续时间（微秒）/1000(微秒/毫秒)*340(毫米/毫秒)/2//化简后得：测量距离（毫米）=持续时间（微秒）*0.17
        distance=pulseIntime*0.17; }
    return distance;
    }
```

以上函数的两个参数分别为trig口引脚号和echo口引脚号，并且近似地取声速为20℃时较为典型的340m/s，该函数的返回值就是传感器与被测物体的距离（单位为毫米）。

方向测量的数学几何模型

超声波测距传感器虽然能检测出传感器与被测物体的距离，但却无法知道被测物体的方位，对于传感器而言，物体有可能分布在它可测夹角内以距离为半径的弧的任意位置，如图4所示。

但是，如果我们考虑在一条直线上放置两个一定间距的超声波测距传感器，如图5所示。不难发现，两条弧最多只有一个交点，而被测物体就在这个交点上。我们可以将问题抽象为一个平面几何问题，如图6所示。

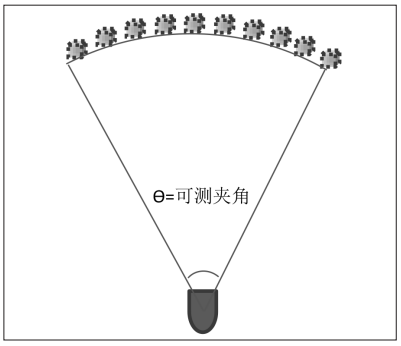

图4

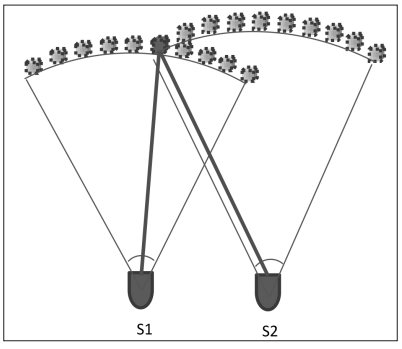

图5

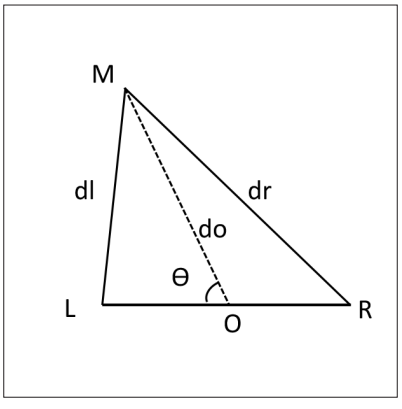

图6

在图6中，O点为线段LR的中点并且已知LR的长度为l，两个超声波测距传感器分别放置在L点和R点上，并且对同一物体M进行测量，测得的距离分别是dl和dr，如果能计算出θ的角度，便可以知道方向。

余弦定理是描述三角形中三边长度与一个角的余弦值关系的数学定理，是勾股定理在一般三角形情形下的推广。如图7所示，是余弦定理分析。

在我们所测量的几何模型中，对于△LOM，我们已知LM和LO两边的边长分别为dl和l/2，如果我们再已知OM的边长do，则可以通过余弦定理计算出以下数值：

$$\cos(\theta)=(do^2+(\frac{l}{2})^2-dl^2)/(2*do*(\frac{l}{2}))$$

那么$\theta=\arccos((do^2+(\frac{l}{2})^2-dl^2)/(2*do*(\frac{l}{2}))$

而对于OM的边长do，则可以通过平行四边形四边对角线平方和定理求出。对于一个平行四边形而言，其两条对角线的平方和等于四边的平方和。在我们的平面几何模型基础上，作OM的反向延长线

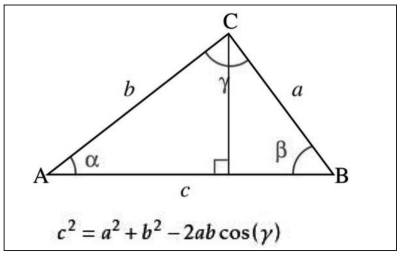

图7

ON，使得OM=ON，则根据平行四边形（如图8）四边对角线平方和定理，我们得出：

$l^2+(2*do)^2=2*dl^2+2*dr^2$，整理后可得：$do=\frac{\sqrt{2*dl^2+2*dr^2-l^2}}{4}$。

根据余弦定理推导的公式 $\Theta=\arccos((do^2+(\frac{l}{2})^2-dl^2)/(2*do*(\frac{l}{2}))$ 我们便可以计算出θ的值。

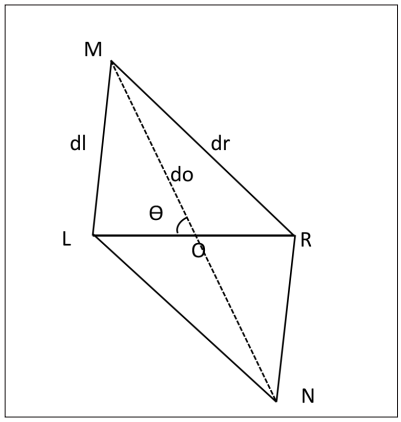

图8

结合Arduino实现方向的跟踪

用两个超声波测距传感器SL和SR测量同一个物体的距离，经过计算得出舵机的偏转角度，进而实现方向跟踪。我们根据前面推导的公式编写如下的自定义函数，即：

```
float get_degree(float distance_l,float distance_r,float sensor_distance){
float x;
float l;
x=sensor_distance/2;
l=sqrt((2*(pow(distance_l,2)+pow(distance_r,2))-pow(2*x,2))/4);
return acos((pow(x,2)+pow(l,2)-pow(distance_l,2))/(2*x*distance_l))*180/PI;
}
```

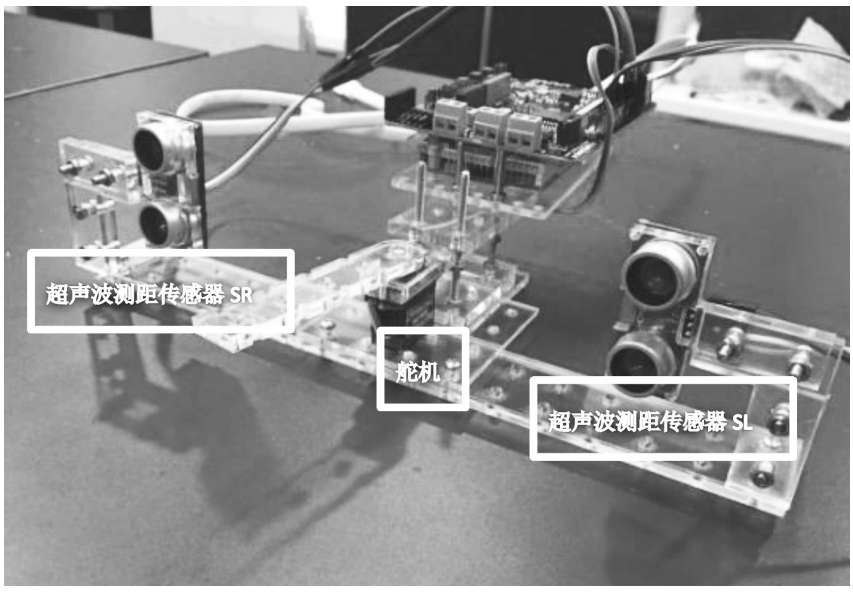

图9

上述自定义函数中，需要的三个参数分别就是SL的测量值、SR的测量值以及SL和SR的中心点距离，在图9中，这个距离为165毫米。函数的返回值为弧度值。

由于舵机输出的是角度值的整数，而上述函数计算的结果是弧度，因此还需要在主函数中将弧度值转换为角度值并取整：

```
degree=int(get_degree(distance1,distance2,SENSORDISTANCE)+0.5)
```

根据舵机的运行方向，在图中的实例中，最终舵机输出的角度为θ的补角，即：

```
s1.write(180-degree);
```

下面给出完整的Arduino代码并附上简单注释：

```
#include"Servo.h"
//需要用到舵机库和数学函数库
#include "math.h"
Servos1;
//定义舵机对象s1
#define SERVOPIN 11
//定义宏：舵机所连接的数字口
#define SENSORDISTANCE 165
//定义宏：两个超声波测距传感器的中心点距离
int degree;
void setup() {
Serial.begin(9600);      //打开串口，便于看到计算结果
s1.attach(SERVOPIN);
}
void loop() {
float distance1=ultrasonic_distance(5,6);//超声波测距传感器SL连接在5,6号数字口
float distance2=ultrasonic_distance(7,8);//超声波测距传感器SR连接在5,6号数字口
Serial.print(distance1);
Serial.print("  ");
Serial.print(distance2);
Serial.print("  ");
```

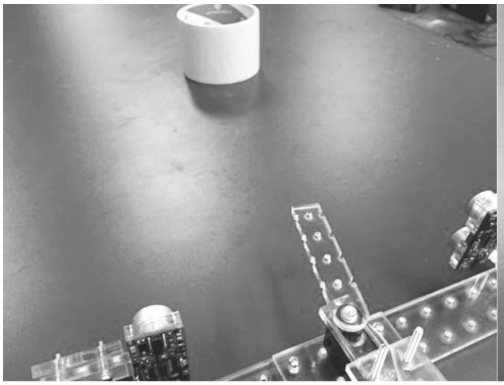

图10

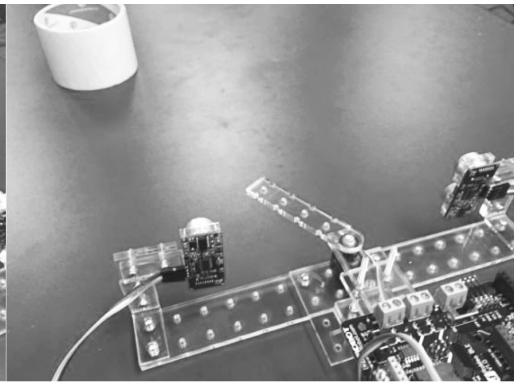

图11

```
        //串口打印出两个传感器的测量值便于查看
    if (int(get_degree(distance1,distance2,SENSOR.DISTANCE)+0.5)!=0 ){
    //有可能会测量失败，如果某个传感器测量失败，则最终计算结果会是0
        degree=int(get_degree(distance1,distance2,SENSOR.DISTANCE)+0.5);
    //如果计算结果不为0，将角度值更新给degree变量
    }
    Serial.println(degree);
    s1.write(180-degree);
    delay(100);
    }
    float get_degree(float distance_l,float distance_r,float sensor_distance){
        float x;
        x=sensor_distance/2;
        float l;
       l=sqrt((2*(pow(distance_l,2)+pow(distance_r,2))-pow(2*x,2))/4);
        return acos((pow(x,2)+pow(l,2)-pow(distance_l,2))/(2*x*distance_l))*180/PI;
    }
    float ultrasonic_distance(int trig,int echo){//超声波测距子函数
        float distance;
        long pulseIntime;
        pinMode(trig,OUTPUT);
        pinMode(echo,INPUT);
        digitalWrite(trig, LOW);
        delayMicroseconds(20);
        digitalWrite(trig,HIGH);
        delayMicroseconds(20);
        digitalWrite(trig,LOW);
         pulseIntime=pulseIn(echo,HIGH,30000);
        if(pulseIntime==0){
            distance=-1;
        }else{
    distance=pulseIntime*0.34/2;
        }
        return distance;
    }
```

我们可以看一下效果（如图10、图11）：舵机上连接的一条亚克力条会动态地指向前方的障碍物。

拓展思考

利用两个超声波测距传感器，结合数学公式研究方向追踪是一个有趣的STEAM项目。虽然我们已经成功地实现了方向追踪，但还是存在一些问题，还需要进一步研究：①因为超声波测距传感器测量障碍物的距离并不是很稳定，而上述程序只是简单地把不合理的值忽略掉，因此会出现舵机抖动的情况，我们可不可以进一步分析测量值，如从一段时间的大量测量值中过滤出较为正确的测量结果？②超声波测距传感器的测量角度并不是很大，因此如果障碍物的方位过于偏向左右，则会出现某个传感器检测不到障碍物的情况，我们可以怎样改进，如调整超声波测距传感器的方向是否有用？

如果对相关内容感兴趣，请关注主持人博客。