

DIY_3D 项目总结

目录

DIY_3D 项目总结

目录

项目概述

模块介绍

Arduino 编程

opencv 编程

三角测距

meshLab

Libigl 编程

存在问题

实验心得

其他备注

项目概述

本项目主要工作是搭建一台三维扫描仪，对模型进行扫描，得到其点云数据及 mesh 图像。然后可以根据得到的数据打印出原模型，实现一个闭环。

涉及到的知识有：Arduino 编程、Libigl 编程、OpenCV、MeshLab 等。

具体运行情况如下：

利用 Libigl 编写的主界面担当着与用户交互的任务，同时也是集成各部分功能的平台。主界面有四个按钮：First Scan、Second Scan、Third Scan、Point Cloud。前三个按钮分别控制三次扫描过程（舵机无法最大旋转角度为180°，故需要旋转三次方能得到完整的信息），最后一个按钮用于显示最后得到的点云图像。

当三个扫描按钮按下时，程序一方面向 Arduino 板发送信号，一方面控制相机拍照。Arduino 板接受到信号后，开启激光灯，控制舵机顺时针 180° 旋转，同时相机在这段时间内拍照并处理图片，即将其二值化处理并得到图片中的激光点坐标（二维）。旋转至180°后，激光灯关闭，相机停止拍照，舵机复位至 0°，然后停止转动，为下一次扫描做准备。

三次扫描结束后，得到三个分别记录各次扫描得到的二维点坐标，然后根据三角测距算法，将其转换为三维坐标点集，最终输出为一个三维坐标点集，一方面此数据可以在 Libigl 编写的主界面中显示出来，另一方面可以通过 MeshLab 等软件处理，最终得到 mesh 图像。

模块介绍

Arduino 编程

本模块主要任务是接受来自程序的信号，然后据此控制激光器的开关和舵机的运动。以下根据程序介绍本部分原理。

定义：

```
Servo myservo;          //控制舵机的伺服函数
int pos = 0;             //舵机初始角度
int ledPin = 9;          //定义数字9 接口
#define BAUD 9600        //统一通信波特
String receivedString;
                        //通信字符串
bool value = false;      //是否进入扫描状态
```

setup() 函数：

统一通信波特，链接各引脚。

另，在调试过程发现：Arduino 一开始工作，舵机会立刻转一角度，即便其在 0° 位置。这是因为舵机有一初始角度。因此在此处预先使其停止在 0°（每次扫描后依旧停止在 0° 位置）：

```
void setup()
{
  Serial.begin(BAUD);      //统一通信波特
  pinMode(ledPin, OUTPUT); //定义小灯接口为输出接口
  myservo.attach(7);       //舵机链接 7 号引脚
  myservo.write(pos);      //舵机预先设置在 0° 位置
}
```

loop() 函数：

扫描状态：开启激光灯，舵机顺时针旋转 180°。转至180°后，一次扫描完成，舵机暂停旋转，手动调整模型角度（为下一次扫描做准备）。为避免调整位置时对数据产生的影响，关闭激光灯。然后舵机复位，停止在 0° 位置。

```
digitalWrite(ledPin, HIGH); //点亮激光灯

//角度从0到180度
for(pos = 0; pos < 180; pos += 1)
{
  myservo.write(pos);      // 舵机旋转角度为pos
  delay(125);              //延时125ms
}

digitalWrite(ledPin, LOW); //关闭激光灯
delay(80);

//从180°恢复到 0°
for(pos = 180; pos > 0; pos-=1)
{
  myservo.write(pos);
  delay(50);
}
```

```
//一次扫描结束后，最终停止在 0°位置  
pos = 0;  
myservo.write(pos);
```

关于 `loop()` 函数的最初设想：当接收到 "ON\n" 时，进入扫描状态，当接收到 "OFF\n" 时，关闭舵机和激光灯。原以为这种想法会使程序陷入 `loop()` 函数的死循环中，也即需要在最恰当的时候输入 "OFF\n" 方能推出扫描状态，但是：

```
if (Serial.available() > 0)    //返回串口缓冲区中当前剩余的字符个数
```

因此这种思路也是可行的！


程序中的思路为：设一布尔变量，初始设为假。当接收到 "ON\n" 时，此变量为真，为真时进入扫描状态，当一次扫描结束后再设为假，便不能进入扫描状态，只有当再次接收到 "ON\n" 时，才能进入扫描状态。

硬件编程，最重要的是简单！！

重点注意 `loop()` 中的循环能否跳出

opencv 编程

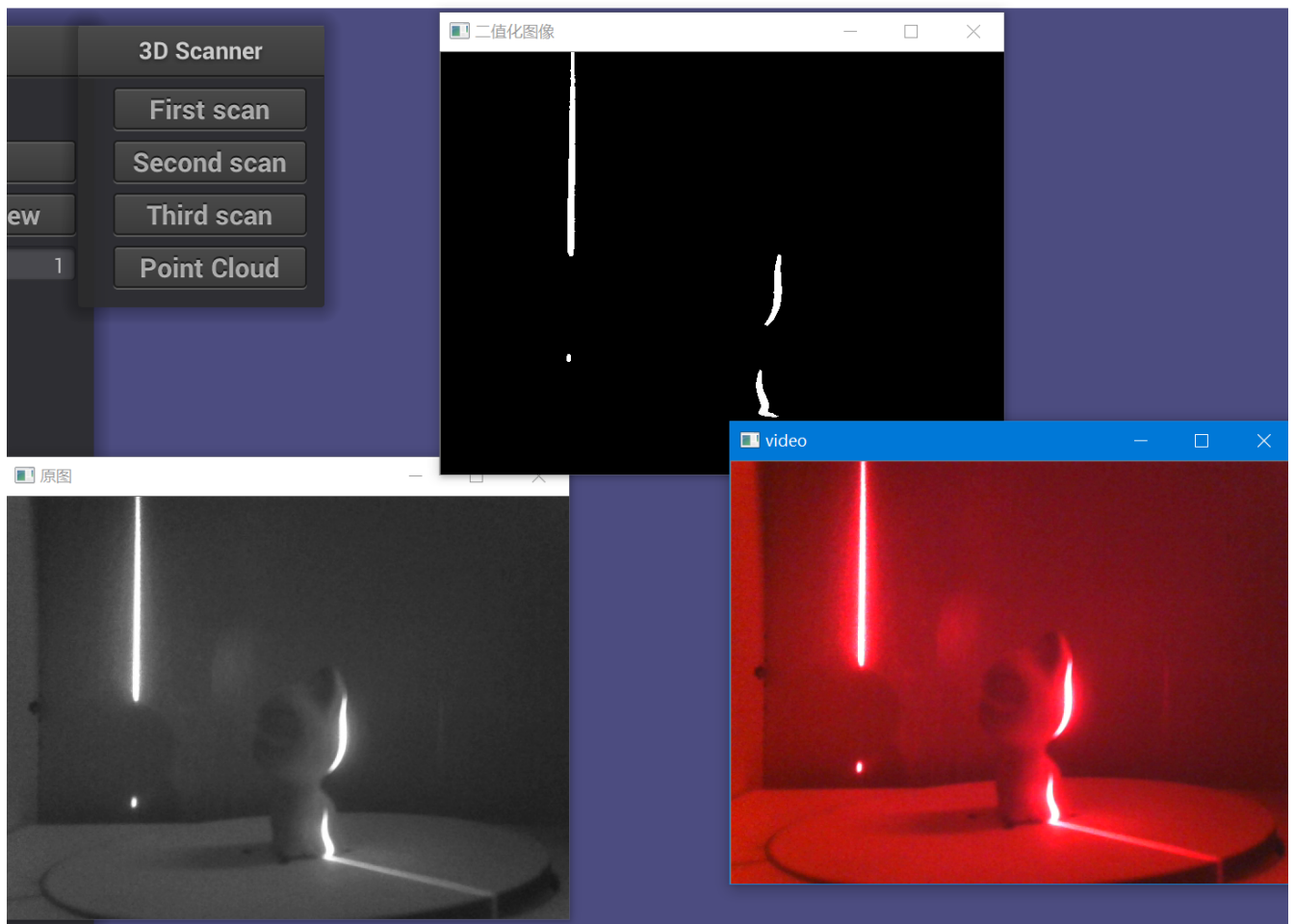
此部分实现的功能包括：控制相机在舵机顺时针旋转时拍照，并且将图片二值化处理，得到图片中的激光点坐标（二维），输出一个包含此次打印获取的坐标信息的 `test.txt` 文件，如下：

第一次扫描

具体代码见项目文件，俺不会，反正很牛/赞/赞/赞。

但是，在配置 [opencv](#) 的过程中明白了很多 `vs` 的功能：解决方案资源管理器、属性管理器、库目录等。

运行结果如下：



三角测距

具体代码见附件，俺原理都没看懂/傻/傻。

mashLab

原项目三维重建采用 `PowerCrust` 算法，但因时间紧迫改为 `mashLab` 操作，觉得操作界面太帅了，一看高大上啊/赞/赞/赞。

Libigl 编程

整合各部分代码。除集成控制相机拍照的代码，剩下部分包括界面设置、串口通信、读取文件显示点云三部分。

界面设置及串口通信如下：

```
arduino = new SerialPort(portName);

//Checking if arduino is connected or not
if (arduino->isConnected()) {
    std::cout << "Connection established at port " << portName << endl;
}

// Init the viewer
igl::viewer::Viewer viewer;
```

```

// Extend viewer menu
viewer.callback_init = [&](igl::viewer::Viewer& viewer)
{
    // Add an additional menu window
    viewer.ngui->addWindow(Eigen::Vector2i(210, 10), "3D Scanner");

    viewer.ngui->addButton("First scan", []() {
        //打开激光器、开启舵机
        arduino->writeSerialPort(ON, MAX_DATA_LENGTH);
        std::cout << "First scan!\n";
        //opencv控制摄像机拍照
        photo1();
    });

    viewer.ngui->addButton("Second scan", []() {
        arduino->writeSerialPort(ON, MAX_DATA_LENGTH);
        std::cout << "Second scan!\n";
        photo2();
    });

    viewer.ngui->addButton("Third scan", []() {
        arduino->writeSerialPort(ON, MAX_DATA_LENGTH);
        std::cout << "Third scan!\n";
        photo3();
    });

    viewer.ngui->addButton("Point Cloud", []() {
        //显示点云
        std::cout << "Point Cloud!\n";
    });

    // Generate menu
    viewer.screen->performLayout();
    return false;
};

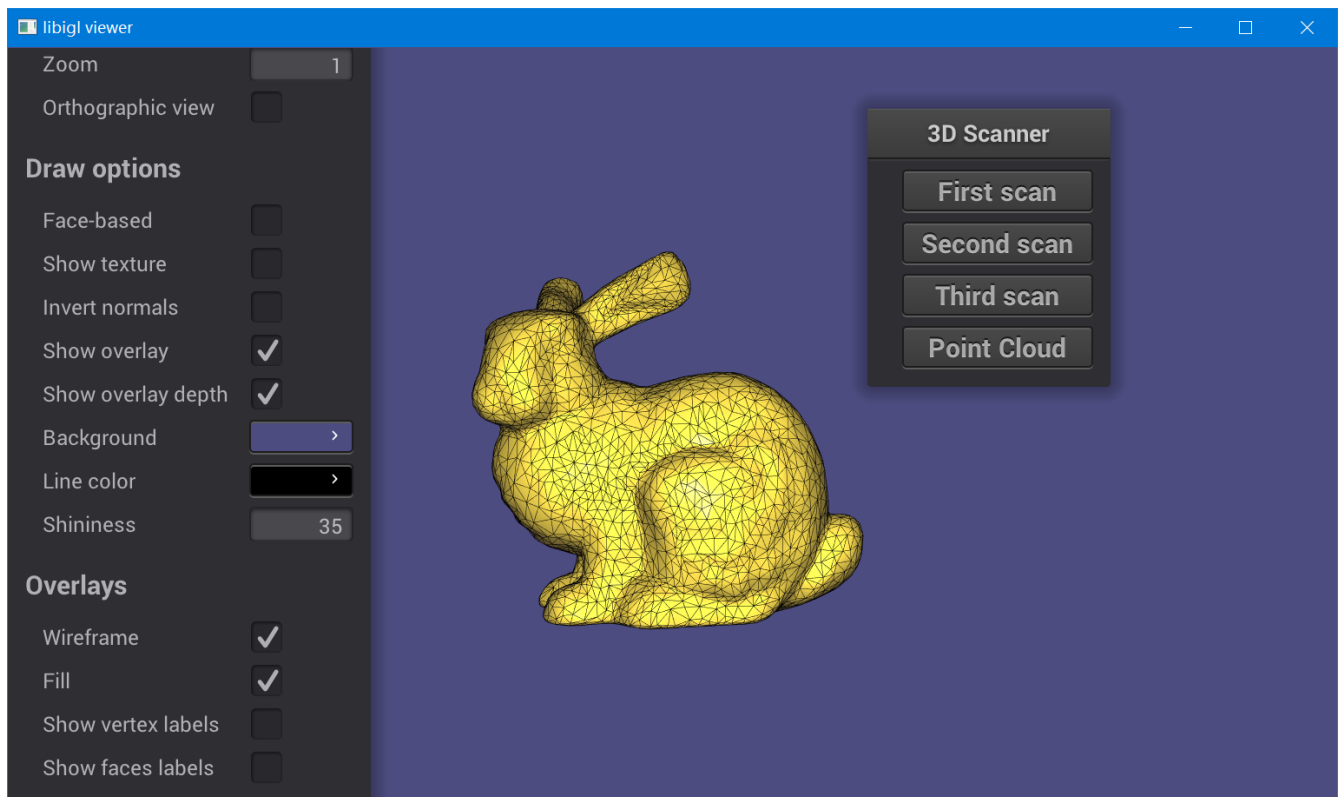
```

显示点云有两种方法，一种是根据已有的 `.off` 文件直接生成点云，`Demo` 中的兔子就是这样显示的：

```

igl::readOFF(TUTORIAL_SHARED_PATH "/bunny.off", V, F);

```



然而，meshLab 软件不能生成 libigl 所需的相适应的 .off 文件，因此只能作罢。

另一种方法是将所有的三维坐标信息写入 vector 容器中，然后进行如下操作：

```
igl::list_to_matrix(VVV, W);
viewer.data.add_points(W, Eigen::RowVector3d(1, 0, 0));
viewer.core.align_camera_center(W);
```

从文件中读入信息时，需要进行文件操作，具体思路是将文件中的数据读入 vector 容器 W 中，然后再将数据读入三维容器 vertex 中：

```
//通过读取txt文件获取三维坐标，将其存入矩阵中
//文件操作!!!
void show_pointCloud()
{
    vector<double> W;          //数据从文件读入这个容器中
    double d;
    VVV.clear();
    vector<double> vertex;
    vertex.resize(3);

    ifstream input("D:\\SD_DIY\\Photo2\\test.txt");    //读入
    string line;                                       //保存读入的每一行

    for (int a = 0; a < 280 * 3; )
    {
        while (getline(input, line))
        {
            d = stringToNum(line);
```

```

        w.push_back(d);
    }
}
input.close();

for(it = w.begin(); it != w.end(); )
{
    vertex[0] = *it;
    it++;
    vertex[1] = *it;
    it++;
    vertex[2] = *it;
    it++;

    vvv.push_back(vertex);
}
}

```

存在问题

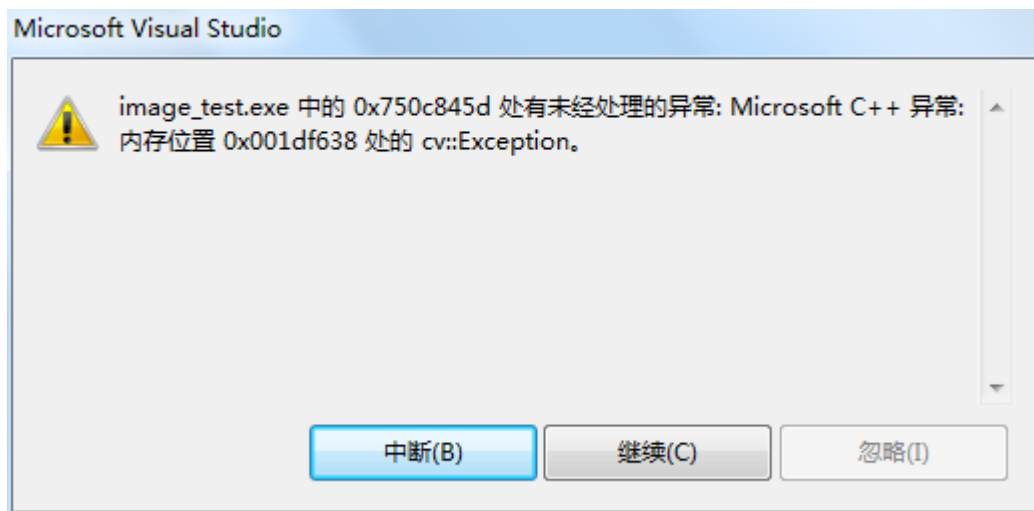
1. 相机启动问题：

程序运行时，第一次启动相机时，难以拍摄符合条件的 280 张图片。

因为一次程序运行需要三次扫描，也即需启动相机三次，而后两次拍摄均得达到符合条件的 280 张图片。

猜测可能是因为程序最初启动，第一次打开相机会有延迟。

2. opencv 本身 bug



出现这个错误时，点击 继续 可以继续运行，然而在这段时间内相机不会进行拍照。因此不能得到正常的280张图片。

3. 程序串行，不能实时显示画面

4. 三次手动 旋转矩阵

无论是舵机带动物体旋转还是手动使物体旋转，两者都会给物体一个角度变化，都有相应的旋转矩阵。因此可以在两次人为的旋转中，确定旋转的角度，也可确定旋转矩阵，即可得到一个点云，无须人工调整三个点云将其重合。

实验心得

其他备注
