2018全国大学生智能互联创新大赛

项目报告

参赛作品名称： 智能隧管检测系统

基于平台： STM32，树莓派等

作品简介：　目前的交通道路有朝着地下化、隧道化方向发展的趋势，地铁隧道也将成为未来城市交通的主要发展方向。对于隧道、管道等类似的封闭环境，现有的缺陷检测方法有着工序繁琐、耗时长等弊端，不利于城市交通系统的可持续发展。因此，本项目设计了一套智能隧管缺陷检测系统，在把对管道、隧道的缺陷检测统一化的同时，将成本控制在合理的范围之内，具有很强的实用性与可推广性。同时，本项目也提出了一种小直径隧管彩色三维重构方法。

目 录

[一、 摘要 3](#_Toc516757668)

[二、 项目背景 3](#_Toc516757669)

[三、 研究目的 3](#_Toc516757670)

[四、 设计方案 4](#_Toc516757671)

[1. 开发工具 4](#_Toc516757672)

[2. 设备硬件结构 4](#_Toc516757673)

[3. 控制系统设计 7](#_Toc516757674)

[4. 视觉识别算法 8](#_Toc516757675)

[5. 三维重构系统 9](#_Toc516757676)

[五、 创新点、实用性与应用前景 12](#_Toc516757677)

[六、 附录 13](#_Toc516757678)

[1. 实物照片 13](#_Toc516757679)

[2. 测试程序源代码 14](#_Toc516757680)

[3. 参考文献 14](#_Toc516757681)

# 摘要

　　目前的交通道路有朝着地下化、隧道化方向发展的趋势，地铁隧道也将成为未来城市交通的主要发展方向。同时各类城市管道的长度也在逐年上升。对于隧道、管道等类似的封闭环境，现有的缺陷检测方法成本高昂，形式单一，有着工序繁琐、耗时极长等弊端，不利于城市交通与管线系统的可持续发展。

　　因此，本项目设计了一套以图像识别与超声波测距为主要传感器，基于STM32与树莓派的小型智能化隧管缺陷检测系统，并通过ROS实现了设备与上位机的通信。我们在把对管道、隧道的缺陷检测统一化的同时，设计了一套离线作业系统，并力求在满足精确、高效的同时，将成本控制在合理的范围之内，进而使得其能够被大规模投放使用，因而具有很强的实用性与可推广性。

同时，本项目也提出了一种基于超声波传感器与摄像头相配合的小直径隧管彩色三维重构方法，并在进行数学推导与可行性论证之后，使用程序模拟实现了三维重构的效果。

关键词：智能交通、图像识别、ROS、超声波测距、三维重构

# 项目背景

随着国家经济的不断发展，为了满足日益增长的通行和城市发展需求，各类管网在近十年不断增长。据统计，截止到2017年，仅国内地铁管网长度就已经达到了世界前列，上海与北京更分列第一第二。

如此大规模的交通管网带来的不仅只有方便，也带来了大量的附加问题。其中，管道维护是目前很多部门关注的重点。以地铁管道为例，现有的管道检测技术无法准确检测出管道裂缝，更无法判断出管道损坏的位置的影响程度，目前大多使用人工肉眼检测，且管道条件复杂，人工检测危险重重，国内外多次出现管道工人意外身亡事故。为了解决这一问题，我们团队参照了现有的管道探伤技术，针对地下交通管道的相关特点，设计了一款针对管道检伤的机器人。

# 研究目的

我们的研究目的是设计一套小型智能化隧管缺陷检测系统：既能对摄像头传来的图像数据进行实时处理，也能够离线的在某区域的隧管内作业，并能准确识别出隧管壁上的裂缝与漏洞等危害安全隐患，从而确保城市地下隧管网络安全、平稳地运行于发展。

# 设计方案

## 开发工具



SolidWorks OpenCV ProcessOn



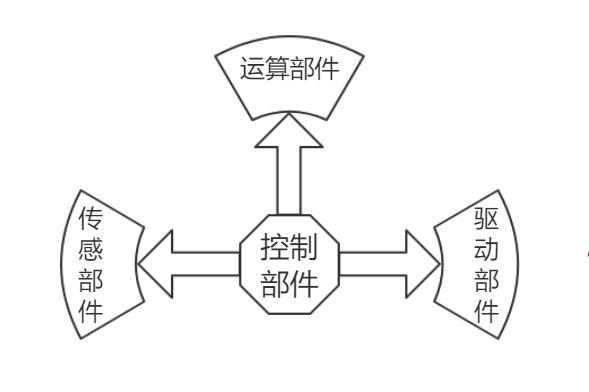
Keil Sublime Text PyCharm

在本项中，我们主要使用SolidWorks进行机械设计，并在图像处理的过程中调用了开源的计算机视觉库OpenCV。在程序编写方面，我们使用Keil来对STM32上的控制系统进行调试，随后使用Sublime Text与PyCharm编写了算法程序的主要代码。最后，我们使用ProcessOn制作了报告文档中各个系统和算法的示意图与流程图。

## 设备硬件结构

#### 硬件平台

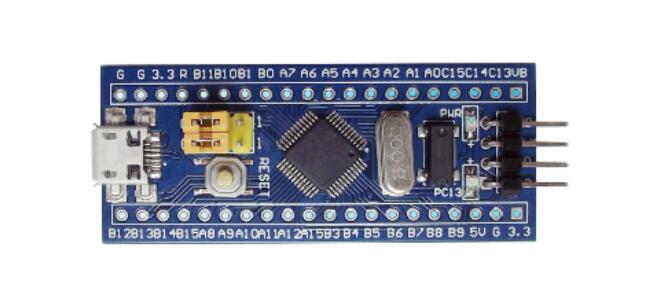
本项目设计的机器人在硬件的整体架构上采用了模块化设计的思想，因而具有很强的可扩展性，未来也将具备在多种环境中工作的能力。目前我们实现了其在一般隧道内进行检测的模块组合。



设备结构原理图

##### 控制部件

该模块采用的是STM32F103C8T6最小系统板。STM32是基于ARM® Cortex® M 处理器内核的 32位闪存微控制器，集高性能、高实时性、数字信号处理、低功耗于一身，基于行业标准，具有大量工具和软件可供选择，是本项目控制部件的的理想选择。



STM32F103C8T6

##### 驱动部件

该模块采用了通用减速直流电机与60mm直径的橡胶轮，具有强磁、带抗干扰等特性。我们会为其提供5V直流供电，使得机器人能在较平缓的隧道内稳步前行。



直流减速电机与轮胎

##### 传感部件

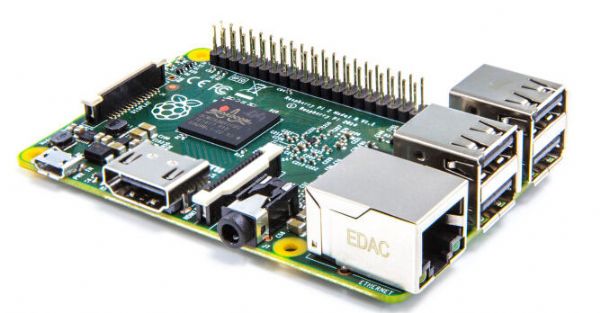
该模块采用的是带自感应式红外补光的RPiCamera摄像头。当环境光较弱时，其两侧的红外补光灯会自动开启，使得机器人能在各种不同光照环境下工作。RpiCamera的数据流可直接通过树莓派发送到上位机，从而实现实时检测的功能。



Rpi Camera

##### 运算部件

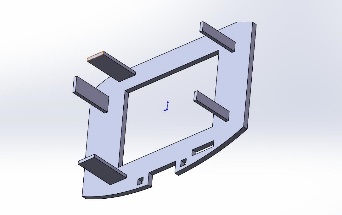
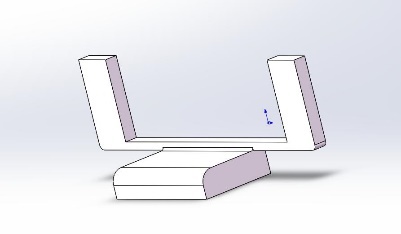
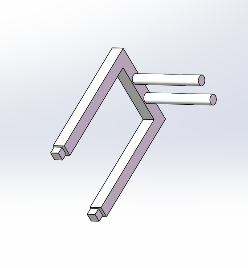
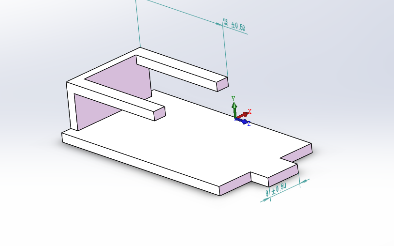
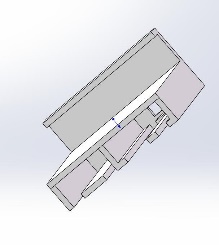
该模块采用了RaspberryPi3B来对摄像头获得的图形进行预处理与储存。RaspberryPi3B是基于ARM内核的性能强大的开源硬件，我们在其上安装了最新的轻量级操作系统Raspbian，使得我们能够非常容易地与其交互。



Raspberry Pi 3B

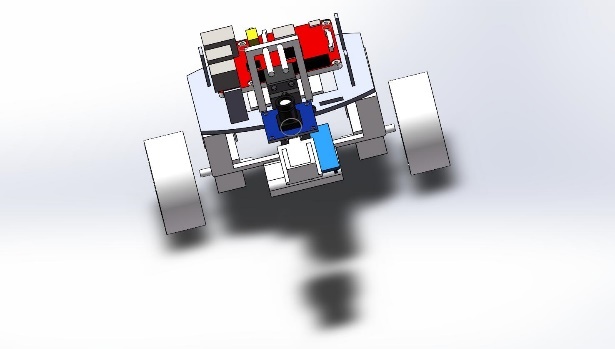
#### 结构件设计

考虑到本项目机器人工作原理的特殊性，我们选择通过3D打印技术自主设计车辆的结构件。

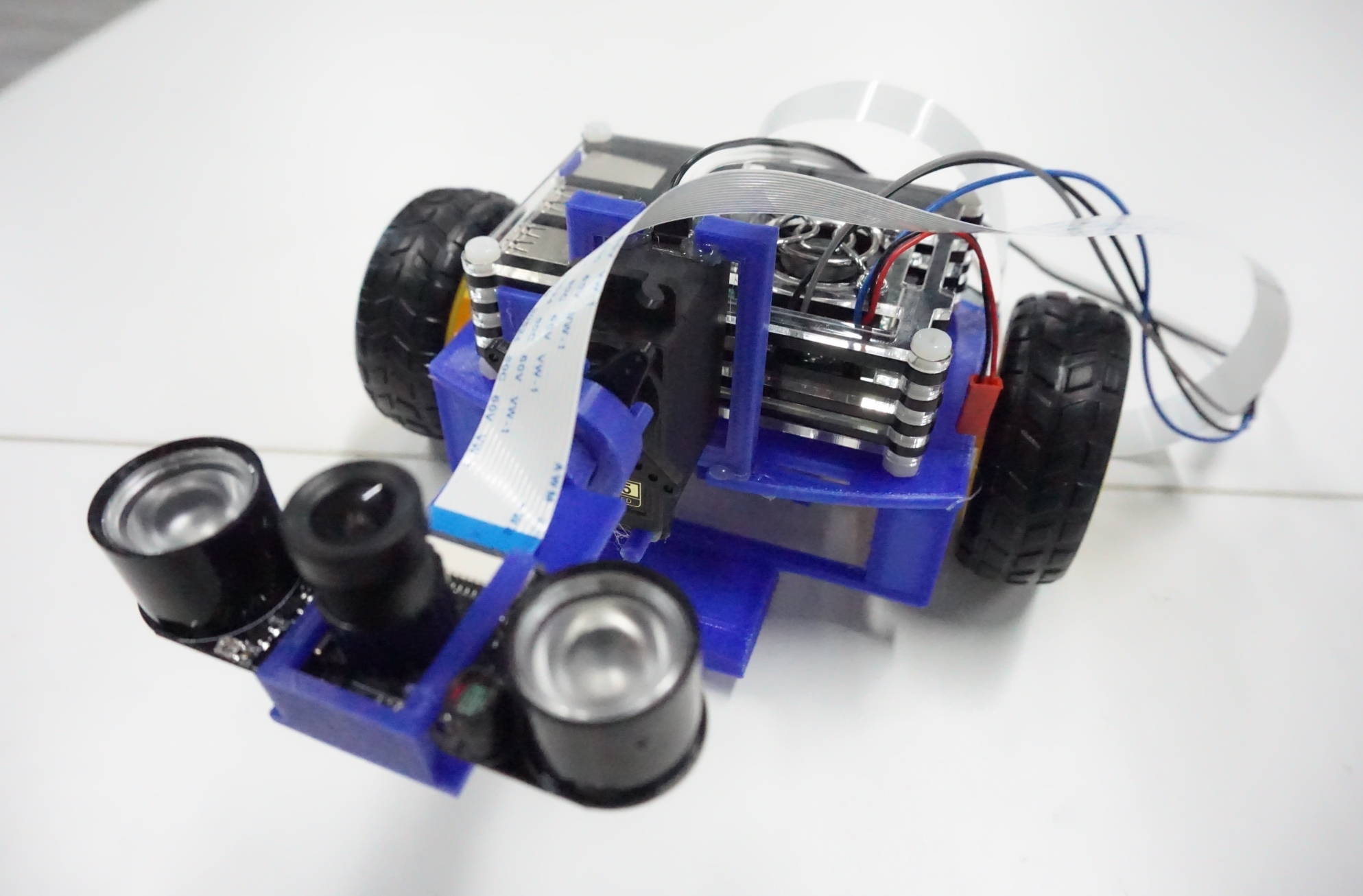


结构件图例

3D打印技术的灵活性使得本项目的机器人结构具有易于拼装、设计开放、适合快速迭代等特点，同时便于我们进行设计意图的表达。

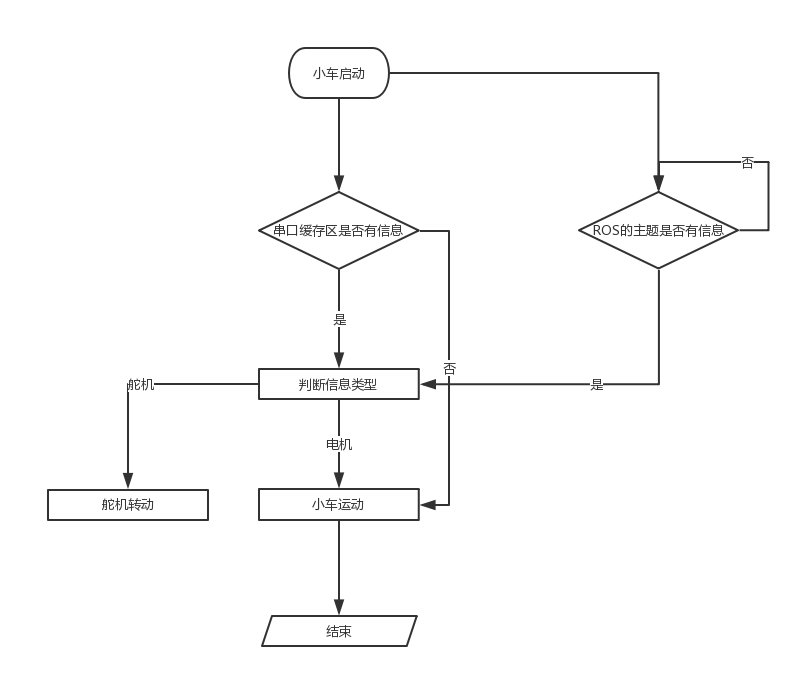


结构件拼接示意图



车体实物图

## 控制系统设计



控制系统流程图

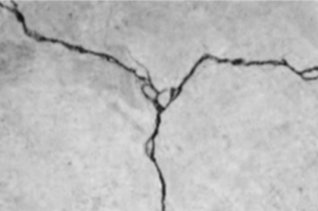
## 视觉识别算法

我们采用了基于视觉识别技术的缺陷检测算法，其原理是不断去除图像上的噪声，仅仅留下我们需要的特征也即裂缝等缺陷的轮廓。

#### 噪点消除

　　由于管道内复杂的环境与摄像头本身的物理特性，我们得到的初始图像数据往往带有较多噪点，这些噪点会影响成像的质量，进而干扰我们对缺陷特征的判断。因此，获取图像后我们首先要做的是使用滤波算法对图像去噪。

　　常用的滤波算法有均值滤波、高斯滤波、中值滤波等。经过我们多次尝试，在模拟的隧管环境下高斯滤波的去噪效果最好。除此之外它也有着较好的平滑化效果。



高斯滤波效果演示一

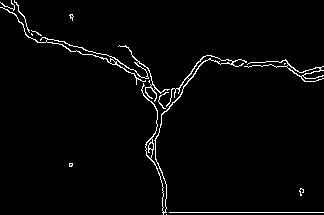
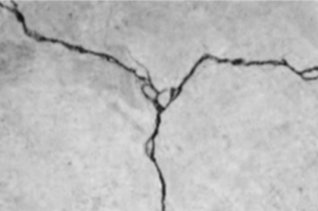


高斯滤波效果演示二

#### 缺陷检测：边缘检测

　　去除噪点之后，我们就可以对图像进行处理了。我们选用的主要缺陷检测手段是边缘检测算法。常用的边缘检测算法有Roberts算子，Sobel算子，Canny算子，形态学算子等等。

　　在对多种算子进行测试后，我们发现使用Canny算子进行腐蚀与膨胀运算后可以较好的检测出条状的裂缝，并且团状的潜在危险因素也能被呈现出来。



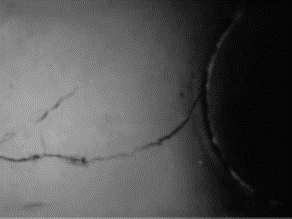
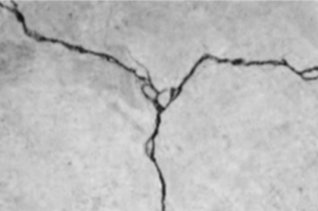
缺陷检测效果演示一



缺陷检测效果演示二

#### 缺陷检测：阈值分割

边缘检测算法之外，还有一类算法也能够较有效地检测出隧管壁上的缺陷，那就是阈值分割算法。阈值分割指以某一阈值为分界线将一幅图像按照灰度值转化成黑白二值图。我们将经过预处理的图像按灰度值分割成两个点集，当被分割成的两个点集之间方差最大时，该灰度值就是将图像二值化所需要的阈值。此处我们所采用的是一种自适应阈值分割算法：使用遗传算法求解的大津法(OTSU)。该方法耗时较长，可用于图像的离线处理过程之中。



使用遗传算法求解的大津法效果演示

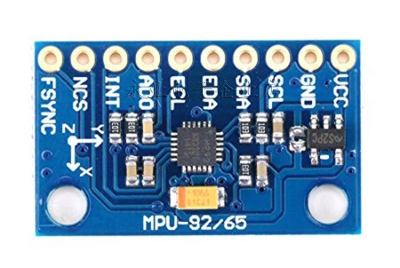
## 三维重构系统

#### 硬件平台

　　我们设计的三维重构系统需要一个类似于激光雷达的传感器。我们的方案是将超声波与姿态传感器被捆绑在一起，挂载到可以三百六十度旋转的舵机上。同时，我们使用低功耗蓝牙模块来实现短距离的数据传输，使树莓派能够收到来自超声波传感器的数据。

##### 九轴姿态传感器

我们准备使用MPU9250模块来获取超声波雷达的位姿。MPU9250能够实时测量其三轴上的加速度、角加速度与地磁角，从而精确地估计出雷达的姿态角。



MPU9250

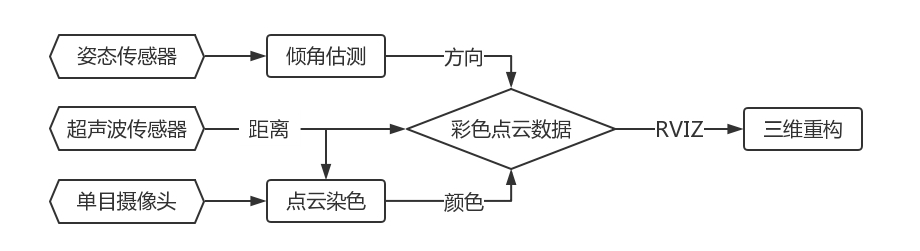
##### 超声波模块

为了提升距离的准确性，我们选用了US-100超声波传感器。US-100有着方向角较小、精度较大的特点，其探测范围可达2cm-450cm，而误差仅有0.3cm±1%，因而可以较好地检测出设备到管道内壁的距离。



US-100

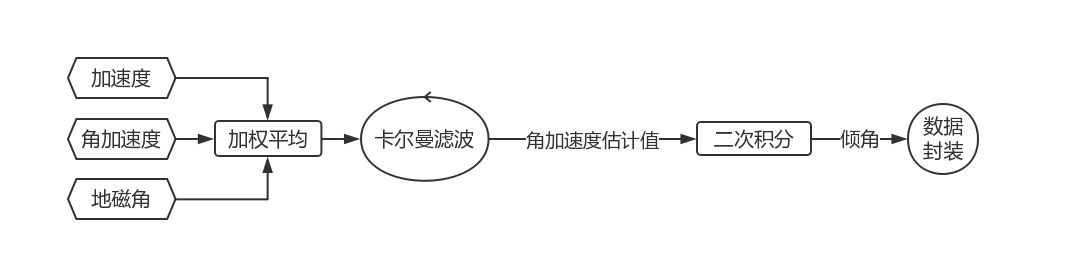
#### 设计思路



三维重构原理

#### 算法实现

##### 倾角估测



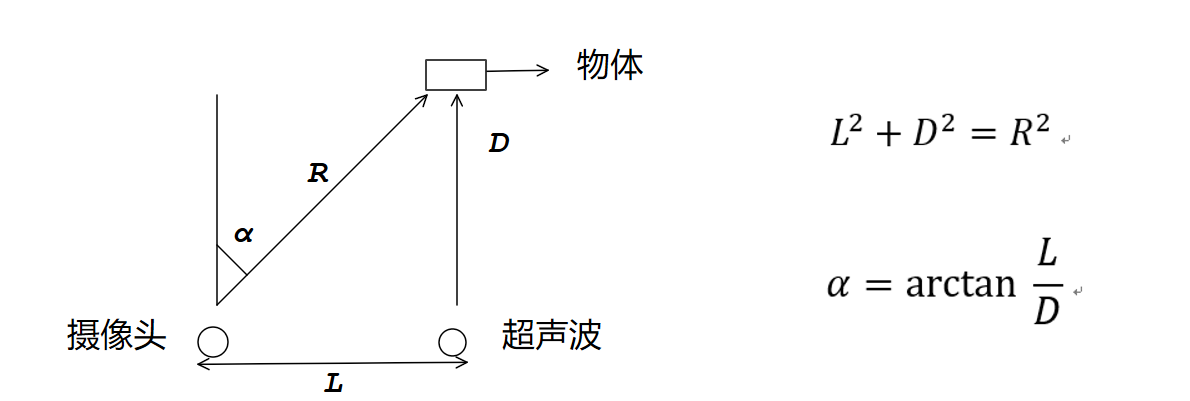
倾角估计算法原理



程序模拟的倾角估计生成的点云效果图

##### 点云染色

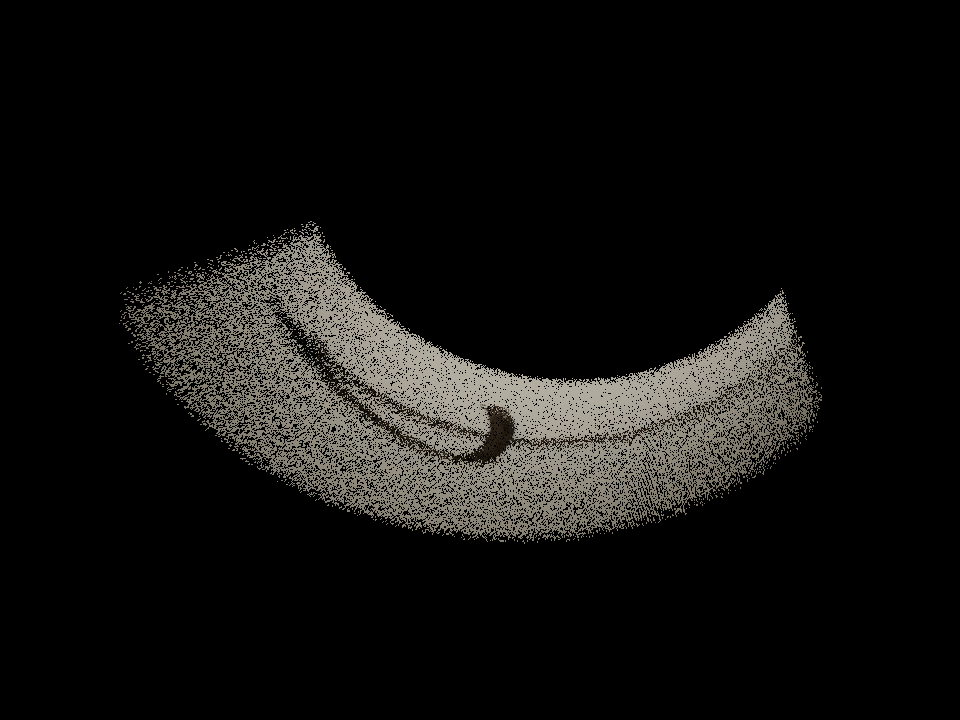
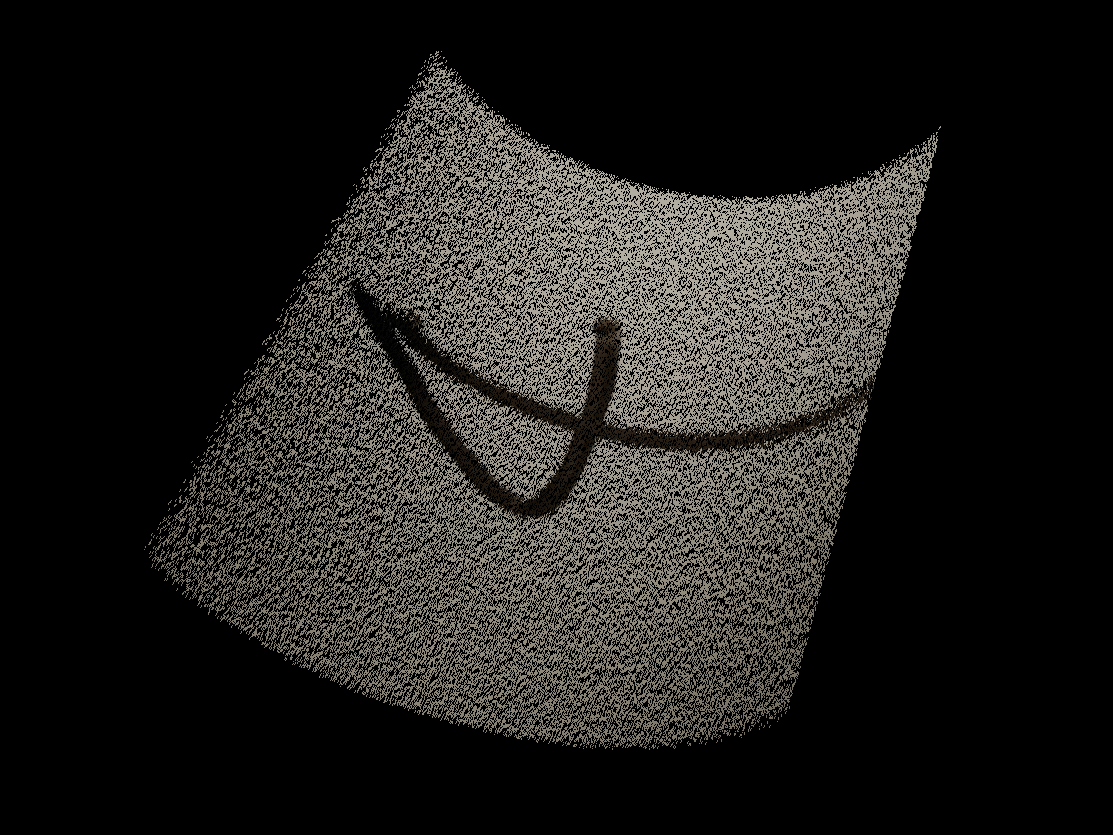
我们使用摄像头与超声波传感器之间的距离值L与点云到超声波传感器的距离D进行三角运算后可估算出物体在摄像机视场内与摄像机法线的夹角α，随后将其正切值与摄像机视角正切值的比例代入计算即可获得其在摄像机所获得的像素图上的具体位置，从而得到该点的像素值。



点云染色算法原理



管道内壁数字图案实拍照片



点云染色效果

##### 三维重构

　　通过将获得的彩色点云数据打包封装成ROS消息发布到ROS系统里，让RVIZ对其进行监听与处理，即可在RVIZ中看到彩色的三维重构结果。

# 创新点、实用性与应用前景

　　本项目的产品的硬件与相关算法均为自主设计，拥有很强的适应性与可扩展性，因而将能够面对复杂多变的各类隧管环境。且本项目的设备在其设计之初便预留较多的工业接口，可以安装多个特征传感器，从而实现“一次检测多种管道，一遍查出多种问题”，填补了此类产品的市场空白。

本项目已实现的管道缺陷检测方法主要依赖于对单目摄像头的图像进行处理，在成本和准确度上进行了取舍，有较强的可移植性和可扩展性。同时我们也采用了由现代生物进化论启发的遗传算法，从而能够使找到全局最优解的可能性大大增加。此外，通过边缘检测与阈值分割两种算法的配合，我们的检测方法能够较好的适应光照不均匀的情况，从而减少检测过程中产生的误差。

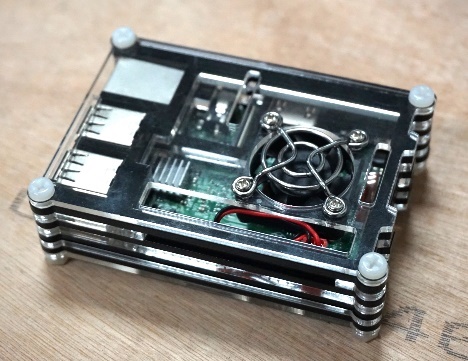
本项目提出了一种基于超声波的低成本三维重构思路，并实现了相关算法。该方案将能对管道内的环境进行较好的呈现。

本项目采用了分布式机器人操作系统ROS作为主体框架。ROS具有硬件抽象、底层设备控制、进程间与跨进程通信以及程序包管理等诸多功能，是时下新兴的智能机器人控制方案。我们实现了从STM32与树莓派端进行数据收集，随后发送至PC端进行处理，再经过PC端的数据处理与分析，将具体指令反馈给树莓派端的控制回路，从而实现了对机器设备进行精准合理的控制以及对数据的快速传输与储存。

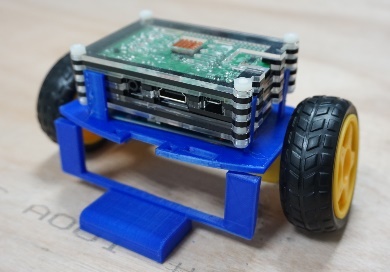
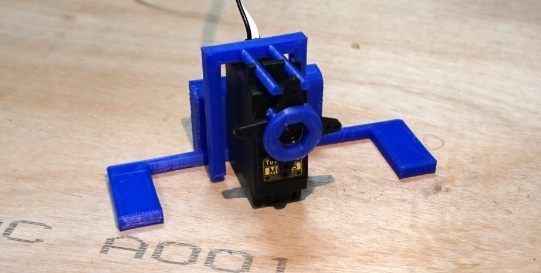
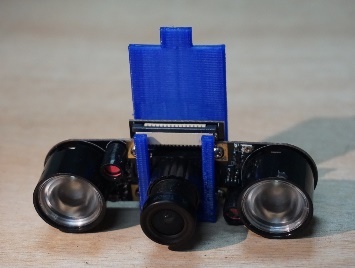
本项目将定制的跨平台的设备管理程序，以实现快速、准确的管道问题检测，并且能精准定位问题管道区域所在位置。未来还将提供依托设备管理软件提供的集群操作功能，单个工作人员即可完成大面积城市隧管网络检测工作，从而极大地减少了人力成本。

# 附录

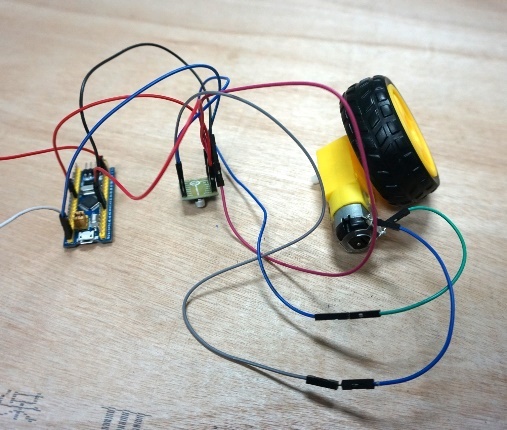
## 实物照片



树莓派裸板 套上了保护套的树莓派



摄像头支架演示、舵机支架与连接轴演示、车体结构演示



超声波模块调试 驱动部件调试



机器人调试演示

## 测试程序源代码

1. **def** genetic\_thres(image, k, m):
2. th = image
3. **for** i **in** list(range(image.shape[0])):
4. **for** j **in** list(range(image.shape[1])):
5. **for** t **in** list(range(1, len(k)-1)):
6. **if** k[t-1] <= image[i][j] < k[t]:
7. th[i][j] = int(k[t-1])
8. **return** th
9. image=thresh2
10. m = 3
11. items\_x = list(range(0, image.shape[0]))
12. items\_y = list(range(0, image.shape[1]))
13. random.shuffle(items\_x)
14. random.shuffle(items\_y)
15. x = items\_x[0:20\*m]
16. y = items\_y[0:20\*m]
17. seed = []
18. Var = 0.0
19. times = 0
20. k = 0
21. P, average = Hist(image)
22. **for** i **in** list(range(0, 20)):
23. code = int(0)
24. **for** j **in** list(range(0, m)):
25. code = code + gray[x[i\*j]][y[i\*j]] << j\*8
26. seed.append(bin(code))
28. **while** times < 2000:
29. Var = fitness(seed, P, average, m)
30. Next = wheel\_selection(seed, Var)
31. Next = Cross(Next, m)
32. Next = expand(Variation(Next), m)
33. seed = Next
34. times = times + 1
36. **for** j **in** list(range(len(Var))):
37. **if** Var[j] == max(Var):
38. k = getTh(Next[j], m)
39. **print** (k)
41. plt.subplot(131), plt.imshow(image, "gray")
42. plt.title("source image"), plt.xticks([]), plt.yticks([])

45. plt.subplot(132), plt.hist(image.ravel(), 256)
46. plt.title("Histogram"), plt.xticks([]), plt.yticks([])

49. th1 = genetic\_thres(image, k, m)
50. plt.subplot(133), plt.imshow(th1, "gray")
51. titleName = ''
52. **for** i **in** list(range(1, len(k)-1)):
53. titleName = titleName + str(k[i]) + ', '
54. titleName = titleName[:len(titleName)-2]
55. plt.title("threshold is " + titleName), plt.xticks([]), plt.yticks([])
56. plt.show()

视觉识别算法（节选）

1. #include <iostream>
2. #include <pcl/io/io.h>
3. #include <pcl/io/pcd\_io.h>
4. #include <pcl/visualization/cloud\_viewer.h>
5. #include <opencv2/opencv.hpp>
6. **using** **namespace** std;
8. **int** main()
9. {
10. pcl::PointCloud<pcl::PointXYZRGB> cloud;
11. pcl::PointCloud<pcl::PointXYZRGB>::Ptr cloud\_ptr(**new** pcl::PointCloud<pcl::PointXYZRGB>);
12. cv::Mat color = cv::imread("rgb.jpg");
13. cv::Mat depth = cv::imread("depth.png");
14. cv::Size dsize = cv::Size(color.cols, color.rows);
15. cv::Mat colored\_depth = cv::Mat(dsize, CV\_8UC3);
16. resize(colored\_depth, depth,dsize);
18. **int** rowNumber = color.rows;
19. **int** colNumber = color.cols;
21. cloud.height = rowNumber;
22. cloud.width = colNumber;
23. cloud.points.resize(cloud.width \* cloud.height);
25. **for** (unsigned **int** u = 0; u < rowNumber; ++u)
26. **for** (unsigned **int** v = 0; v < colNumber; ++v) {
27. unsigned **int** num = u\*colNumber + v;
28. cloud.points[num].b = color.at<cv::Vec3b>(u, v)[0];
29. cloud.points[num].g = color.at<cv::Vec3b>(u, v)[1];
30. cloud.points[num].r = color.at<cv::Vec3b>(u, v)[2];
31. cloud.points[num].x = u;
32. cloud.points[num].y = v;
33. cloud.points[num].z = ((**double**)depth.at<uchar>(u, v));
34. }
35. \*cloud\_ptr = cloud;
36. pcl::visualization::CloudViewer viewer("Cloud Viewer");
37. **return** 0;
38. }

三维染色算法

## 参考文献

1. 胡皙. 基于图像处理的地铁隧道裂缝检测技术研究[D]. 北京交通大学:胡皙, 2014.