基于互联网的摄像测量系统

摘要

本次参赛选题为基于互联网的摄像测量系统，采用树莓派 4b 作为开发板，利用以太网交换机在各个节点间传输数据。树莓派 4b 搭载单目摄像头通过 OpenCv 和机器强化学习对待测物体的轮廓、花纹特征和边长、位置进行识别；使用 Pillow 对待测物体进行几何空间的定位;两个摄像节点使用树莓派开发板并各自测量特定距离；利用交换机，传输距离数据至终端，终端树莓派开发板则对数据集进行空间坐标转换等计算，求出最终结果。经过多次测量分析，在角度任意时，针对 L 的计算精度较高，保持在 0.8cm 的误差以内。角度选取 180 或 0 度时，计算误差也能保持在 1.5cm 以内，测量角度时，利用数据拟合和机器学习算法，在 30-45 度时具有较高的准确率。同时，在考虑速度优先的状况下，L 误差也能维持在 2cm 以内。

一、 系统方案设计方案主要内容

1、 设计方案工作原理

• 预期实现目标定位

（1）摄像节点测量并显示待测物体的形状，轮廓，颜色，花纹，记录待测物体的移动轨迹。

（2）摄像节点可以自动追踪目标物体。

（3）终端可以通过交换机时时接受传输的视频图像等数据。

（4）终端采用树莓派去实现，对相关数据进行空间坐标等一系列计算，并求出最终结果。

• 技术方案分析比较

（1）YOLO 算法拥有极高的精度，SSD 算法在处理分析小尺度上的物体具有优势，同时速度极快，但卷积神经网络对 CPU 要求较高。

（2）findContours()函数难以识别复杂物体，但对于简单的物体轮廓具有极大的优势。

终端树莓派的 CPU 不能支撑起神经网络的计算，最终我们选择采用 opencv 自带的 findContours()函数去实现。

• 系统结构工作原理

整个系统各个节点统一使用树莓派 4b 开发板，摄像节点使用 90° 广角的摄像头，显示并时时传输视频。终端节点在接受各个节点的传输数据后，计算分析，输出结果。

就终端计算结果而言，我们选择使用长曝光的形式，根据摄像节点传输的视频进行灰度转换等格式处理，在利用终端的运算力，将转换好的视频进行长曝光，得到物体完整的移动轨迹图像，据此，便可以轻松分析出物体的摇摆的最低点，再依赖于空间坐标系，利用相似三角形等数学公式辅助，便可以求出绳长 L

• 功能指标实现方法

系统选择 90° 广角摄像机保证实时跟踪激光笔，并外接 HDMI 显示器，展示追踪视频，终端选择树莓派 4B，并外接灯光蜂鸣器等设备，提示测试完成。

• 测量控制分析处理

为保证测量的精度，有效数字近 8 位。并且多次测量解决手动测量的误差。无论是估值处理还是异常数据分析都有使用，大大减小了因误差带来的影响。

2、 核心部件电路设计

• 关键器件性能分析

树莓派 4B：基于 ARM 的微型电脑主板，以 SD/MicroSD 卡为内存硬盘，卡片主板周围有 1/2/4 个 USB 接口和一个 10/100 以太网接口。我们根据其性能搭载了最适合的程序，保证能流畅完美的运行

以太网交换机（D-Link DES-1024R）：速度 1000Mbps，完全满足系统的数据传输需要。

• 电路结构工作机理

后附

• 电路实现调试测试

图 1：电路实现调试测试图

• 关键电路驱动接口

GPIO 接口，SPI 串行外设接口，MiniHDMI 转 HDMI

• 核心电路设计仿真

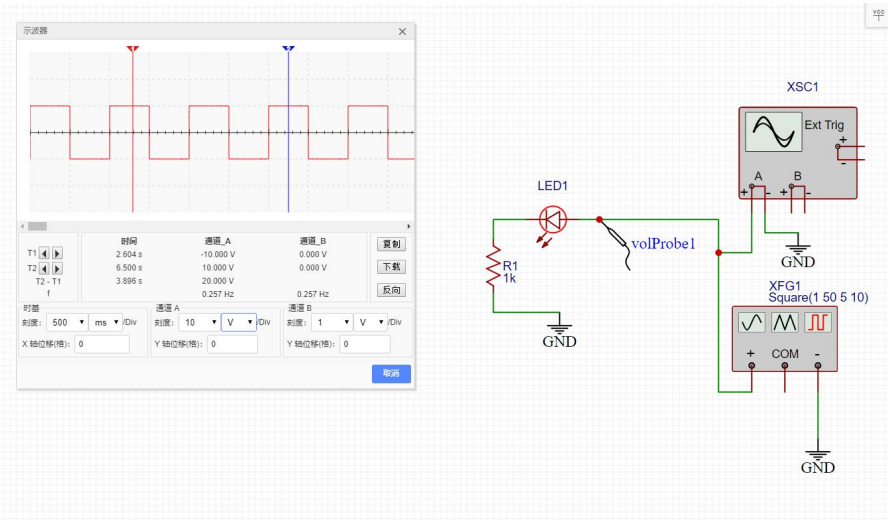
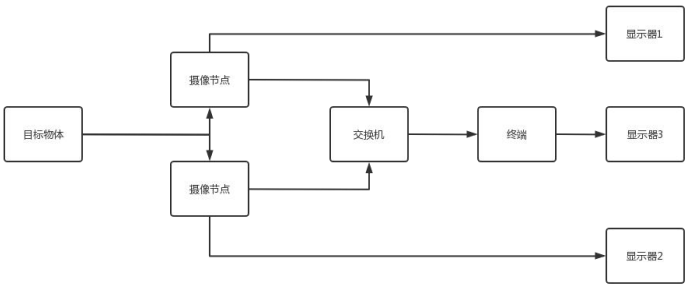
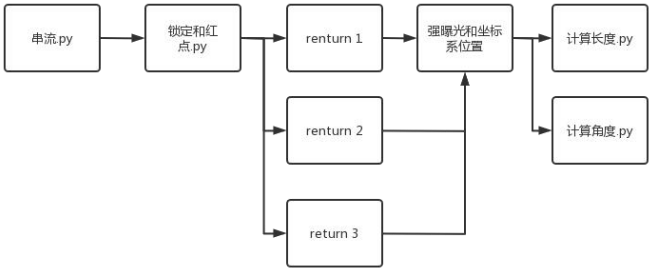


图 2：核心电路设计仿真图

3、 系统软件设计分析





• 系统总体工作流程

图 3：硬件设计流程图

图 4：软件运行流程图

• 主要模块程序设计

详见附件

• 关键模块程序清单

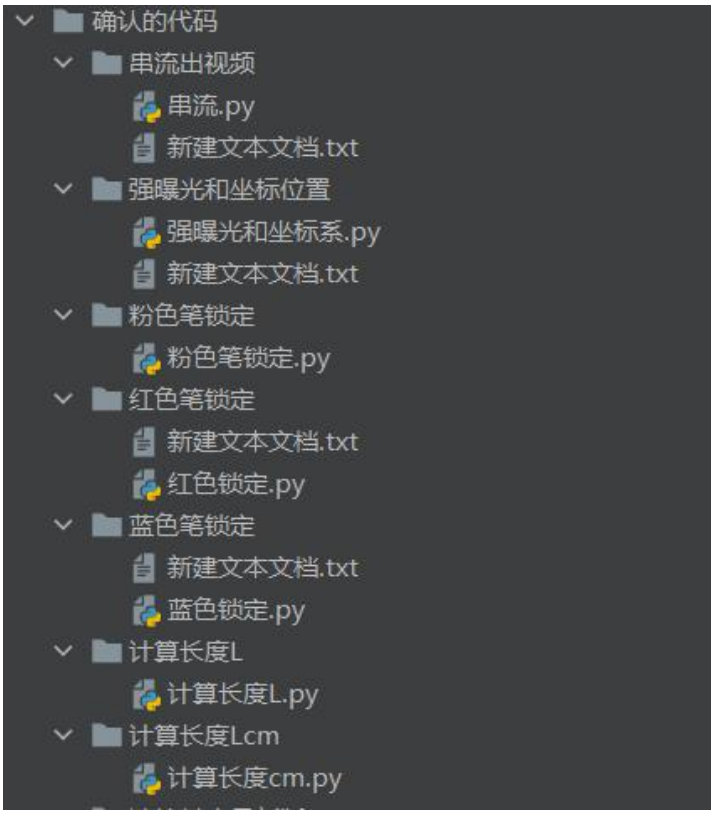


图 5：程序清单

图 6：程序清单

Y	实际长度	L	结果	计算公式：L=(Y_min[0]-479)*21/92 + 142.3087
102.134	56	53.28493913	符合	
130	61	59.64565652	符合	
153.32	66	64.9687	符合	
162.65	67	67.09837391	符合	
166.52	68.5	67.98174348	符合	
175.24	70	69.97217826	符合	
179.68	71	70.98565652	符合	
230.62	82.5	82.61326522	符合	
249.85	87	87.00272174	符合	
268.36	91.1	91.22783043	符合	
289.56	94	96.06696087	符合	
352.47	111.3	110.4268522	符合	
396.25	121.3	120.420113	符合	
365.26	113.5	113.3463087	符合	

图 7：测试状况

4、 竞赛工作环境条件

• 设计分析软件环境

Pycharm:

PyCharm 是一种 Python IDE，带有一整套可以帮助用户在使用 Python 语言开发时提高其效率的工具，比如调试、语法高亮、Project 管理、代码跳转、智能提

示、自动完成、单元测试、版本控制。此外，该 IDE 提供了一些高级功能，以用于支持 Django 框架下的专业 Web 开发。其提供了一个带编码补全，代码片段，支持代码折叠和分割窗口的智能、可配置的编辑器，可帮助用户更快更轻松的完成编码工作。

• 仪器设备硬件平台

树莓派 4b 开发板:

它是一款基于 ARM 的微型电脑主板，以 SD/MicroSD 卡为内存硬盘，卡片主板周围有 1/2/4 个 USB 接口和一个 10/100 以太网接口（A 型没有网口），可连接键盘、鼠标和网线，同时拥有视频模拟信号的电视输出接口和 HDMI 高清视频输出接口，以上部件全部整合在一张仅比信用卡稍大的主板上，具备所有 PC 的基本功能。

图 7：树莓派 4b 图片

以太网交换机（D-Link DES-1024R）： 应用层级：二层；传输速率：10/100Mbps；端口数量：24 个；背板带宽：4.8Gbps；VLAN：不支持；包转发率：10Mbps:14880pps 100Mbps:148800pps；MAC 地址表：8K；网络标准：IEEE 802.3，IEEE 802.3u，IEEE ...；交换方式：存储-转发；端口描述：24 个 10/100Mbps 端口

• 配套加工安装条件

（1） 热熔胶

（2） 3D 打印机

（3） 焊接设备

• 前期设计使用模块

（1）铝合金钢架

（2）物体识别检测算法

5、 作品成效总结分析

• 系统测试性能指标

就基本要求而言，要求测试时 L 的误差在 2cm 内，我们经过多次的验证调试，系统的精确度已经达到了最高，完全符合要求。

针对拓展要求的θ角度计算，为避免激光笔重心不稳带来的旋转，偏移等现象，我们根据形状，设计了相关的钩子，并增加一定质量，减少室内风流的影响。

透明细线具有一定弹性，在摇摆激光笔的过程中，由于惯性，长度 L 将达到最大，为此我们特意选择了弹性最弱的鱼线，在一定程度上减小误差。

• 成效得失对比分析

此次参赛作品的功能符合比赛要求，并全部实现，总体而言是成功的。但不足之处也很明显，缺乏一部分细节功能，时间有限，部分设计也没有实现个性化，所以，赛后我们会继续完善作品，达到最佳。

• 创新特色总结展望

四天三夜的电子设计竞赛，对小组的要求很高，只有配合默契才可以提高效率，节约时间；只有及时沟通，才可以了解到各自任务间的差异，磨合好每个模块。我们小组彼此信任，彼此依靠支持，遇到难以解决的困难也会集思广益帮忙解决，当团结一致的时候，世上便再也没有过不去的坎。同样，提前做好准备也是必要的。我们小组在赛前，充分掌握了相关的专业知识，软硬件分开，却又彼此联系，合理的分工，大大减少了知识盲点，使得工作有条不紊的展开。四天三夜的比赛，我们学到了很多，也对日后参加各类学科竞赛奠定了扎实的信心与基础。

6、 参考资料及文献

- [1]（美）塞利斯基著.计算机视觉 算法与应用[J].工业技术,2020(01).
- [2]（英）西蒙 J.D.普林斯著；苗启广，刘凯，孔韦韦等译. 计算机科学丛书 计算机视觉 模型、学习和推理. 北京：机械工业出版社, 2017.06.
- [3]陈胜勇，刘盛等编著. 基于 OpenCV 的计算机视觉技术实现. 北京：科学出版社, 2008.05.
- [4]（英）易卜拉欣（DoganIbrahim）著. 树莓派高级编程. 南京：东南大学出版社, 2015.09.
- [5]Sibeesh Venu.Asp.Net Core and Azure with Raspberry Pi 4.2020

7、 附件材料

• 重要程序清单

（1）θ角度计算：利用题目基本要求所求出的 L 长度和空间坐标系，标识出激光笔的位置，依赖于坐标系，求出结果。

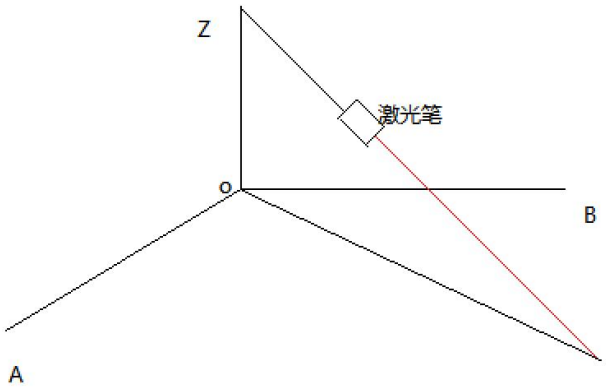


图 8：3 个节点位置安排

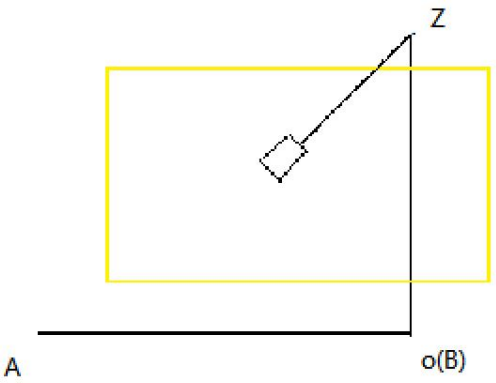


图 9：摄像节点 B，黄框为 B 的观测视角

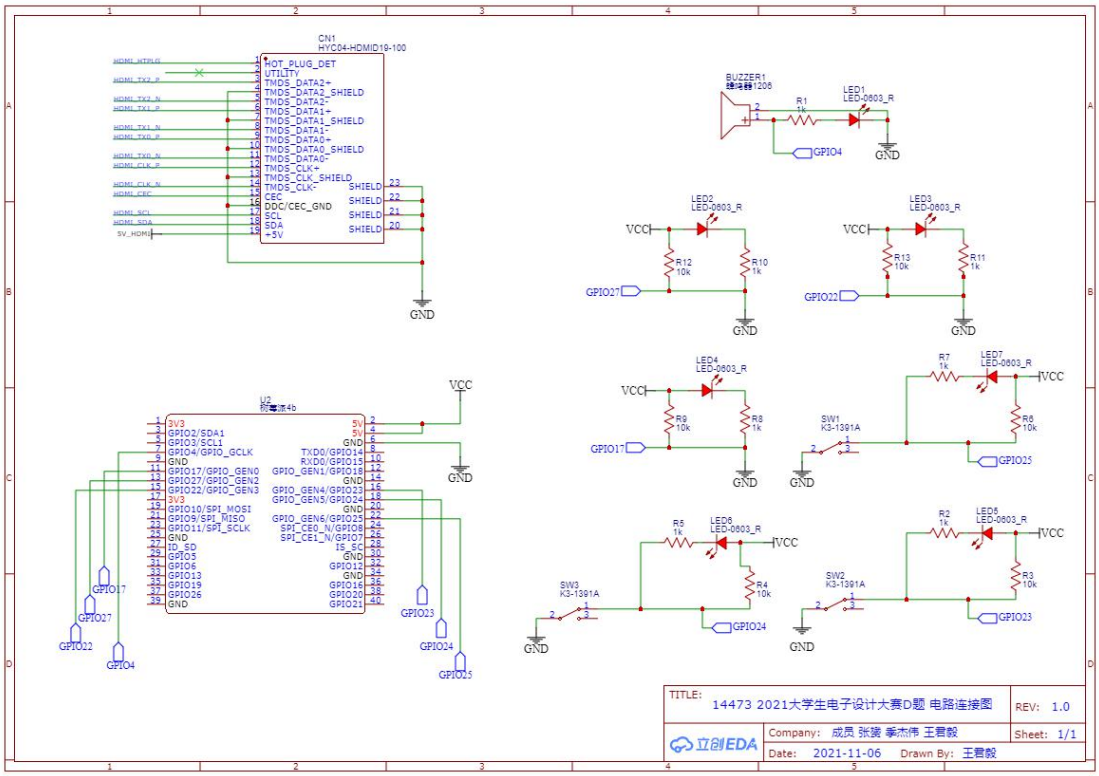


图 10：电路原理图

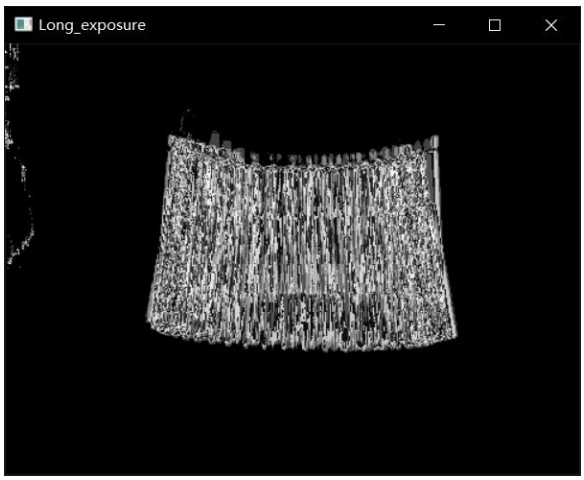


图 11：最终的长曝光截图

图 12：格式化处理的视频截图

图 13：未经处理的原视频

（2）机器学习框架：为保证系统识别的准确性，我们针对激光笔做了机器强化学习，采用 4 个池化层，2 个展平层，大大增加了置信度，降低数据的耦合性。

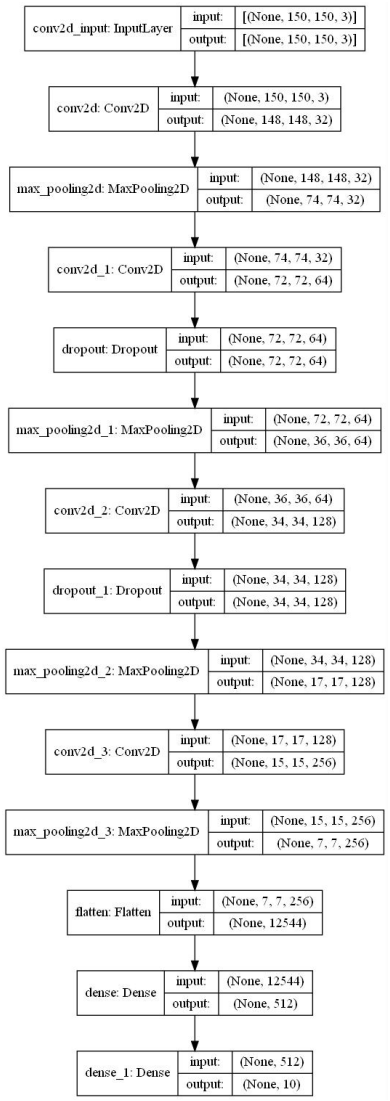


图 14：神经网络结构图

（3）数据集收集，针对目标物体，使用 labelimg 制作出相关的训练集，测试集，验证集

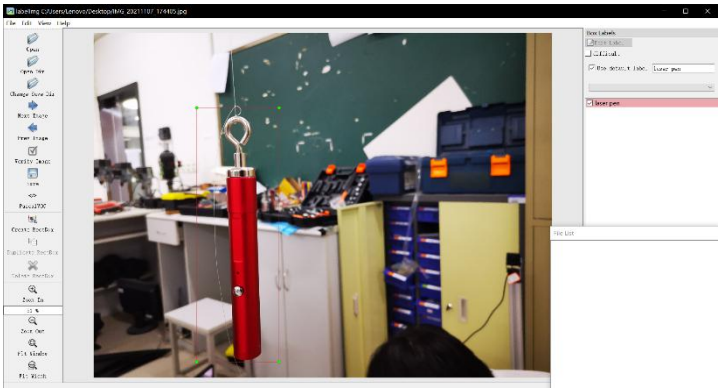


图 15：labelimg 制作训练集

（4）物体锁定：此代码用于识别扫描出激光笔，并标识，实现跟踪的功能。

```
import cv2
import numpy as np

def nothing(value):
    pass
hsv_l = np.array([133,156,59])
hsv_u = np.array([255,255,255])
# def setup_trackbars():
#     cv2.namedWindow("Trackbars",0)
#
#     for i in ["MIN","MAX"]:
#         v =0 if i =="MIN" else 255
#
#         for j in 'HSV':
#             cv2.createTrackbar("%s_%s" % (j,i),"Trackbars",v,255,nothing)
#
# def get_trackbar_values():
#     values =[]
#
#     for i in ["MIN","MAX"]:
#         for j in 'HSV':
#             v =cv2.getTrackbarPos("%s_%s" % (j,i),"Trackbars")
#             values.append(v)
#     return values

cap =cv2.VideoCapture(1)
# setup_trackbars()
while True:
    ret,frame =cap.read()
    if not ret:
        break
    frame =cv2.resize(frame,(480,360))
    img_hsv  =cv2.cvtColor(frame , cv2.COLOR_BGR2HSV)
    # hl,sl,vl,hu,su,vu =get_trackbar_values()
    mask = cv2.inRange(img_hsv,hsv_l,hsv_u)
    #先复制一份
    mask_morph =mask.copy()
    # 函数的第一个参数表示内核的形状
    # 矩形：MORPH_RECT;
    # 交叉形：MORPH_CROSS;
    # 椭圆形：MORPH_ELLIPSE;
```

```
    # ,内核的尺寸以及锚点的位置
    kernel =cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(5,5))
    # 开运算
    mask_morph =cv2.morphologyEx(mask_morph,cv2.MORPH_OPEN,kernel)
    # 闭运算
    mask_morph =cv2.morphologyEx(mask_morph,cv2.MORPH_CLOSE,kernel)
    output =cv2.bitwise_and(frame,frame,mask=mask_morph)


    # 查找图像的轮廓，返回图像的所有轮廓，从而找到所有大的联通区域，-2是取方法返回中的第二个参数
    cnts=cv2.findContours(mask_morph,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)[-2]
    if cnts:
        #  取出最大的解锁边缘，解锁条件key是面积
        c =max(cnts,key=cv2.contourArea)
        # 根据最大的轮廓来读取外包圆
        ((x,y),raduis) = cv2.minEnclosingCircle(c)
        # 计算轮廓的矩
        M = cv2.moments(c)
        # 计算轮廓的重心
        center = (int(M["m10"] / M["m00"]), int(M["m01"] / M["m00"]))
        # # 计算坐标
        # cx = int(M['m10'] / M['m00'])  # 求x坐标
        # cy = int(M['m01'] / M['m00'])  # 求y坐标
        # 矩形
        w, h =  25, 25
        # 只处理尺寸足够大的轮廓
        if raduis > 5:
            # # 画出最小外接圆
            # cv2.circle(frame, (int(x), int(y)), int(raduis), (0, 255, 255), 2)
            # 矩形
            cv2.rectangle(frame, (int(x)-int(raduis), int(y)-int(raduis),int(2*raduis),int(2*raduis)),
color=(0,0, 255), thickness=1)  # BGR
            # 画出重心
            # cv2.circle(frame, center, 5, (0, 0, 255), -1)

        # 如果满足条件，就画出圆，画图函数，frame，中心，半径，颜色，厚度
        # if raduis>10:
        #     cv2.circle(frame,(int(x),int(y)),int(raduis),(0,255,0),6)
    # print("重心坐标是", cx, ",", cy, ")")
    cv2.imshow("original",frame)


    # cv2.imshow("mask",mask)
    # cv2.imshow("mask_morph",mask_morph)
```

```
    # cv2.imshow("output",output)

    if cv2.waitKey(300) & 0xFF is ord("q"):
        break


cap.release()
cv2.destroyAllWindows()
```

（5）测定 L：使用曝光图像，确定出激光笔中心点的位置，依赖于空间坐标系，根据相似三角形等几何性质，求出 L 的长度。

```
import cv2
import numpy as np


def nothing(value):
    pass


hsv_l = np.array([110, 45, 15])
hsv_u = np.array([255, 255, 255])
# def setup_trackbars():
#     cv2.namedWindow("Trackbars",0)
#
#     for i in ["MIN","MAX"]:
#         v =0 if i =="MIN" else 255
#
#         for j in 'HSV':
#             cv2.createTrackbar("%s_%s" % (j,i),"Trackbars",v,255,nothing)
#
# def get_trackbar_values():
#     values =[]
#
#     for i in ["MIN","MAX"]:
#         for j in 'HSV':
#             v =cv2.getTrackbarPos("%s_%s" % (j,i),"Trackbars")
#             values.append(v)
#     return values

cap = cv2.VideoCapture(0)
if( cap.isOpened() ):
 print("cap.isOpened() ")
```

```
# setup_trackbars()
Y_min=[]
for i in range(0,120):
    ret, frame = cap.read()
    if not ret:
        break
        print("2")
    frame = cv2.resize(frame, (480, 360))
    img_hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    # h1,s1,v1,hu,su,vu =get_trackbar_values()
    mask = cv2.inRange(img_hsv, hsv_l, hsv_u)
    # 先复制一份
    mask_morph = mask.copy()
    # 函数的第一个参数表示内核的形状
    # 矩形：MORPH_RECT;
    # 交叉形：MORPH_CROSS;
    # 椭圆形：MORPH_ELLIPSE;
    # ,内核的尺寸以及锚点的位置
    kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (5, 5))
    # 开运算
    mask_morph = cv2.morphologyEx(mask_morph, cv2.MORPH_OPEN, kernel)
    # 闭运算
    mask_morph = cv2.morphologyEx(mask_morph, cv2.MORPH_CLOSE, kernel)
    output = cv2.bitwise_and(frame, frame, mask=mask_morph)

    # 查找图像的轮廓，返回图像的所有轮廓，从而找到所有大的联通区域，-2 是取方法返回中的第二个参数
    cnts = cv2.findContours(mask_morph, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)[-2]
    if cnts:
        #  取出最大的解锁边缘，解锁条件 key 是面积
        c = max(cnts, key=cv2.contourArea)
        # 根据最大的轮廓来读取外包圆
        ((x, y), raduis) = cv2.minEnclosingCircle(c)
        # 计算轮廓的矩
        M = cv2.moments(c)
        # 计算轮廓的重心
        center = (int(M["m10"] / M["m00"]), int(M["m01"] / M["m00"]))
        # 计算坐标
        cx = int(M['m10'] / M['m00'])  # 求 x 坐标
        cy = int(M['m01'] / M['m00'])  # 求 y 坐标
        # 矩形
        w, h = 25, 25
        # 只处理尺寸足够大的轮廓
        if raduis > 5:
            # # 画出最小外接圆
```

```
            # cv2.circle(frame, (int(x), int(y)), int(raduis), (0, 255, 255), 2)
            # 矩形
            cv2.rectangle(frame, (int(x) - int(raduis), int(y) - int(raduis), int(2 * raduis), int(2 * raduis)),
                          color=(0, 0, 255), thickness=1)  # BGR
            # 画出重心
            cv2.circle(frame, center, 5, (0, 0, 255), -1)

        # 如果满足条件，就画出圆，画图函数，frame，中心，半径，颜色，厚度
        # if raduis>10:
        #     cv2.circle(frame,(int(x),int(y)),int(raduis),(0,255,0),6)
    cv2.imshow("original", frame)

    # print("重心坐标是(", cx, ",", cy, ")")
    Y_min.append(cy)
    # cv2.imshow("mask",mask)
    # cv2.imshow("mask_morph",mask_morph)
    # cv2.imshow("output",output)

    if cv2.waitKey(300) & 0xFF is ord("q"):
        break
Y_min.sort(reverse=True)
print(Y_min[0])
cap.release()
cv2.destroyAllWindows()
```

（6）长曝光处理：使用 opencv 的相关函数，对实时的视频进行灰度等格式化处理，再调整 RGB 色彩得到完整的清晰的曝光图像。

```
# coding:utf-8
# ************************************************************************************************************
'''
说明：利用 python/opencv 实现摄影中的图像长曝光
算法思路：
        1)读取视频图像，将每一帧图像进行通道分离；
        2)采用简单的平均思想，对序列帧进行处理；
      3)合并处理后分离的三通道图像；
'''

import cv2
import numpy as np
import argparse
import os
from PIL import Image
import matplotlib.pyplot as plt
```

```
if __name__ == '__main__':
    print('[INFO] 读取视频流......')
    cap = cv2.VideoCapture("output.mp4")
    num = 0
    (ravg, gavg, bavg) = (None, None, None)
    while True:
        while (num < 300):  # 只处理本地视频的前 300 帧，因为后续视频与当前场景不同
            # 你可以根据自己的视频选择
            ret, frame = cap.read()
            print(num)
            if not ret:
                print('    视频流读取失败或读取完成    ')
                break
            # 图片通道分离
            cv2.imshow('frame', frame)
            cv2.waitKey(20)
            (B, G, R) = cv2.split(frame)
            if ravg is None:
                ravg, gavg, bavg = R, G, B
            else:
                ravg = (R + num * ravg) / (num + 0.01)
                gavg = (G + num * gavg) / (num + 0.01)
                bavg = (B + num * bavg) / (num + 0.01)
            num += 0.01
        # 通道合并为长曝光图像
        long_exposure_img = cv2.merge([bavg, gavg, ravg]).astype('uint8')
        cv2.imshow('Long_exposure', long_exposure_img)
        cv2.imwrite("1.jpg", long_exposure_img)
        img = Image.open("1.jpg")
        plt.figure("Image")  # 图像窗口名称
        plt.imshow(img)
        plt.axis('on')  # 关掉坐标轴为 off
        plt.title('image')  # 图像题目

        plt.show()

        cv2.waitKey(0)
```

（7）框识和锁定跟踪：此代码应用于摄像节点的树莓派上，利用 90°摄像头，完全实现实时跟踪处理。

```
import cv2
import numpy as np


def nothing(value):
    pass


hsv_l = np.array([110, 45, 15])
hsv_u = np.array([255, 255, 255])
# def setup_trackbars():
#     cv2.namedWindow("Trackbars",0)
#
#     for i in ["MIN","MAX"]:
#         v =0 if i =="MIN" else 255
#
#         for j in 'HSV':
#             cv2.createTrackbar("%s_%s" % (j,i),"Trackbars",v,255,nothing)
#
# def get_trackbar_values():
#     values =[]
#
#     for i in ["MIN","MAX"]:
#         for j in 'HSV':
#             v =cv2.getTrackbarPos("%s_%s" % (j,i),"Trackbars")
#             values.append(v)
#     return values

cap = cv2.VideoCapture(0)
# setup_trackbars()
Y_min=[]
for i in range(0,120):
    ret, frame = cap.read()
    if not ret:
        break
    frame = cv2.resize(frame, (480, 360))
    img_hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    # h1,s1,v1,hu,su,vu =get_trackbar_values()
    mask = cv2.inRange(img_hsv, hsv_l, hsv_u)
    # 先复制一份
    mask_morph = mask.copy()
    # 函数的第一个参数表示内核的形状
    # 矩形：MORPH_RECT;
    # 交叉形：MORPH_CROSS;
```

```
# 椭圆形：MORPH_ELLIPSE;
# ,内核的尺寸以及锚点的位置
kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (5, 5))
# 开运算
mask_morph = cv2.morphologyEx(mask_morph, cv2.MORPH_OPEN, kernel)
# 闭运算
mask_morph = cv2.morphologyEx(mask_morph, cv2.MORPH_CLOSE, kernel)
output = cv2.bitwise_and(frame, frame, mask=mask_morph)

# 查找图像的轮廓，返回图像的所有轮廓，从而找到所有大的联通区域，-2 是取方法返回中的第二个参数
cnts = cv2.findContours(mask_morph, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)[-2]
if cnts:
    #  取出最大的解锁边缘，解锁条件 key 是面积
    c = max(cnts, key=cv2.contourArea)
    # 根据最大的轮廓来读取外包圆
    ((x, y), raduis) = cv2.minEnclosingCircle(c)
    # 计算轮廓的矩
    M = cv2.moments(c)
    # 计算轮廓的重心
    center = (int(M["m10"] / M["m00"]), int(M["m01"] / M["m00"]))
    # 计算坐标
    cx = int(M['m10'] / M['m00'])  # 求 x 坐标
    cy = int(M['m01'] / M['m00'])  # 求 y 坐标
    # 矩形
    w, h = 25, 25
    # 只处理尺寸足够大的轮廓
    if raduis > 5:
        # # 画出最小外接圆
        # cv2.circle(frame, (int(x), int(y)), int(raduis), (0, 255, 255), 2)
        # 矩形
        cv2.rectangle(frame, (int(x) - int(raduis), int(y) - int(raduis), int(2 * raduis), int(2 * raduis)),
                      color=(0, 0, 255), thickness=1)  # BGR
        # 画出重心
        cv2.circle(frame, center, 5, (0, 0, 255), -1)

    # 如果满足条件，就画出圆，画图函数，frame，中心，半径，颜色，厚度
    # if raduis>10:
    #     cv2.circle(frame,(int(x),int(y)),int(raduis),(0,255,0),6)
cv2.imshow("original", frame)

# print("重心坐标是(", cx, ",", cy, ")")
Y_min.append(cy)
# cv2.imshow("mask",mask)
# cv2.imshow("mask_morph",mask_morph)
```

```
        # cv2.imshow("output",output)

        if cv2.waitKey(300) & 0xFF is ord("q"):
            break
Y_min.sort(reverse=True)
print(Y_min[0])
cap.release()
cv2.destroyAllWindows()
```

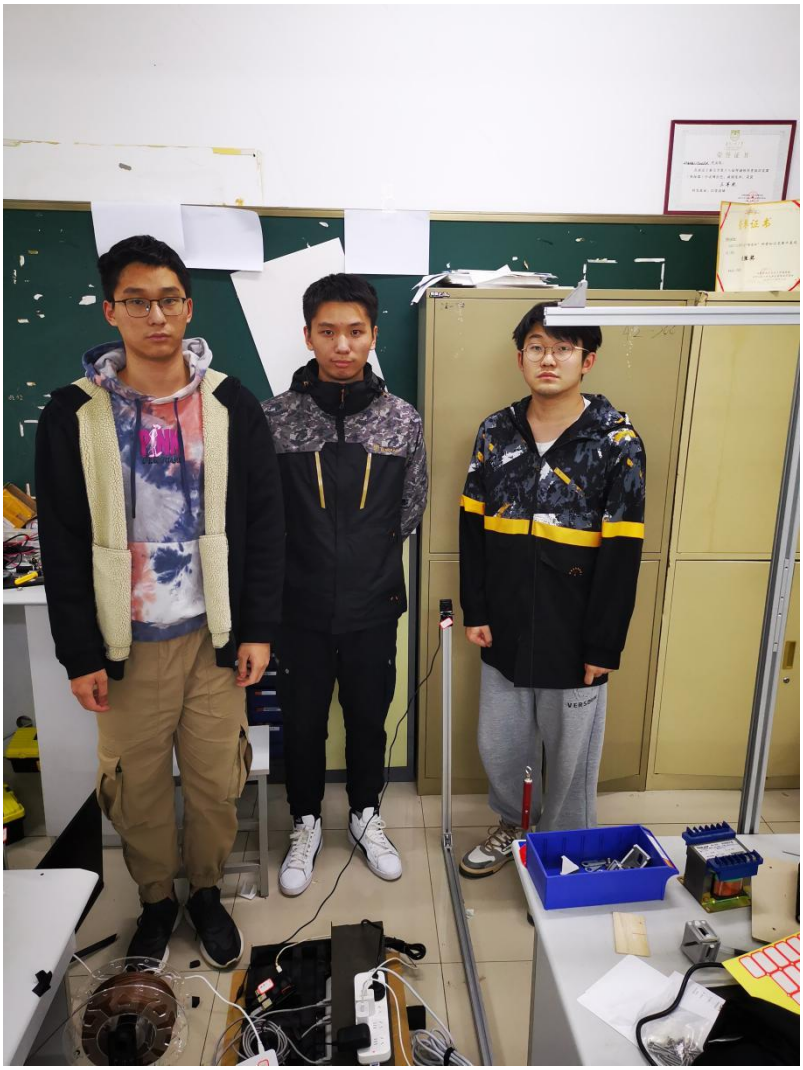


图 16：三人合照

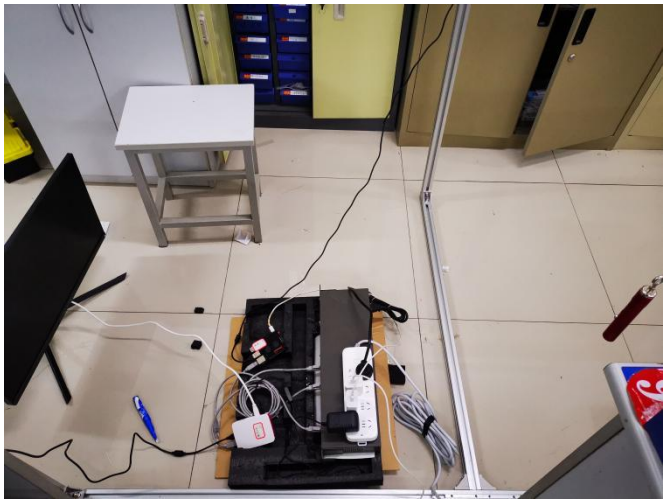


图 17：作品图像