基于互联网的摄像测量系统报告

摘 要：本设计实现了一种基于互联网的摄像测量网络，通过两个正交拍摄的摄像节点，完成对激光笔悬线长度和轨迹投影夹角的测量，最终测量系统的各项性能均满足题目要求。在前景提取方面，本设计采用了基于混合高斯模型的图像差分算法，通过参数的合理选取和逻辑判断，实现了激光笔目标的稳定检出；采用 socket TCP 协议实现了视频从摄像节点到终端的传输。在方案论证中从分散各节点运算压力和实现难易程度出发，将前景提取和测量值计算算法分别部署在不同的节点上，使系统的整体性能达到最优。整体方案的各项指标均达到题目要求，且两测量量的可测量范围较题目要求都有提高。

关键词：互联网；计算机视觉；目标检测；测量系统

目录

1.设计方案工作原理

1.1 预期实现目标定位

本题要求制作一个基于互联网的摄像测量系统对激光笔运动参数进行测量，系统中摄像头节点和终端节点通过网络交换机实现网络互连，终端节点通过来自摄像头节点的回传数据实现对悬线长度l与轨迹投影夹角θ的测量。摄像测量系统如图 1 所示。

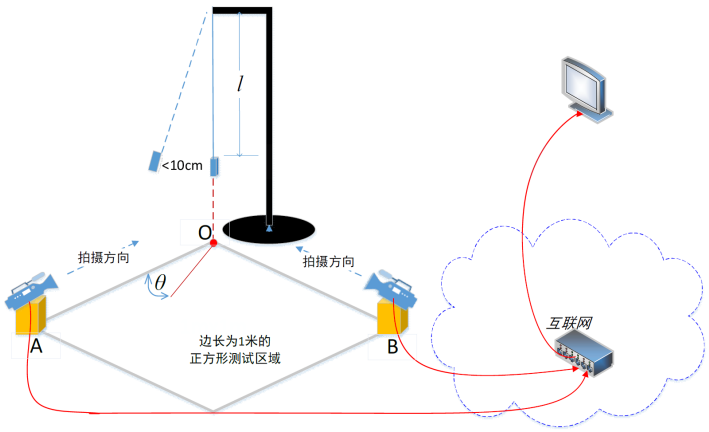

图 1 摄像测量系统示意图

1.2 技术方案分析比较

（1）摄像节点设计

方案 1：使用 OpenMV 嵌入式图像处理模块。摄像头模块集成嵌入式系统，可以直接在数据采集端进行数据处理，但模块没有板载网络接口，实现网络开发较为困难。

方案 2：USB 摄像头与树莓派组合。树莓派读取摄像头视频帧后运行前景提取算法，将处理好的视频帧通过网络传输至终端，与终端节点在同一平台开发，网络传输实现方便。

综合以上两种方案，选择方案 2。

（2）激光笔目标检测

方案 1：基于颜色明度的目标检测。将激光笔特定位置用红，绿或蓝色标记，通过检测 RGB 通道中像素值最高的像素获得激光笔位置。优点是近距离处检测精确，但目标较小时准确度下降，且受光线，环境影响较大。

方案 2：基于浅层神经网络的目标检测。使用标记数据集对网络模型进行训练，可以实现对激光笔较为准确的检测，但数据集标定需要耗时，且运算量较大。

方案 3：基于混合高斯模型的运动前景提取。混合高斯模型基于对每个像素点的亮度分布进行高斯建模以实现前景和背景的分离，对小物体有较高的检测灵敏度，但计算资源占用稍大，容易检出其他前景。

综合以上两种方案和实地试验，选择方案 3。

（3）视频流网络传输

方案 1：基于 ZeroMQ 的 TCP 协议传输。ZeroMQ 传输的是消息而不是字节流，在开发上更加方便，但其进行通信时使用的时消息队列，两个摄像节点必须同时开始数据发送，对线程间的同步要求较高。

方案 2：基于 socket 的 TCP 协议传输。传输建立在服务端和客户端的可靠连接上，实时性较好，但由于传输的是字节流，错位将会导致随后的数据接收出错，进而引发程序异常停止。

综合以上两种方案，本题对传输的实时性要求较高，因此选择方案 2。

1.3 系统结构工作原理

（1）激光笔目标检测和关键点提取原理

激光笔目标检测是通过高斯混合模型（GMM）背景建模与背景减除法实现的。高斯混合模型背景建模的思想是用 K 个高斯分布的叠加来描述像素点p处的像素值分布，也就是

$$M_p = \{G_1(\mu_1, \sigma_1^2, \omega_1), \dots, G_i(\mu_i, \sigma_i^2, \omega_i), \dots, G_k(\mu_k, \sigma_k^2, \omega_k\} \quad (1)$$

其中$\mu_i, \sigma_i^2, \omega_i$分别是第$i$个高斯分布的均值，方差和权重。新一视频帧到来时，将其中每个像素点与混合高斯模型进行匹配，如果成功则判定该点为背景点，否则为前景点。高斯混合模型的性能主要由方差和均值两个参数决定，不同参数的选取将直接影响到模型的稳定性，精确性和收敛性。

提取出激光笔前景后，使用面积最小的矩形框选出激光笔笔身，得到矩形的左上角顶点坐标和矩形宽高x, y, w, h，考虑到激光笔笔身呈细长型，故选取矩形对角线中心作为笔身中心特征点用以进行相关计算，即将$(x + w/2, y + h/2)$取为特征点。

（2）socket 视频流网络传输原理

socket 为复杂的 TCP/IP 协议提供了一组函数调用接口，将数据发送到网络端进行传输，基于 socket 的 TCP 协议通信流程如图 2 所示。在服务端创建 socket 套接字后，绑定不同的 IP 和端口号并监听，随后程序阻塞等待建立连接。成功建立连接后，才会开始数据的发送/接收，连接一旦断开，套接字便会关闭，数据停止发送/接收。数据的发送/接收建立在可靠连接之上，没有连接时不会发送任何数据，从而保证客户端和服务端即使不同时运行程序，也能通过阻塞的形式，使得数据同步，实现本系统终端节点实时显示摄像节点视频流的功能。

为传输视频流，将每一帧图片编码为 JPEG 格式的字节流以减少数据在网络层的传输时间，并在接收端解码。图片压缩编码过程中，不同的图片产生的字节流数据量并不相等，为防止出现数据读取错位，在传输每一帧的数据之前需要将图片的字节流长度传输给接收端，接收端根据这个长度接收并解码图片，多帧连续地收发便可以实现视频流的传输。

（3）悬线长度l与轨迹投影夹角θ测量原理

①悬线长度测量

据牛顿第二定律，终端系有重物的长度为l的轻绳在竖直方向上做周期摆动时满足如下运动学方程

$$\frac{d^2\varphi}{dt^2}+\frac{g}{l}sin\,\varphi=0 \tag{2}$$

当此单摆摆幅φ较小时，有近似关系$\sin\varphi\approx\varphi$，此时（2）式就变为

$$\frac{d^2\varphi}{dt^2}+\frac{g}{l}\varphi=0 \tag{3}$$

这是一个二阶常系数线性微分方程，其通解为$\varphi=A\cos(\omega t+\gamma)$，式中$A,\gamma$为由初始条件确定的常数，而

$$\omega^2=\frac{g}{l} \tag{4}$$

于是单摆的运动被线性近似为简谐运动，其周期

$$T=\frac{2\pi}{\omega}=2\pi\sqrt{\frac{l}{g}} \tag{5}$$

从而得到

$$l=\frac{gT^2}{4\pi^2} \tag{6}$$

由此，在已知当地重力加速度g的条件下，通过对视频中激光笔摆动周期的测量，可以间接测量摆线长度。

②轨迹投影夹角θ测量

根据图 1 的几何关系，激光笔轨迹的在地面上长度为l垂直投影在 OA，OB 端的投影长度l_{OA},l_{OB}分别为

$$l_{OA}=l\,cos\,\theta \tag{7}$$

$$l_{OB}=l\,sin\,\theta \tag{8}$$

从而解得θ

$$\theta = arctan\frac{l_{OB}}{l_{OA}} \tag{9}$$

考虑到两摄像头的拍摄条件相同，投影与轨迹在像素中的长度成正比，因此可以将轨迹投影的像素长度代入式（9）求得夹角θ。

1.4 功能指标实现方法

（1）拍摄和显示视频功能

使用 USB 摄像头与树莓派构成摄像节点，每个摄像节点配置屏幕即可独立显示。

（2）终端显示视频功能

使用 socket 协议，将视频帧转换为文本流即可通过交换机传输至终端显示。

（3）激光笔识别功能

使用高斯混合模型做前景提取即可实现。

（4）线长与角度测量功能

使用单摆周期近似公式与简单几何建模即可实现。

1.5 测量控制分析处理

（1）测试工具

①卷尺；②量角器；③秒表。

（2）测试方法

按照题目要求，依如下功能测试：

①拍摄和显示视频功能测试；②终端显示视频功能测试；③激光笔识别功能测试；④线长和轨迹投影测量功能测试。

2.核心部件设计

2.1 关键器件性能分析

（1）树莓派 4B：板载 ARM Cortex-A53 1.5G 四核处理器，千兆以太网接口，USB 2.0 和 3.0 接口各 2 个，2 个 micro HDMI 接口。能够实现外接 USB 摄像头和与网络交换机的数据传输，算力足以满足视频帧的处理。

（2）英伟达 Jetson Nano：板载 128 核心 Maxwell 架构的 GPU 和 4GB 64 位 LPDDR4 内存，千兆以太网接口，可以提供 0.5TFlop 的算力，能够满足终端节点对计算能力的需求。

（2）USB 摄像头：5V 供电，最大分辨率 1080P。

（3）TP-LINK 千兆无线网络交换机：机载 1 个千兆 WAN 口和 3 个千兆 LAN 口，能够满足摄像节点和终端节点的有线接入。

2.2 电路结构工作机理

（1）系统供电和网络连接

系统中使用的树莓派均采用 USB Type-C 接口供电，电压 5V，使用手机充电器实现供电；网络交换器与显示屏均采用配套电源供电，统一接于同一排插上。各节点使用千兆网线与交换机相连。

（2）声光指示与按键电路模块

系统中用于启动测量的外部按键电路和声光模块电路如图 2 所示。

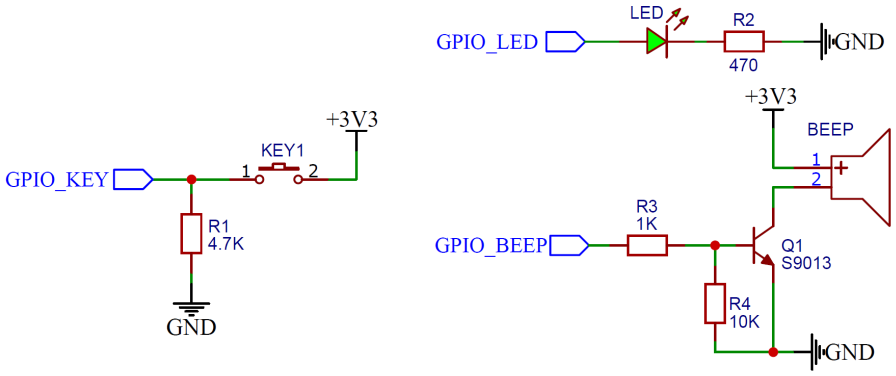

图 2 按键与声光电路模块

连接按键的 GPIO 经过一下拉电阻接地，按键未按下时呈低电平，按键按下时在树莓派 GPIO 产生一上升沿电平，触发测量开始。声光模块由直插封装 LED 与有源蜂鸣器构成，有源蜂鸣器通过三极管 9018 与 GPIO 相连，提高树莓派的带载能力。所有电路均工作在 3.3V 电平下。

2.3 系统实现调试测试

我们使用 Photoshop 绘制并印刷了符合题目要求的$1.2m \times 1.2m$场地贴图（见内容补充），并在实验室借助三脚架等器材搭建了简易测试平台，在该平台上对系统算法的性能进行测试和迭代。

3. 系统软件设计分析

3.1 系统总体工作流程

图 3-1 和图 3-2 分别为摄像节点和终端节点的软件工作流程。

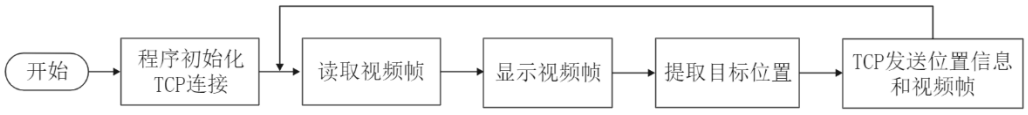

图 3-1 摄像节点软件工作流程

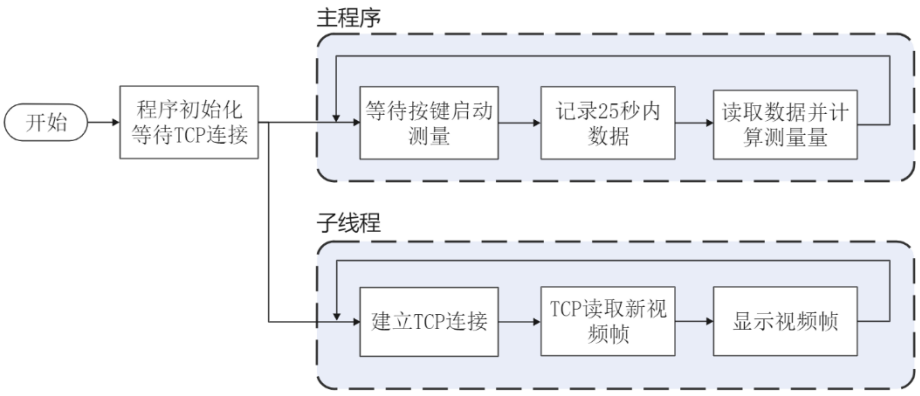

图 3-2 终端节点软件工作流程

3.2 主要模块程序设计

（1）激光笔目标检测模块

此模块封装了用于检测激光笔目标的类 Detector。将 Detector 实例化为对象时，程序将创建混合高斯背景模型并初始化，随后的每一视频帧将与背景进行差分得到前景图像。考虑到可能出现细小前景误检的情况，程序中对前景大小进行了限制，另外，由于激光笔在视频中基本呈竖直状态，采用大小为(7,7)的核对前景进行图像形态学上的膨胀处理，并将框选前景的小矩形中心作为目标点坐标返回。

（2）socket TCP 数据收发模块

由于 socket TCP 协议传输的数据为字节流，因此需要先将图像编码为字节数据才能使用 TCP 协议进行收发。为避免占用过大数据带宽，程序将 USB 摄像头读取的 1080p 视频重置为640×480大小的图像并压缩为 jpg 编码。摄像节点将目标点坐标和编码为字节的图像数据循环交替发送，终端节点循环交替接收，从而实现视频流和目标坐标的传输。对于两个摄像节点，使用不同的端口号发送以区分数据。

（3）悬线长与投影夹角测量模块

此模块内主要完成两个步骤：异常点排除和测量量计算。针对目标检测模块可能返回的位置异常的坐标值，将某坐标与其相邻两坐标距离是否小于某一阈值，作为该点是否为异常点的判据，若是异常点则将其排除。

剩下的正常数据点中，由式（9）知，将正交方向的数据在时间上对齐后，投影角度可以对两个正交方向单摆摆幅的比值取反正切函数得到，此方法可将测量范围由$0\sim90°$拓展到$-90°\sim90°$。同时对数据进行线性拟合，以提高检测角度的精度。

悬线长度的测量方法由式（6）确定，其中单摆摆动周期是采用逐差法计算得出的。针对投影角度接近0°或90°时的情形，采用非线性加权的方式提高置信度高的通道在测量中的影响因子。

3.3 关键模块程序清单

（1）客户端

①Detector.py：封装激光笔目标检测模块，用于提取激光笔位置信息

②XXX_TCP.py（XXX 是客户端 ip 后三位）：客户端 socket TCP 发送程序

（2）服务端

①Calculator.py：封装悬线长，投影夹角测量函数

②XXX_TCP.py（XXX 是服务端 ip 后三位）：服务端 socket TCP 接收程序

4.竞赛工作环境条件

4.1 设计分析软件环境

①Raspbian OS；②Python 3；③OpenCV 4；④Visual Studio Code。

4.2 仪器设备硬件平台

①1080P 分辨率网络摄像头；②树莓派 4B；③TP-LINK 千兆无线网络交换机。

4.3 配套加工安装条件

①金属型材拼装 1.5m 高激光笔支架；②广告公司印刷$1.2 \times 1.2m$测试场地贴图；③摄像头三脚架。

4.4 前期使用设计模块

①OpenMV 嵌入式图像处理模块。

5.作品成效总结分析

5.1 系统测试性能指标

（1）摆线长度l测量准确率：取$l = 50cm, 100cm, 150cm$三个摆长进行测试，在不同投影夹角下测量长度，计算准确率；

（2）投影夹角θ测量准确率：每个摆长取值下，取$\theta = -45°, 0°, 30°, 45°, 60°, 90°$五个特殊投影夹角进行测量，计算准确率；

（3）测量时间：按下测量按键到系统输出测量结果的耗时。

5.2 成效得失对比分析

对前文所述指标进行实际测量，记录得到表 1 所示的结果。

表 1 系统测量结果与误差

摆长实际取值/cm	投影夹角实际取值/°	摆长测量值/cm	摆长测量误差η_l/%	投影夹角测量值/°	投影夹角测量误差$\|\Delta\theta\|$/°	测量时间/s
50	−45°	50.58	1.16	-43.7	1.3	25.034
	0°	50.59	1.18	1.5	1.5	25.031
	30°	50.36	0.72	32.8	2.8	25.031
	45°	50.32	0.64	46.4	1.4	25.029
	60°	50.17	0.34	59.3	0.7	25.030
	90°	50.52	1.04	87.7	2.3	25.041
100	−45°	100.80	0.80	-47.7	2.7	25.033
	0°	101.21	1.21	2.7	2.7	25.052
	30°	100.95	0.95	29.3	0.7	25.026
	45°	100.83	0.83	43.1	1.9	25.027
	60°	100.88	0.88	58.6	1.4	25.032
	90°	101.30	1.30	88.9	1.1	25.026
150	−45°	151.14	0.76	-42.6	2.4	25.032
	0°	149.07	-0.62	1.6	1.6	25.028
	30°	150.16	0.11	29.9	0.1	25.028
	45°	150.40	0.27	45.3	0.3	25.026
	60°	149.68	-0.21	61.0	1.0	25.025
	90°	150.08	0.05	89.1	0.9	25.026

经过多次实验数据测量，本摄像测量系统性能符合题目的要求，本设计全部完成了题目的基本要求和发挥部分的要求，具有较高的稳定性。

5.3 创新特色总结展望

本设计采用混合高斯模型背景建模实现了视频流中运动前景的提取，采用 socket TCP 协议实现了视频从摄像节点到终端的传输，摄像节点和终端节点分别采用了树莓派 4B 和英伟达 Jetson 运算核心以提高系统的数据处理能力，最终测量系统的各项性能均满足题目要求。混合高斯模型算法的难点在于对模型参数的选取，为达到理想的效果，在不同光照，背景下对算法进行了反复试验并增加了逻辑判断模块，最终达到了良好的提取效果。在方案论证中从分散各节点运算压力和实现难易程度出发，将前景提取和测量值计算算法分别部署在不同的节点上，使系统的整体性能达到最优。整体方案的各项指标均达到题目要求，且两测量量的可测量范围较题目要求都有提高。

6. 参考资料及文献

[1] OpenCV 官方文档[DB/OL]. https://docs.opencv.org/

[2] 程守洙, 江之永.普通物理学[M]. 第七版. 北京: 高等教育出版社,2016

[3] Richard Blum, Christine Bresnahan. Linux 命令行与 Shell 脚本大全[M]. 第二版. 武海峰. 北京: 人民邮电出版社, 2012

[4] 胡仁杰, 堵国樑, 黄慧春. 全国大学生电子设计竞赛优秀作品设计报告选编(2015年江苏赛区)[M]. 南京: 东南大学出版社, 2016

7. 内容补充

（1）$1.2m \times 1.2m$测试场地地面贴图

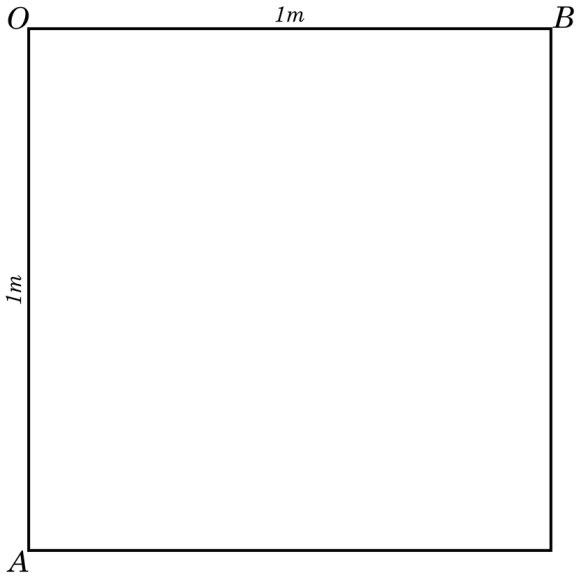

图 3 测试场地地面贴图（$1.2m \times 1.2m$）

8. 附件

（1）激光笔目标检测模块源码 detector.py

```
# encoding: utf-8
import cv2
import numpy as np

class Detector(object):
    def __init__(self):
        self.fgbg = cv2.createBackgroundSubtractorMOG2(history=250, varThreshol
d=100, detectShadows=False)

    def apply(self, frame):
        fgmask = self.fgbg.apply(frame)

        # 图像形态学膨胀操作
```

```
        fgmask = cv2.dilate(fgmask, cv2.getStructuringElement(cv2.MORPH_RECT,
(7, 7)), iterations=3)

        contours, _ = cv2.findContours(fgmask, cv2.RETR_EXTERNAL, cv2.CHAIN_APP
ROX_SIMPLE)

        # ...(省略逻辑判断部分内容)

        return center, frame, fgmask
```

（2）摄像节点 socket TCP 发送源码 XXX_TCP.py

```
# encoding: utf-8

# ...(省略部分初始化代码)
cap = cv2.VideoCapture(0)
detector = Detector()

try:
    sock = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
    # 开启 socket TCP 连接
    sock.connect(address)
except socket.error as msg:
    print(msg)
    exit(1)

# 图像质量
encode_param=[int(cv2.IMWRITE_JPEG_QUALITY), 50]

while True:
    ret, frame = cap.read()
    if not ret:
        break

    # 发送当前时间
    now = time()
    sock.send(str.encode(str(now).ljust(64)))

    center, _, _ = detector.apply(frame)
    cv2.imshow(name, frame)
    key = cv2.waitKey(1)
    if((key & 0xff) == 27):
        break

    # 发送坐标
    if(center is None):
        sock.send(str.encode(str(-1).ljust(16)))
        sock.send(str.encode(str(-1).ljust(16)))
    else:
        sock.send(str.encode(str(center[0, 0]).ljust(16)))
        sock.send(str.encode(str(center[1, 0]).ljust(16)))
```

```
    # 发送图片
    result, imgencode = cv2.imencode('.jpg', frame, encode_param)
    data = np.array(imgencode)
    stringData = data.tostring()
    sock.send(str.encode(str(len(stringData)).ljust(64)))
    sock.send(stringData)

sock.close()
cv2.destroyAllWindows()
```

（3）终端节点 socket TCP 接收源码 XXX_TCP.py

```
    #   ...(省略了初始化部分代码)

# 线程函数
def ReceiveVideo(PORT, name):
    address = ('0.0.0.0', PORT)
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    sock.bind(address)
    sock.listen(1)

    def recvall(sock, count):
        buf = b''
        while count:
            newbuf = sock.recv(count)
            if not newbuf: return None
            buf += newbuf
            count -= len(newbuf)
        return buf

    # 没有连接则等待有连接
    conn, addr = sock.accept()
    print('connect from:'+str(addr))

    while True:
        now = float(recvall(conn, 64))  # 获取当前图片的时间
        x = round(float(recvall(conn, 16)))
        y = round(float(recvall(conn, 16)))
        if((x != -1) and (y != -1) and dataRead):
            if(name[-1] == '2'):
                global List1
                List1.append([now, x, y])
            else:
                global List2
                List2.append([now, x, y])

        # 获得图片文件长度
        length = int(recvall(conn, 64))
        # 根据获得的文件长度，获取图片文件
        stringData = recvall(conn, length)

        data = np.frombuffer(stringData, np.uint8)
        # 解码图像
```

```
        decimg=cv2.imdecode(data,cv2.IMREAD_COLOR)
        cv2.imshow(name, decimg)

        k = cv2.waitKey(1) & 0xff
        if(k == 27):  # ESC
            sock.close()
            cv2.destroyAllWindows()
            GPIO.cleanup()
            os._exit(0)

t1 = threading.Thread(target=ReceiveVideo, args=(8102, "192.168.1.102"))
t2 = threading.Thread(target=ReceiveVideo, args=(8108, "192.168.1.108"))
t1.setDaemon(True)
t2.setDaemon(True)
t1.start()
t2.start()
while(1):
    print("ON")
    GPIO.cleanup()
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(KEY, GPIO.IN)
    GPIO.setup(BEEP, GPIO.OUT, initial=0)
    GPIO.setup(LED, GPIO.OUT, initial=0)
    GPIO.wait_for_edge(KEY, GPIO.RISING)
    print("Press time: " + str(datetime.now())[0:-7])

    List1.clear()
    List2.clear()
    dataRead = True
    startTime = time()
    while(1):
        if(time() - startTime >= maxTime):
            dataRead = False
            break
    l, th = calculate_angle_period(List1, List2)

    print(len(List1), len(List2))
    print("\nlength = %.2fcm" % (float(l) * 100))
    print("theta = %.1f°" % th)

    GPIO.output(LED, 1)
    print("time = %.4fs\n" % (time() - startTime))
    sleep(1)
    GPIO.output(BEEP, 0)
    GPIO.output(LED, 0)
    print("OFF")
t1.join()
t2.join()
```