系统分析

1、可行性分析

（1）、QZXing：Qt包装ZXing的解码库。ZXing是条形码处理类库，它是一个开源Java类库用于解析多种格式的1D/2D条形码。包含了联系到其他语言的端口。

（2） Opencv：OpenCV(开源的计算机视觉库)是BSD许可下发布的,因此它是免费的学术和商业用途。c++,C,Python和Java接口,支持Windows,Linux,Mac OS,iOS和Android。

2、效率分析

（1）、QXZing： 最长20秒内识别到二维码。

（2）、OpenCV：OpenCV是为计算效率而设计的，并且专注于实时应用程序用，该库采用优化的c/c++编写，可以利用多核处理。

3、准确性分析

（1）、QXZing：识别二维码效率达到98%以上。

系统设计

1、HID通讯模块 CHidCmdThread

功能：HID命令排队操作线程，HID命令添加请一律通过该类进行处理

包含类

class ProtocolUtility 通讯协议类

class HIDOpertaionUtility USB操作类

class HIDReadThread USB读线程类

类关系



2、二维码识别模块 QRCodeDetector

功能：从图片中提取二维码并识别、解析二维码数据，发送到测试线程。

包含类

class Hidopertaionutility USB操作类

class Protocolutility 通讯协议类

class Opencvutility Opencv操作类

class QZXing 二维码解码类

class CHidCmdThread HID通讯类

类关系



3、标准机亮度校正模块 StandardBrightness

功能：输入标准机灯光值，获取绿色分量曲线数据

包含类：

class CHidCmdThread HID通讯类

class Hidopertaionutility USB操作类

class Protocolutility 通讯协议类

class Opencvutility Opencv操作类

class ParamsConfiguration 读写配置文件类

class ThreadTesting 测试类

类关系：



4、目标机亮度校正模块OrdinaryBrightmess

功能：寻找目标机灯光值，使目标机灯光效果与标准机一致

包含类

class ParamsConfiguration 读写配置文件类

class CalculateGrayscale 计算绿色分量类

class CHidCmdThread HID通讯类

class ProtocolUtility 通讯协议类

class HIDOpertaionUtility USB操作类

class OpencvUtility Opencv操作类

class LineFinder 寻找直线类

类关系：



系统实现

1、HID通讯模块 CHidCmdThread

1.1功能：HID命令排队操作线程，HID命令添加请一律通过该类进行处理

1.2数据结构

struct HIDCmdData //命令类型和命令数据集合

{

quint16 cmdType; //命令类型

QVector<QByteArray> byteArrayVect; //命令数据集合

};

1.3信号流向图

1.4私有槽

// HID命令结果返回信号处理槽

void SlotHIDCmdComplete(quint16 cmdType ,bool result);

1.5公有成员方法

void ClearCmd(); //清空HID命令

void AddCmd(HIDCmdData hidCmdData); //添加HID命令

void AddCloseHIDCmd(); //添加关闭HID命令

void AddResetMotorCmd(quint16 resetSpeed);// 添加马达复位命令

//添加马达转动命令

void AddRotateMotorCmd(quint16 speed, quint16 step, quint16 direction);

//添加开灯命令

void AddOpenLedCmd(int iLedIndex, quint16 iBrightness);

//添加写取设备参数命令

void AddWriteDevParamsCmd(DevConfigParams devConfigParams);

//添加不带命令数据的命令

void AddCmdWithoutCmdData(quint16 qCmdType);

//添加升级命令

void AddUpgradeSubControlCmd(QString qFilePathStr);

1.6私有成员变量

static CHidCmdThread\* s\_hidCmdThreadInstance; 通讯模块静态对象

QQueue<HIDCmdData> m\_hidCmdDataQueue; 上位机往设备下发命令队列

HIDCmdData m\_curHIDCmdData; 目前正在执行的命令数据

DevConfigParams m\_devConfigParams; 仪器参数

1.7 文件名

CHidCmdThread.cpp 、CHidCmdThread.h

1.8 文件位置

AdjustLight\..

2 class ProtocolUtility

2.1功能：封装上下位机通信协议

2.2数据结构

enum Direction //电机转动方向

{

DIRECTION\_FORWARD = 0, //正转

DIRECTION\_BACKWARD, //反转

};

2.3 成员方法

//LED灯开关命令

static QByteArray GetLEDCmd(quint16 ledIndex, quint16 brightness);

//电机转动命令

static QByteArray GetMotorRotateCmd(quint16 direction, quint16 step, quint16 speed);

static QByteArray GetMotorResetCmd(quint16 speed); //电机复位命令

static QByteArray GetCloseAllLEDCmd(); //关闭所有灯指令

static QByteArray GetCloseAllLEDAndStopMotorCmd(); //关闭所有灯与电机立刻停止

//读取仪器参数并记忆(总共包含10个包)

static QVector<QByteArray> GetReadParamsFromDevCmd();

//写仪器参数到设备(总共包含10个包)

static QVector<QByteArray> GetWriteParamFromDevCmd(DevConfigParams devConfigParams);

static QByteArray GetReadVersionCmd(); //读取仪器下位机软件版本

static QByteArray GetAddTestCountCmd();//仪器测量次数加一命令

static QByteArray GetReadTestCountCmd();//读取仪器测量次数

static QByteArray GetClearTestCountCmd();//仪器测量次数清零

static QByteArray GetUpgradeAppStartCmd();//仪器下位机程序升级开始命令

static QByteArray GetUpgradeAppEndCmd();//仪器下位机程序升级结束命令

//仪器下位机程序升级数据接收(包的总数根据升级文件大小而定)

static QVector<QByteArray> GetUpgradeAppCmd(QString filePath);

2.4 私有成员方法

static void SetCmdLength(QByteArray& dataByteArray); //设置命令长度

//填充命令公共收尾部分

static void DealWithCmdEnding(QByteArray& dataByteArray, QDataStream& out);

//填充命令公共头部

static void DealWithCmdHead(QDataStream& out);

//获取命令中不包含命令数据的命令ByteArray

static QByteArray GetCmdByteArrayWithoutCmdData(quint16 qCmdType);

2.5 成员变量

//上位机设备地址：0x0100

static const quint8 PC\_ADDR\_HIGH = 0x01;

static const quint8 PC\_ADDR\_LOW = 0x00;

//下位机设备地址：0x0101

static const quint8 DEV\_ADDR\_HIGH = 0x01;

static const quint8 DEV\_ADDR\_LOW = 0x00;

//命令类型

static const quint16 CMD\_DEV\_CLOSE = 0x0000;//关闭设备HID通信

static const quint16 CMD\_OPEN\_OR\_CLOSE\_LED = 0x0001;//LED灯开关命令

static const quint16 CMD\_ROTATE\_MOTOR = 0x0002;//电机转动命令

static const quint16 CMD\_RESET\_MOTOR = 0x0003;//电机复位指令

static const quint16 CMD\_CLOSE\_ALL\_LED = 0x0004;//关闭所有灯指令

static const quint16 CMD\_CLOSE\_ALL\_LED\_AND\_STOP\_MOTOR = 0x0005;//关闭所有灯与电机立刻停止

static const quint16 CMD\_WRITE\_PARAM\_TO\_DEV = 0x0006;//写仪器参数

static const quint16 CMD\_READ\_PARAM\_FROM\_DEV = 0x0007;//读取仪器参数

static const quint16 CMD\_READ\_DEV\_VERSION = 0x0008;//读取仪器下位机软件版本

static const quint16 CMD\_ADD\_TEST\_COUNT = 0x0009;//仪器测量次数加1

static const quint16 CMD\_READ\_TEST\_COUNT = 0x000A;//仪器测量次数读取

static const quint16 CMD\_CLEAR\_TEST\_COUNT = 0x000B;//仪器测量次数清零

static const quint16 CMD\_UPGRADE\_APP\_START = 0x000C;//仪器下位机程序升级开始

static const quint16 CMD\_UPGRADE\_APP\_DATA = 0x000D;//仪器下位机程序升级数据

static const quint16 CMD\_UPGRADE\_APP\_END = 0x000E;//仪器下位机程序升级结束

//命令参数

static const quint8 COMMON\_PARAM\_HIGH = 0x00;

static const quint8 COMMON\_PARAM\_LOW = 0x01;

2.6 文件名

ProtocolUtility .cpp 、ProtocolUtility .h

2.7 文件位置

AdjustLight\..

3 class HIDOpertaionUtility

3.1 功能：操作USB接口

3.2 信号流向图



3.3 公有信号

void SignalHIDOpen();//打开 HID信号

void SignalHIDClose(); //关闭 HID信号

void SignalHIDWrite(QByteArray writeByteArray);// HID写数据信号

void SignalReceiveDevVersion(QString devVersion);// HID读取下位机版本信号

void SignalReadDevParams(); // HID读取参数数据信号

void SignalErrInfo(EnumTypeErr qErrorMsg);// HID状态信号

void SignalOperationComplete(quint16 m\_iCmdType,bool result);// HID命令完成信号

3.4 私有槽

bool SlotOpen(); //打开USB HID

bool SlotClose(); //关闭USB HID

bool SlotWrite(QByteArray writeByteArray); //写命令到USB HID

bool SlotReadDevParams(); //读设备参数

bool SlotWriteDevParams(DevConfigParams devConfigParams);//写设备参数

bool SlotUpgradeSubControl(QString filePath);// 升级下位机

3.5 公有成员方法：

//目前适用于非读/写设备参数的其他控制指令

void HIDWrite(QByteArray writeByteArray);

void HIDOpen(); // HID 打开

//停止关闭操作比较特殊，需要立刻停止，这样之前的排队命令会被取消等待直接进行停止操作

void HIDClose();

bool HIDRead(quint8\* recvDataBuf, int delaytime); // HID读数据

bool IsDeviceOpen();// HID 是否打开

//处理从设备接收到的命令数据，处理在读取线程中

void ReceiveNewCmdFromDev(QByteArray data);

QString GetVersion();//获取设备版本

void HIDReadDevParams(); //读设备参数命令

void HIDWriteDevParams(DevConfigParams devConfigParams);// 写设备参数命令

void HIDUpgradeSubControl(QString filePath);//下位机升级

3.6 私有成员变量：

static HIDOpertaionUtility\* instance; // HIDOpertaionUtility实例

HANDLE mHidHandle; //USB hid 句柄

QThread mWorkThread;//工作线程，构造函数内使用moveToThread转到后台线程操作 Qt::HANDLE mWorkHandle;//工作线程句柄

volatile bool mIsDeviceOpened;//设备是否已打开

HIDReadThread\* mReadThread;//设备读线程

bool mAckResult;//ACK结果

bool mIsWaitForAck;//是否正在等待ACK,用于读写两个线程的信号同步

bool mReturnResult;//结果

bool mIsWaitForReturn;//是否正在等待结果，用于读写两个线程的信号同步

quint16 m\_iCmdType;//当前处理命令的命令类型

QString mDevVersion;//设备版本

DevConfigParams mParams;//设备参数

quint8\* mDevConfigParamsByte;//配置信息字节数组，用于每次从设备读取配置参数时临时存储数据

quint32 m\_qTestCount;//仪器测试次数

3.7 文件名

HIDOpertaionUtility .cpp 、HIDOpertaionUtility .h

3.8 文件位置

AdjustLight\..

4 class HIDReadThread

4.1 功能：USB读线程

4.2 文件名

HIDReadThread .cpp 、HIDReadThread .h

4.3 文件位置

AdjustLight\..

5、 二维码识别模块 QRCodeDetector

5.1 功能：从图片中提取二维码并识别、解析二维码数据，发送到测试线程。

5.2 数据结构

enum EnumTypeCup //杯类型

{

TypeTCup = 0, //圆杯（尿液）

TypeKCup5 = 1, //5联旋钮杯（尿液）

TypeSCup10 = 4, //10联唾液方杯（唾液）

TypeKCup6 = 6 //6联旋钮杯（尿液）

};

enum EnumTypeLight //灯光类型

{

DownLightGreen = 1, // 下方绿灯

DownLightWhite = 2, // 下方白灯

UpLightGreen = 3, // 上方绿灯

UpLightWhite = 4, // 上方白等

LeftLightGreen = 5, // 左方绿灯

LeftLightWhite = 6, // 左方白灯

RightLightGreen = 7, // 右方绿灯

RightLightWhite = 8 // 右方白等

};

struct InfoProject // 项目信息

{

int iIndexProgram; //项目序号

QString strProjectName; //项目名称

int dThresholdUp; //上阈值

int dThresholdDown; //下阈值

double dSensitivityUp; //上灵敏度

double dSensitivityDown; //下灵敏度

};

struct QRCodeInfo 二维码信息

{

QString iProductLot; // 产品批号（通过读取二维码自动生成）

QDate qExprationDate; // 产品有效期（通过读取二维码自动生成）

QString strProductID; // 产品ID（通过读取二维码自动生成）

EnumTypeCup eTypeCup; //杯类型

int iProgramCount; //项目总数

QString strVerson; //版本

QList<InfoProject> listProject; //项目信息

qint32 iQRCodePosition; //二维码位置（距离复位需要顺时针转的步数）

};

5.3 信号流向图



5.4公有信号

void SignalQRCodeLocation(QString pathPic); //发送二维码图片路径信号

void SignalQRCodeInfo(QRCodeInfo info); //发送二维码信息信号

void SignalErrInfo(EnumTypeErr eErr); //发送错误信息

void SignalGetQRCode(); //发送开始获取二维码信号

5.5 私有槽

void SlotGetQRcode(); //开始获取二维码

5.6公有成员方法

bool InitDevice(); //设备初始化

void TestGetQRCode();//获取二维码信息，发送获取二维码信号

//定位和识别二维码

bool locationQRCode(QString &strQRCodeInfo,qint32 &iQRCodePosition);

//获取二维码照片

bool GetQRCodeImage(QString &strImagePath);

//识别二维码

bool GetQRCodeImageInfo(const QString strImagePath,QString &strQRCodeInfo);

//根据V2或V5版本协议 解析二维码字符串

bool DecodeQrcode(const QString strdecode,QRCodeInfo &qrCodeInfo);

//寻找二维码定位矩形

bool FindQRcodeLocationRect(IplImage \*dstImg);

//开灯 成功 0 失败-1

int TestLightUp(EnumTypeLight type);

//关灯

int TestLightDown();

// 封装二维码信息

QRCodeInfo PackageQRCodeInfo(QString strBatchNumber, //项目批号

QString strValidityData, //有效期

QString strCardNumber, //卡序列号

QString strCupType, //杯型

QString strAllCount, //总子条数

QString strVersion, //版本

QVector<QStringList> qv\_strListitem); //项目信息

QString GetProjectName(const int iIndex); //解析项目名称

void mSleep(qint32 msec); //延时

//提取二维码

bool ExtractQRCode(QString strSrcImage,QString &strDesImage);

5.7 私有成员变量

二维码定位后，距离复位位置的距离（顺时针转到复位位置的步数）

int m\_iQRCodePosition;

//二维码解析库对象指针

QZXing \*m\_pZxingDecoder;

5.8 文件名

QRCodeDetector.cpp 、QRCodeDetector.h

5.9 文件位置

AdjustLight\.

6 class Opencvutility

6.1 功能：Opencv操作类，对摄像头和图片进行操作

6.2 信号流向图



6.3 公有信号

void SignalErrInfo(EnumTypeErr eErr); //发送错误信号

6.4 公有成员方法

bool OpenVideo();//打开摄像头

bool GetVideo();//录像

bool GetVideoCapture(QString \*strImagePath);// 拍照

6.5 私有成员变量

VideoCapture m\_VideoCapture; 视频操作类对象

static OpencvUtility\* instance; opencv操作类对象指针

6.5 文件名

Opencvutility.cpp 、Opencvutility.h

6.6 文件位置

AdjustLight\.

7 标准机亮度校正模块 StandardBrightness

7.1 功能：输入标准机灯光值，获取绿色分量曲线数据

7.2 结构体

struct BrightnessValue //标准机灯光结构体

{

int iNo1; // 上绿灯

int iNo2; // 上白灯

int iNo3; // 下绿灯

int iNo4; // 下白灯

int iNo5; // 左绿灯

int iNo6; // 左白灯

int iNo7; // 右绿灯

int iNo8; // 白右灯

int iCupType; //杯型

};

7.3 信号流向图



7.4 公有信号

//发送信号到UI界面更新

void SignalSendPictureToUI(QString strPicturePath,QList<int> iGreenComponuntList);

7.5 公有槽

void SlotSaveBrightnessValue(); //保存灯光结构体参数

7.6 保护槽

//接收界面设置的灯光结构体参数

void SlotGetBrightValue(BrightnessValue brightnessValue);

7.7 公有成员方法

//设置灯光值

bool SetBrightnessValue(BrightnessValue brightnessValue);

//获取绿色分量曲线

bool GetGreenComponunt(qint16 iCupType,const QString strSaveImagePath);

//保存到配置文件

bool SaveStandardParams(QString strFileName,QString ParamsType,BrightnessValue brightnessValue,QList<int> iGreenComponuntList);

//从配置文件读取参数

bool ReadStandardParams(QString strFileName,QString ParamsType,BrightnessValue &brightnessValue,QList<int> &iGreenComponuntList);

7.8 私有成员变量

BrightnessValue m\_brightnessValue; //标准机灯光参数结构体

QList<int> m\_iGreenComponuntList; //绿色分量链表

7.9 文件名

StandardBrightness.cpp 、StandardBrightness.h

7.10 文件位置

AdjustLight\.

8 目标机亮度校正模块OrdinaryBrightmess

8.1 功能：寻找目标机灯光值，使目标机灯光效果与标准机一致

8.2 数据结构

struct BrightnessOrdinaryValue //目标机灯光参数结构体

{

int iBrightNo1; //上绿灯

int iBrightNo2; //上白灯

int iBrightNo3; //下绿灯

int iBrightNo4; //下白灯

int iBrightNo5; //左绿灯

int iBrightNo6; //左白灯

int iBrightNo7; //右绿灯

int iBrightNo8; //右白灯

double iGreenComponentNo1; //上绿灯绿色分量

double iGreenComponentNo2; //上白灯绿色分量

double iGreenComponentNo3; //下绿灯绿色分量

double iGreenComponentNo4; //下白灯绿色分量

double iGreenComponentNo5; //左绿灯绿色分量

double iGreenComponentNo6; //左白灯绿色分量

double iGreenComponentNo7; //右绿灯绿色分量

double iGreenComponentNo8; //右白灯绿色分量

};

8.3 信号流向图



8.4 公有信号

//发送配置参数到UI更新

void SignalImportValueToUI(BrightnessOrdinaryValue brightnessValue);

//发送校准参数到UI更新

void SignalCalibrationValueToUI(BrightnessOrdinaryValue brightnessValue);

//发送读出来参数到UI更新

void SignalReadValueToUI(BrightnessOrdinaryValue brightnessValue);

8.5 公有槽

//接收UI校准按钮信号

void SlotOrdinaryCalibration();

//接收UI导入按钮信号

void SlotOrdinaryImport();

//接收UI保存按钮信号

void SlotOrdinarySave();

//接收UI读取按钮信号

void SlotOrdinaryRead();

8.6 公有成员方法

//获取绿色分量标准差最小的灯光值

bool GetLightValue(const int iBrightNo,QPoint qCenterPoint,const int iStandardBright,const double dStandardGreenComponent,int &iOrdinaryBright,double &dOrdinaryGreenComponent);

//保存到配置文件

bool SaveBrightnessValueParams(QString strFileName,QString ParamsType,BrightnessOrdinaryValue brightnessValue);

//读取配置文件

bool ReadBrightnessValueParams(QString strFileName,QString ParamsType,BrightnessOrdinaryValue &brightnessValue);

//初始化设备

bool InitMachine(QPoint \*CenterPoint);

//校准灯光

void OrdinaryCalibration(BrightnessOrdinaryValue \*brightnessValue);

//寻找C6 C8 色块

QPoint findCenterPoint(QString strImagePath);

8.7 私有成员变量

//目标机灯光参数结构体

BrightnessOrdinaryValue m\_OrdinarybrightnessValue;

//标准机灯光参数结构体

BrightnessOrdinaryValue m\_StandardMachinebrightnessValue;