**captureone 使用说明文档**

**一 结构体定义**

1. **typedef struct GetPointRect；**

|  |  |
| --- | --- |
| 参数 | 意义 |
| int rect\_x | 矩形左上角点的横坐标 |
| int rect\_y; | 矩形左上角点的纵坐标 |
| int rect\_weight; | 矩形的宽 |
| int rect\_hight; | 矩形的高 |

这个结构体用于传递矩形的信息。

1. **typedef struct FindCouterRangle；**

|  |  |
| --- | --- |
| 参数 | 意义 |
| int fincout\_x; | 矩形限制条件，要求的最低宽，默认为100像素 |
| int fincout\_y; | 矩形限制条件，要求的最低高，默认为100像素 |
| int findcut\_size; | 矩形限制条件，要求的最小面积，默认为10000 |

作为矩形筛选条件，因为不进行筛选会有许多大大小小不一的矩形，对于检测不好；默认情况下图像中只有一个符合条件的矩形。

1. **typedef struct FindChipPostion；**

|  |  |
| --- | --- |
| 参数 | 意义 |
| float angel; | 检测到矩形的偏转角度，小于5度的可以忽略 |
| int center\_x; | 检测到矩形的中心横坐标 |
| int center\_y; | 检测到矩形的中心纵坐标 |

用来返回检测得到的矩形的信息；

**二 Class CaptureOne的函数使用**

**1 CaptureOne();**

类的构造函数，初始化一些私有成员；

**2 ~CaptureOne();**

类的析构函数。

1. **void CaptureIndex(int index);**

|  |  |
| --- | --- |
| 参数 | 意义 |
| int index | 输入，摄像头硬件设备的ID号 |
| 无返回值 |  |

这个是静态属性，必须在调用run()之前使用，可以重复设置但是必须在没有运行run之前。或者关闭摄像头之后stopCapture()之后重新设置

1. **bool iscapture\_openOK();**

|  |  |
| --- | --- |
| 参数 | 意义 |
| 返回值 bool | true可以打开摄像头 false 不能打开摄像头 |

静态查询，在调用CaptureIndex（int index）之后检测设置的摄像头ID是否正确，调用之后不能再重新设置CaptureIndex（int index）,需要关闭摄像头才能重新设置

1. **void startrun();**

|  |  |
| --- | --- |
| 参数 | 意义 |
| 无返回值 |  |

开启摄像头运行的主线程，直接调用就可以不需要另外开启线程

1. **bool capturestatus();**

|  |  |
| --- | --- |
| 参数 | 意义 |
| bool返回值 | 检测摄像头的运行状态，tru运行正常，false摄像头出问题，用来判断摄像头是否断开。 |

建议使用一个线程在开启摄像头之后一直访问获取这个状态，用来判断摄像头是否接触不好，从而进行程序重启操作（而不是机器重启）

1. **void SetRectRangle(GetPointRect rectrangle);**

|  |  |
| --- | --- |
| 参数 | 意义 |
| GetPointRect rectrangle | 详见结构体typedef struct GetPointRect |
| 无返回值 |  |

设置图像限定区域，即ROI，默认是原始图像，注意大小不能超过原始图像的大小；动态属性设置，可以在RUN之前设置，也可以在run之后设置

1. **void SetSrcImage();**

|  |  |
| --- | --- |
| 参数 | 意义 |
| 无返回值 |  |

设置图像为原始图像的大小，取消SetRectRangle的限定范围；

1. **int SetThreshold(int mythreshold);**

|  |  |
| --- | --- |
| 参数 | 意义 |
| int mythreshold | 输入，二值化图像的阈值，默认为180 |
| 返回值int | 0设置成功，-1设置不成功（超出范围1到254） |

这个是动态属性，同上

1. **int SetPhologyType(int type);**

|  |  |
| --- | --- |
| 参数 | 意义 |
| int type | 输入，形态学变化的类型，默认为0，范围0~5 |
| 返回值int | 0设置成功，-1设置不成功（超出范围） |

动态属性

1. **Mat getFrameOut ();**

|  |  |
| --- | --- |
| 参数 | 意义 |
| 返回值 Mat | 返回的是Mat类型的图片 |

返回的图片类型为Mat,不同的软件显示需要自己转换格式。

1. **void SetDrawResult();**

|  |  |
| --- | --- |
| 参数 | 意义 |
| 无返回值 |  |

显示检测的结果，图像中会用框画出来

动态属性

1. **void StopDrawResult();**

|  |  |
| --- | --- |
| 参数 | 意义 |
| 无返回值 |  |

不显示检测的结果

1. **int WriteMatchTemp();**

|  |  |
| --- | --- |
| 参数 | 意义 |
| 返回值int | 0 写图片成功 -1不成功 |

用于获取模板图片的函数，会自动写到目录下，matchtemp.jpg，一张用于比较的原始图片。在run之后调用，（确保检测出来的矩形是正确的）

1. **int CalSrc\_PHashCode();**

|  |  |
| --- | --- |
| 参数 | 意义 |
| 返回值int | 0计算成功 -1计算不成功（无图片） |

基于比较算法，需要一张原始图片，有WriteMatchTemp（）函数获得，不成功·即是不存在原始图像，需要重新写一张，在run之后调用

1. **int getHammingDistance(int \* d);**

|  |  |
| --- | --- |
| 参数 | 意义 |
| int \* d 整数指针 | 输出，返回计算得到的结果值，理论上0~5相似度比较接近，越大相似度越大，用这个值来判断芯片上面是否有贴条，或者涂改，（差距比较大的时候） |
| 返回值int | 0比较成功， -1比较失败（用来比较的图像获取不到）； |

在run之后调用

1. **void SetFindCouter(FindCouterRangle findcouterrangle);**

|  |  |
| --- | --- |
| 参数 | 意义 |
| FindCouterRangle findcouterrangle | 输入，详见结构体typedef struct FindCouterRangle |
| 无返回值 |  |

设置矩形筛选的条件,动态属性

1. **int getSingleRectangle(FindChipPostion \* findchippostion);**

|  |  |
| --- | --- |
| 参数 | 意义 |
| FindChipPostion \* findchippostion | 输出，详见结构体typedef struct FindChipPostion |
| 返回值int | 0获取成功 -1获取失败（检测不到矩形） |

在run之后调用

1. **void stopCapture();**

|  |  |
| --- | --- |
| 参数 | 意义 |
| 无返回值 |  |

关闭线程run

1. **int capturerealse();**

|  |  |
| --- | --- |
| 参数 | 意义 |
| 返回值 int | 0摄像重置零成功 -1 摄像头重置不成功（没有关闭线程run） |

如果调用了iscapture\_openOK（）发现设置不成功，调用这个函数置零，重新调用CaptureIndex（）和 iscapture\_openOK（）

1. **void StartIsp();**

|  |  |
| --- | --- |
| 参数 | 意义 |
| 无返回值 |  |

开启调试模式，前期用于检测所设置的参数是否具有意义，会显示处理的图像，前期用来调试图像的一些参数；

1. **void StopIsp();**

|  |  |
| --- | --- |
| 参数 | 意义 |
| 无返回值 |  |

关闭调试模式

**三 sample以及demo**

**1调用顺序**

CaptureOne captureone;

int ret;

bool bfret;

GetPointRect rectrangle

{

………………………….

}

captureone. CaptureIndex(0);

bfret = captureone.iscapture\_openOK();

if(true != bfret)

{

//这里也可以重新设置 参考20

break;

}

captureone. SetRectRangle(rectrangle);

captureone. SetThreshold(mythreshold);

captureone. SetPhologyType(type);

captureone. SetFindCouter(findcouterrangle);

captureone.start();

captureone. SetDrawResult();

开启start之后就可以获取图像显示了

**2 计算芯片整洁度**

ret = captureone.CalSrc\_PHashCode();

if(ret != 0 )

{

ret = captureone. WriteMatchTemp();

if(ret != 0)

{

return -1;

}

}

ret = captureone.CalSrc\_PHashCode();

if(ret != 0 )

{

return -1;

}

在run之后如果要检测芯片的整洁度，确保有图片存在

//之后可以在另一个线程里获取比较结果

ret = captureone. getHammingDistance (&num);

if( ret != 0)

{

do again;

}

if(num > stdnum )

{

//图片相似度不一样

}

else{

//图片相似度一样

}

1. **检测摄像头状态**

…………

while(1)

{

bool isrun = captureone. capturestatus();

if(false == isrun)

{

//报警 摄像头出问题

// 重新拔插之后可以再一次程序重启摄像头

//调用1

}

}

5 其它的数据获取例如例如检测到的芯片FindChipPostion 如同之前的接口说明文档就可以了