## 切割机接口说明文档

**2019-04-02 王修文**

**一 宏定义与结构体**

1 #define CPATURE\_WIDTH 640 ，默认情况下没有设置摄像头的参数，其获取到的图像都是640 \*480的规格 #define CPATURE\_HEIGHT 480， 当然目前也没有提供可以设置摄像头参数的接口，主要是摄像头的参数还没有程序可以准确的获取；

**2 typedef struct RectangleParam;**

|  |  |
| --- | --- |
| 参数 | 意义 |
| int rect\_x | 矩形的横坐标 |
| int rect\_y | 矩形的纵坐标 |
| int rect\_width | 矩形的宽 |
| int rect\_height | 矩形的高 |

结构体用来获取或者设置矩形的属性，可以用来限定区域范围，也可以用来获取得到的矩形区域；

**二 CaptureCutting接口说明**

**1 void SetCaptureIndex(int index);**

|  |  |
| --- | --- |
| 参数 | 意义 |
| int index | 设置摄像头的设备ID号 ，输入 |
| 无返回值 |  |

用来设置摄像头的设备ID号，这个是静态属性，再调用StartRun（）之前调用

静态属性：只能在线程StartRun（）跑起来之前设置，并且运行中不能改变

动态属性：能在线程StartRun（）跑起来之后继续调用设置，可以重复调用

**2 bool CheckCaptureAble();**

|  |  |
| --- | --- |
| 参数 | 意义 |
| bool返回值 | true摄像头开启正常 false摄像头开启错误 |

检测设置ID之后的摄像头能否正常开启，静态熟悉

注意在设置void SetCaptureIndex(int index)之后调用，如果错误请调用int RealeaseCapture()释放摄像头资源在重新调用void SetCaptureIndex(int index)设置。

**3 int RealeaseCapture();**

|  |  |
| --- | --- |
| 参数 | 意义 |
| int 返回值 | 0 成功释放摄像头资源 -1释放失败 |

用来释放摄像头资源，可以调用这个函数之后重新设置摄像头ID号，以此来切换摄像头，但是注意需要停止StartRun()线程，不然释放会失败；

**4 void StartRun();**

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

这个是一个开启摄像头的线程，里面是调用void TrueRun()私有函数，因为目前有MFC和QT两种版本，所以这是一个修改的函数，这里实现的是C++11新添加的thread库，如果调用有问题，可以把私有函数void TrueRun()改成public,MFC调用 DWORD WINAPI FunctionName 来调用TrueRun, 而QT可以直接继承Qthread类；总之线程的调用修改比较灵活，另外可能里面的锁mutex，也需要修改一下，如果修改了注意头文件的加减。

注意：一个类同时只能调用一次

**5 int ReadMatchTemple();**

|  |  |
| --- | --- |
| 参数 | 意义 |
| int 返回值 | 0读取图片成功 -1读取图片成功 |

读取匹配图片的模板，模板是之前制作好的，以后会添加手动设置的功能，在StartRun()之后调用，一定要调用一次，如果返回错误，就是图片不存在；

**6 int SetRectROI(RectangleParam &rectroi);**

|  |  |
| --- | --- |
| 参数 | 意义 |
| RectangleParam &rectroi | 详见结构体RectangleParam ， 输入 |
| int返回值 | 0 设置成功 -1设置失败 |

用来缩小图片的参数，就是取原图的一部分进行计算，这样运算速度就可以提高，这是个动态属性，设置不成功一般是给的限制范围不正确， 动态属性

**7 int SetMatchType(int matchtype);**

|  |  |
| --- | --- |
| 参数 | 意义 |
| int matchtype | 默认值为3， 设置匹配的方法 ，范围为0~5 |
| int 返回值 | 0 设置成功 -1设置失败 |

模板匹配的匹配算法，目前测算出来的3默认值是最好的，不用可进行修改，设置失败是因为越界了。动态属性

**8 int GetMatchPoint(cv::Point &matchpoint);**

|  |  |
| --- | --- |
| 参数 | 意义 |
| cv::Point &matchpoint | 输出，得到匹配点的坐标，左上角 |
| int返回值 | 0 获取成功 ，-1获取失败 |

只有成功返回的值才有意义；这个是调用的主要对象，获取匹配的位置，即后续程序用来计算坐标的关键接口；动态属性

**9 void StopRun();**

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

关闭线程StartRun()，关闭之后可以重新开启，注意要调用设置的还是需要设置一边

**10 cv::Mat GetFrame();**

|  |  |
| --- | --- |
| 参数 | 意义 |
| 返回值Mat | 返回当前的画面 |

如果需要显示的图像嵌入软件之类，需要调用此函数获取图像。

**11 void StratDeBugShow();**

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

这个是用来直接显示当前画面的窗口，以方便调试 ，动态属性

**12 void StopDeBugShow();**

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

关闭显示窗口 动态属性

**13 int WriteImageCurrent();**

|  |  |
| --- | --- |
| 参数 | 意义 |
| int1返回值 | 0图像截取成功 -1图像截取失败 |

用来截取当前图像，会在当前目录生成src.jpg，利用src.jpg编辑截取适当范围可以用作模板图片matchtemple.jpg ，可以重复调用

**14 void UnSetRectROI();**

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

取消设置的ROI范围，图片以原图来计算。

**三 调用sample**

|  |
| --- |
| CaptureCutting captureCutting;  captureCutting. SetCaptureIndex(0);  int ret;  bool bret;  bret = captureCutting. CheckCaptureAble();  if(!bret)  {  captureCutting. RealeaseCapture();  //重新设置 或者报错  //，，，，，，，，，，，，，，，，，， }  captureCutting. StartRun();  ret = captureCutting. ReadMatchTemple();  if(0 != ret )  {  // printf(“error matchtemple.jpg is not exist \n”);  //需要制作 matchtemple.jpg 如果返回错误-1 之后的函数就不能调用会出错  }  //之后就可以在别的地方设置或者调用其动态属性的函数 |

**制作matchtemple.jpg图片**

在上面的ret得到为非0之后，调用int WriteImageCurrent();

然后在当前目录下找到src,jpg图片，右键点击编辑，使用裁剪功能裁剪模板，并且命名为matchtemple.jpg

如下图

