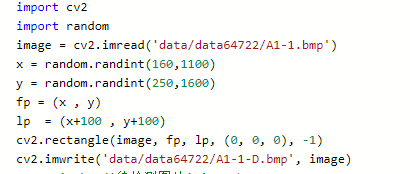
**BottlePrintCheck项目报告**

1. **随机缺陷生成**
2. 随机缺陷生成代码



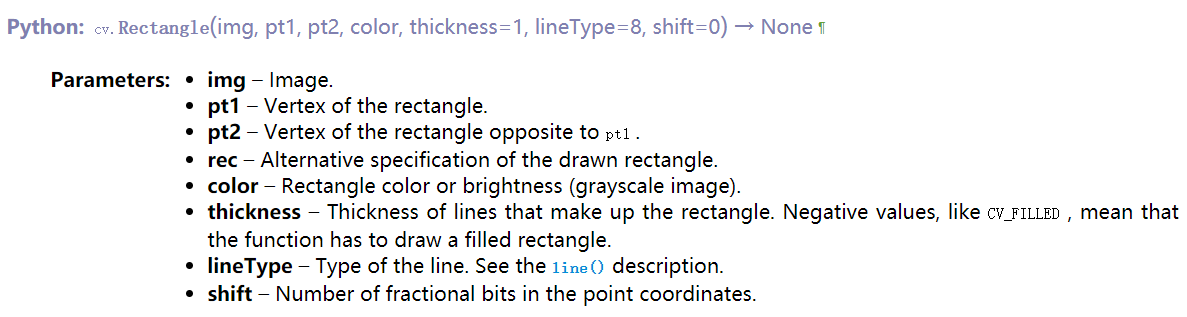
1. 随机缺陷生成原理

在图片内部（待检测部分）随机画一个大小为100\*100的实心黑色矩形，如下图：



1. 随机缺生成的具体实现

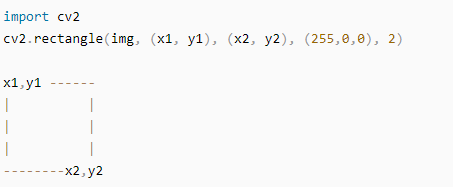
随机缺陷生成采用OpenCv里的cv2.rectangle函数。该函数函数的作用是在图像上绘制一个简单的矩形。具体参数解释如下：



本方法所用的参数为：

1）img：该参数为需要生成缺陷的图片路径

2）pt1，pt2：这两个参数用来确定所绘画矩形的大小以及位置。详情如下：



从这里可以看出，pt1主要确定位置本方法对应于fp，pt2主要确定大小本方法中对应于lp。

在本方法中为了使缺陷的位置能够随机生成，采用了random方法生成随机数：



这样就能确保位置随机。

为了使大小确定我们通过如下方式确定lp：



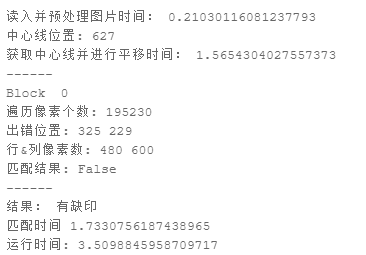
这样每个矩形的大小均是100\*100。

1. Color：该颜色参数需要RGB值，黑色的RGB值为（0，0，0）
2. Thickness：该参数用来确定绘制矩形的线条的粗细。由于需要绘制实心的矩形因此该参数值为-1。
3. **缺陷检测时间分析**
4. **待检测图片缺陷位置靠上且待检测图片与模板图片相比不偏移**



待检测图片 模板图片

**运行时间与结果：**



1. **待检测图片缺陷位置靠下且待检测图片与模板图片相比不偏移**



待检测图片 模板图片

**运行时间与结果：**

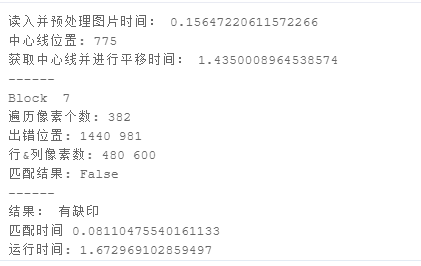


1. **待检测图片缺陷位置靠上且待检测图片偏移**



待检测图片 模板图片

**运行时间与结果：**

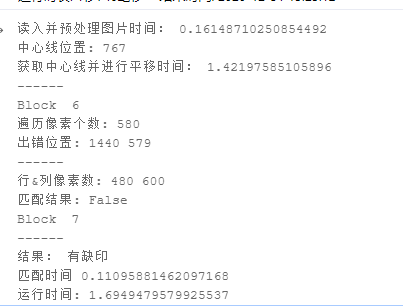


1. **待检测图片缺陷位置靠下且待检测图片偏移**



待检测图片 模板图片

**运行时间与结果：**



**注：由于待检测图片偏移角度过大而**

1. **待检测图片无缺陷且不偏移**



待检测图片 模板图片

**运行时间与结果：**

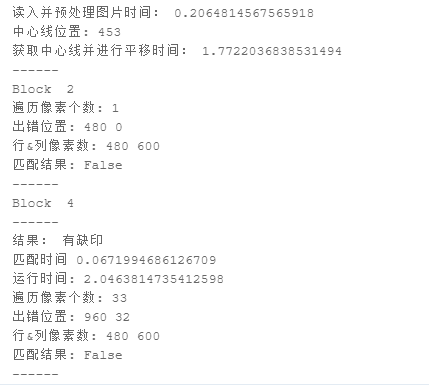


1. **待检测图片无缺陷且偏移**



待检测图片 模板图片

**运行时间与结果：**



**总结：**

**精确度：**

经过如上分析我们可以看到无论缺陷位置如何，**待检测图片不偏移**，该方法均可以精确的检测出缺陷。但是**当待检测图片发生偏移时，**该方法在检测时精确度不够，考虑其原因可能是待检测图片偏移角度过大有些字母没有拍摄进去。如下图：



待检测偏移图片 不偏移的图片

可以看到由于拍摄角度的问题箭头指向的位置‘N’并没有拍摄进去，因此检测的时候出现了检测失败的问题。

**运行时间：**

**项目环境为：**



**经过分析，出现缺陷的位置会影响检测的时间，具体为：**

1. **缺陷位置越偏下时间越长。**
2. **无缺陷的检测时间最长，在本项目环境下时间为8.56s。在普通机器上时间长达15s，因此使用GPU可以提高运行时间。考虑最坏情况本项目的运行时间不会高于8.56s，最快情况下可以达到1.5s。**