工作记录

昨天的工作是编写图片切割组合的代码，昨天事物过多，虽然完成了，但是效果不好，速度缓慢（比之前快，但是速度不快），今天采用新的方式进行实现。

今日任务：

1.编写新版的图片切割组合代码，实现加速停车场图片生产效率

2.所有图片处理代码组合，图形化界面编写

3.相关文章文献阅读

实现：

1．编写新版图片切割组合代码，实现停车场图片生产效率

已经实现，速度非常快，具体速度未统计,50 分钟 6000 张图片

具体实现思路：

将停车场图片中相应停车位块切割下来，黏贴到空白的停车地面上，生产全新图片，所有过程随机化，保证图片的可靠性

实现代码：

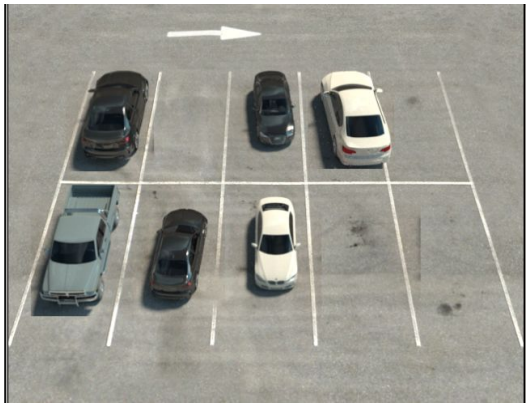
```
posis_=parsexml(Xml_fileName)
print posis_

rgbimg_pathes,rgbimg_names=walk_dir(RGB_IMG_DIR)
binaryimg_pathes,binaryimg_names=walk_dir(Binary_IMG_DIR)

for car_index in range(car_num):
    binaryimg_pathes_new=[]
    rgbimg_pathes_new=random.sample(rgbimg_pathes,10)
    for rgbimg_path in rgbimg_pathes_new:
        binaryimg_path=rgbimg_path.replace('images','images3',1)
        binaryimg_pathes_new.append(binaryimg_path)
    #binaryimg_pathes=random.sample(binaryimg_pathes,10)
        print binaryimg_path,rgbimg_path
    posi_index=0
    for rgbimg_path in rgbimg_pathes_new:
        rgbimg_path_load=cv2.imread(rgbimg_path)
        binaryimg_path_load=cv2.imread(binaryimg_pathes_new[posi_index])
        img_rgb_copy[int(posis_[posi_index][1]):int(posis_[posi_index][3]),int(posis_[posi_index][0]):int(posis_[posi_
        img_binary_copy[int(posis_[posi_index][1]):int(posis_[posi_index][3]),int(posis_[posi_index][0]):int(posis_[po

        if save_img(img_rgb_copy,os.path.join(output_dir_rgb_img,str(car_index+1)+'.png')):
            print
        else:
            print 'fail save rgb img \n'
        if save_img(img_binary_copy,os.path.join(output_dir_binary_img,str(car_index+1)+'.png')):
            print
        else:
            print 'fail save binary img\n'
        posi_index=posi_index+1
    print   os.path.join(output_dir,'image/'+str(car_index)+'.png')
    print 'the '+str(car_index+1)+' save success\n'
```

实现效果：



2.所有图像处理代码融合，界面编写

a) 功能确定

i. 图像归一化大小　灰度处理　二值化　目标位置计算等（同一目录即可实现）

ii. 图像切割和拼接

iii. 图片标记

b) 界面编写

c) 功能代码复制优化

3.文献查看