工作记录

昨天工作中对代码的切割的代码已经写好，今天的主要工作是：

1．对切割后的黑白图像中车辆位置进行计算，同时还原车辆位置

2．深度信息图像的切割保存和原始图像的切割保存

3．代码的整理和用户界面的编写

注：之前写过很多类似的数据和图像处理工具，但是却没有进行好好整理过，每次使用都必须更改部分参数，浪费大量精力（主要没有审美能力，编写界面太丑，编写完成后自己都不愿意使用），这次编写是对之前代码的整理，也是对自己设计简单易用界面的的一次练习。

工作实现：

1．对切割后的黑白图像中车辆位置进行计算，同时还原车辆位置（车辆生产未完成，只使用部分做实验）

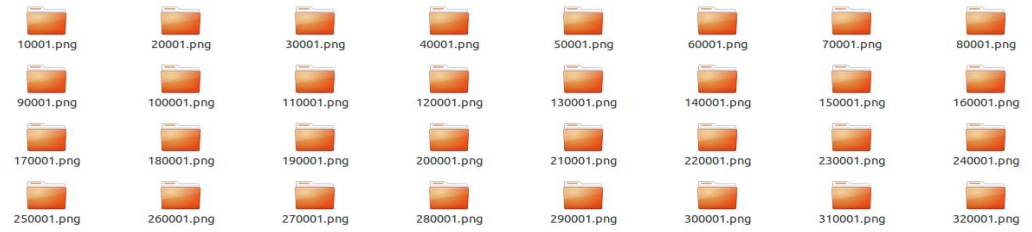
i. 所有黑白图像切割与图像保存

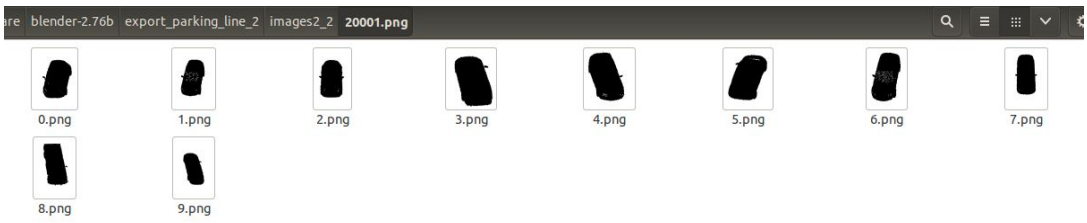
实现如下：

```
for f_path in file_pathes:
    img_test=cv2.imread(f_path)
    img_crop=img_test.copy()
    file_dir_create=os.path.join(file_dir2,file_names[f_index])
    if os.path.exists(file_dir_create):
        print
    else:
        os.mkdir(file_dir_create)
    for i in range(0,len(posis_)):
        cut_img=img_crop[int(posis_[i][1]):int(posis_[i][3]),int(posis_[i][0]):int(
        newImg_name=os.path.join(file_dir_create,str(i)+'.png')
        if save_img(cut_img,newImg_name):
            print 'sava the '+str(f_index+1)+' img success'

    f_index=f_index+1
```

结果如下：





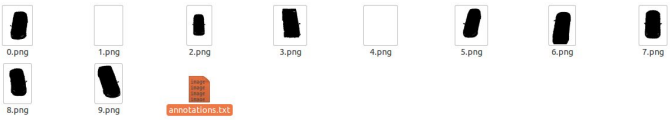
ii. **所有切割后的图像的位置计算**

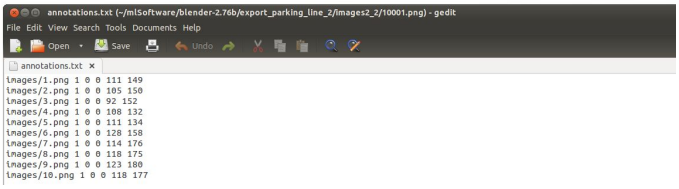
实现：

```
png_depth_reader = png.Reader(newImg_name)
zbuf=png_depth_reader.read_flat()
zbuf_w = zbuf[0]
zbuf_h = zbuf[1]
zbuf_p = zbuf[2][::3]
bounds = [0, 0, 0, 0]
# top
for y in range(0, zbuf_h):
    flag = False
    for x in range(0, zbuf_w):
        if zbuf_p[y * zbuf_w + x] < 65535:
            bounds[0] = y
            flag = True
            break
    if flag == True:
        break


# right
for x in range(zbuf_w - 1, -1, -1):
    flag = False
```

效果：





注：最终结果均为图片的大小，结果有错误,寻找其他方法

实现：

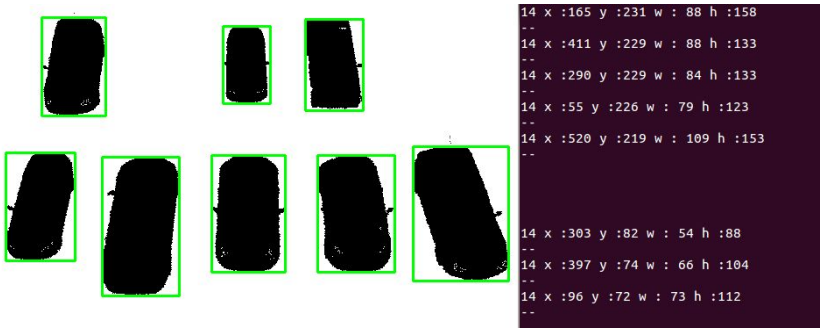
```
def get2objects(mask,original):
    cnts= cv2.findContours(mask.copy(),  cv2.RETR_EXTERNAL, cv2.CHAIN_AP
    cnts = cnts[0] if imutils.is_cv2() else cnts[1]
    print len(cnts)
    for c in cnts:
        x,y,w,h = cv2.boundingRect(c)

        if (w==1) or (h==1):
            print
        else:
            cv2.rectangle(original,(x,y),(x+w,y+h),(0,255,0),2)
            #get position and width and height
            print(str(len(cnts))+' x :'+str(x)+' y :'+str(y)+' w : '+str
    cv2.drawContours(mask,cnts,-1,(0,255,0),3)

    cv2.imshow("img", mask)


frame = cv2.imread('10001.png')
getCoordinates(frame)
k = cv2.waitKey(0)
if k == 27:         # wait for ESC key to exit
    cv2.destroyAllWindows()
```

打印结果：



iii. **所有切割后的图像的位置还原**

采用新的方法后，不需要还原，直接得到在图中的位置信息：

```
14 x :411 y :229 w : 88 h :133
--
14 x :290 y :229 w : 84 h :133
--
14 x :55 y :226 w : 79 h :123
--
14 x :520 y :219 w : 109 h :153
--



14 x :303 y :82 w : 54 h :88
--
14 x :397 y :74 w : 66 h :104
--
14 x :96 y :72 w : 73 h :112
```

注：x y 坐标和宽度高度

2．深度信息图像的切割保存和原始图像的切割保存

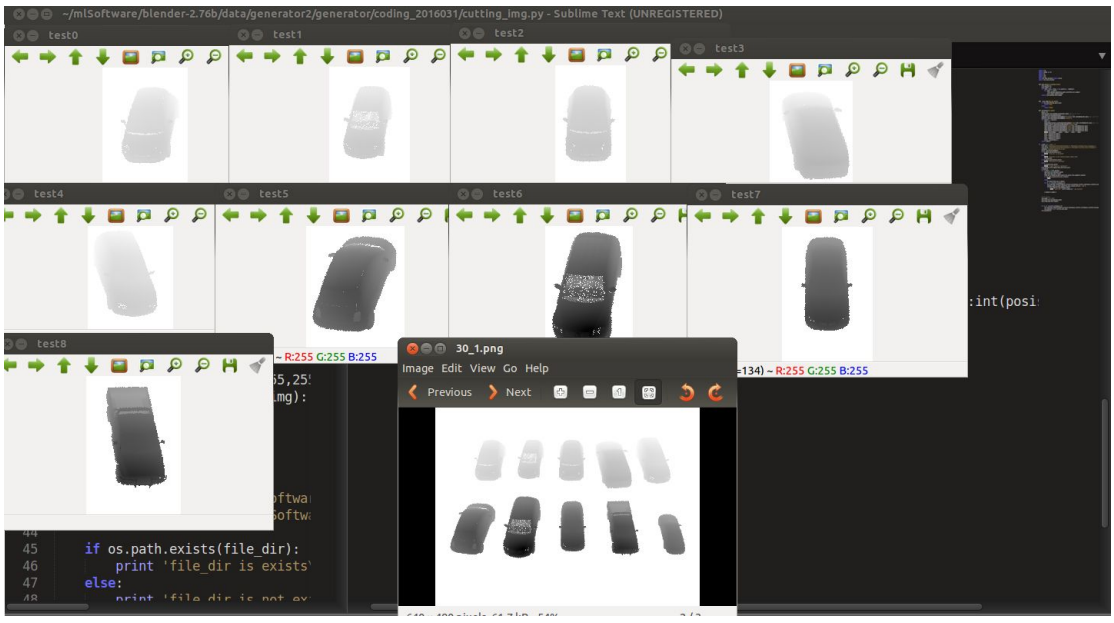
i. **深度信息图像切割**

实现：使用之前的代码进行切割

```
img_test=cv2.imread(img_name)
img_crop=img_test.copy()


for i in range(0,len(posis_)):
    cut_img=img_crop[int(posis_[i][1]):int(posis_[i][3]),
    cv2.imshow('test'+str(i),cut_img)
cv2.waitKey(0)
```

效果：



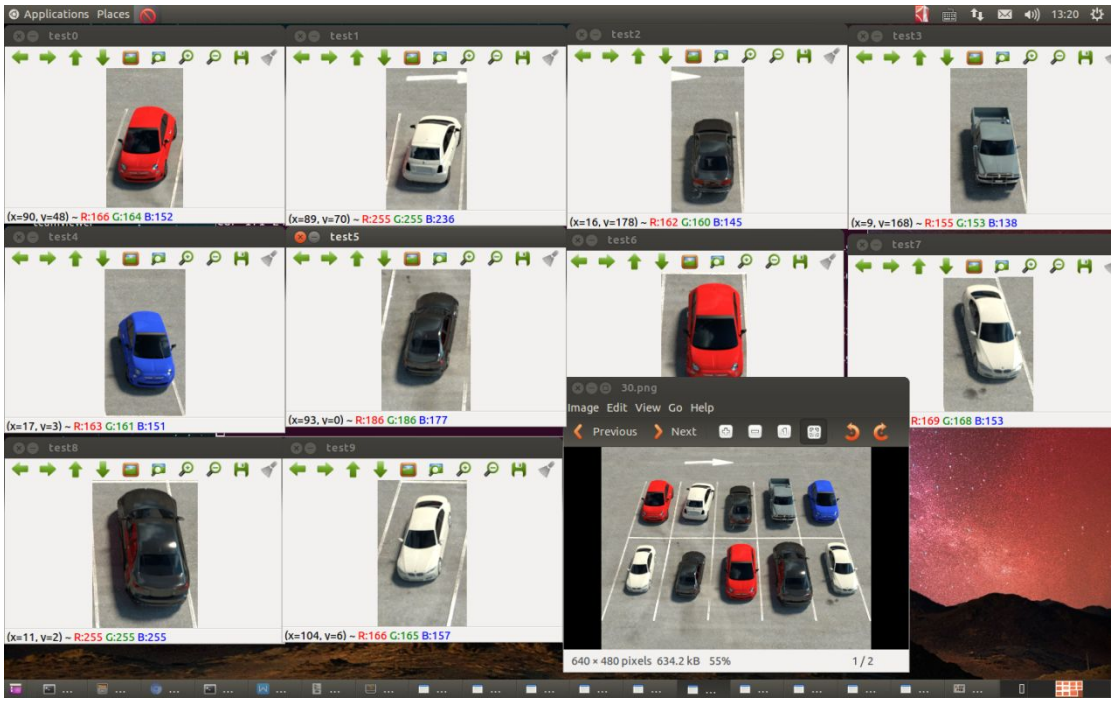
ii. **原始图像切割**

实现：使用之前的代码进行切割

```
img_test=cv2.imread(img_name)
img_crop=img_test.copy()


for i in range(0,len(posis_)):
    cut_img=img_crop[int(posis_[i][1]):int(posis_[i][3]),
    cv2.imshow('test'+str(i),cut_img)
cv2.waitKey(0)
```

效果：



3．代码的整理和用户界面的编写

应用包含功能：

1．图片批量处理（深度图片变黑白图片　黑白图片车辆位置计算　图片归一化处理...）

2．处理过程可视化

3．图片简单编辑

4.生产 10-80 度的摄像头照片：

i. **使用递推公式计算，摄像头位置和角度变化关系，不可行**

ii. **使用分段方式处理（10-80 度分段计算摄像头位置和角度的关系）**