工作记录

1．Blender 图片生产：

a.要求

i. 不同角度 30 40 50 60

ii. 车辆位置颜色光照全部随机

iii. 车辆种类随机排布

iv. 车位数量 2 5 左右，车辆随机率为 0.8

注：确定图片生产数量和图片生产时间，为后期优化提供帮助

b.实现：

使用 func2，只需加入角度设置即可，定义 3 个角度的数组，随机调用，确定调用的概率分配平衡，保证分布

昨天遇到的问题：

1．模型非正面，有斜角度，摄像头调整到正面，拍摄模型仍然是斜面，切摄像头无法调整至与模型平行（与模型面平行，即可得到正面视图）

2．摄像头焦距比较难选择合适值

解决办法：

1．使用代码调整，得到合适的摄像头参数

2．其他方法：手动调整或者更换摄像头

1．使用代码进行调整

主要设置摄像头映射到地面的位置 loaction 和摄像头的偏转方向和角度 rotation_euler

代码如下：

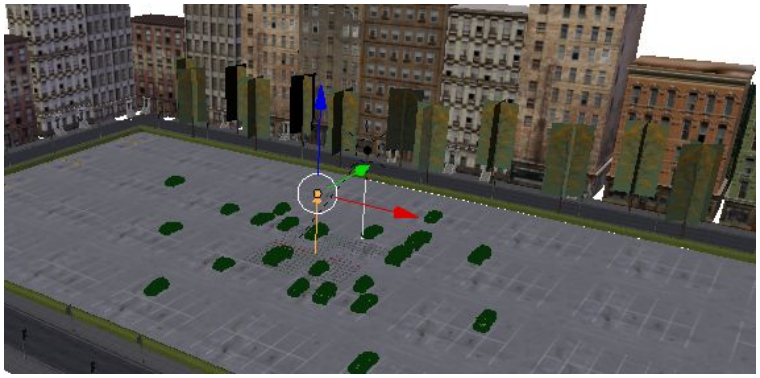
```
#set camera  location
bpy.data.objects['Camera'].location = [-18,0 , 75]  # x y z
bpy.data.objects['Camera'].rotation_euler = [math.radians(0),math.radians(0), math.radians(0)]
```

Location 3 个参数为 x y z 三个方向，z 是离地面高度

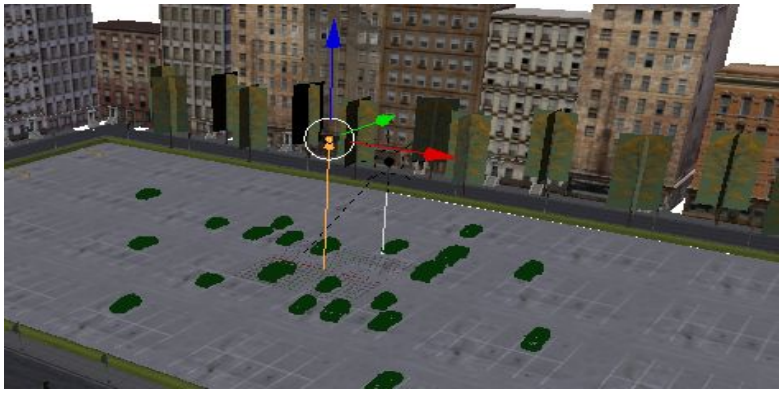
Rotation_euler 参数也为 x y z 3 个方向，x y 是摄像头偏向，z 是摄像头旋转

效果如下：

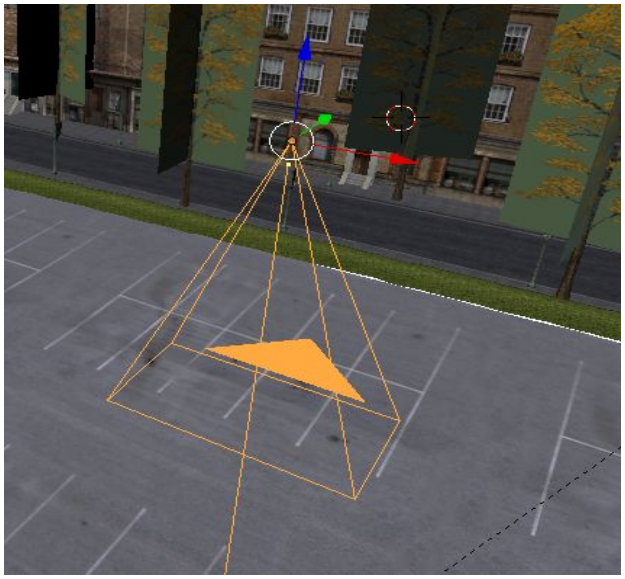
1．bpy.data.objects['Camera'].location = [0, 0, 10]



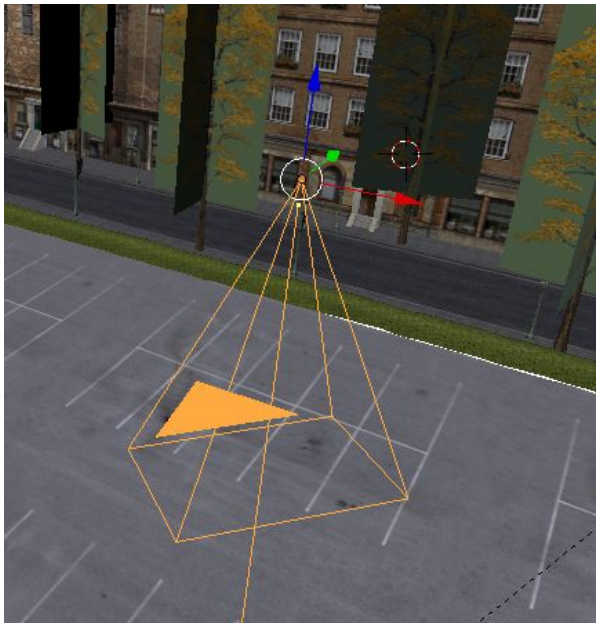
2．bpy.data.objects['Camera'].location = [0, 0, 20]



3.bpy.data.objects['Camera'].rotation_euler=[math.radians(0),math.radians(0), math.radians(0)]



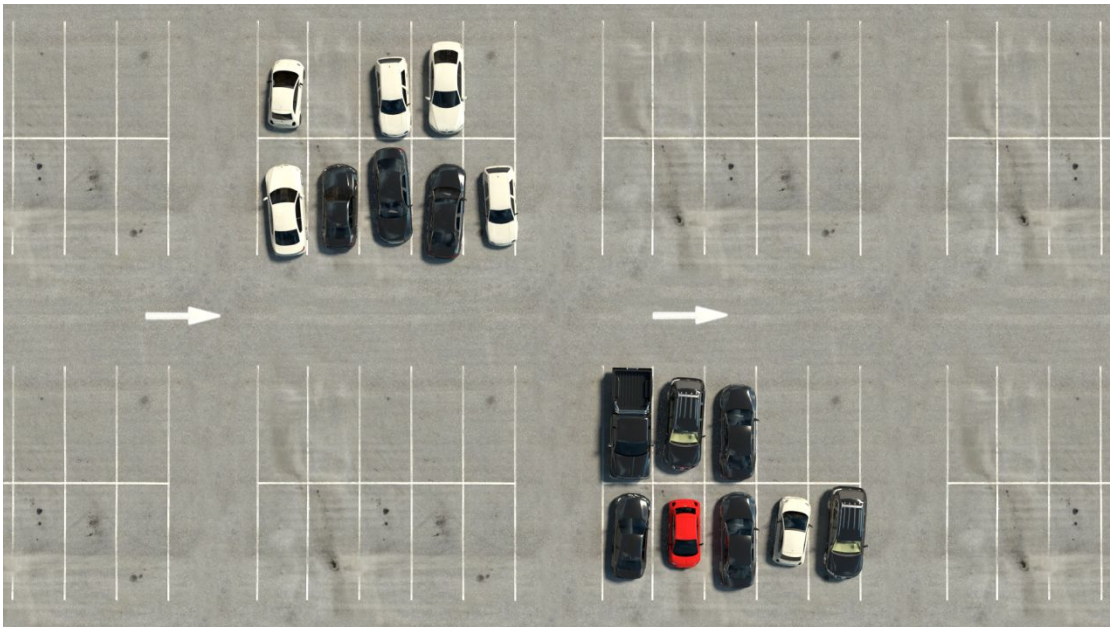
4.bpy.data.objects['Camera'].rotation_euler = [math.radians(0), math.radians(0), math.radians(50)]



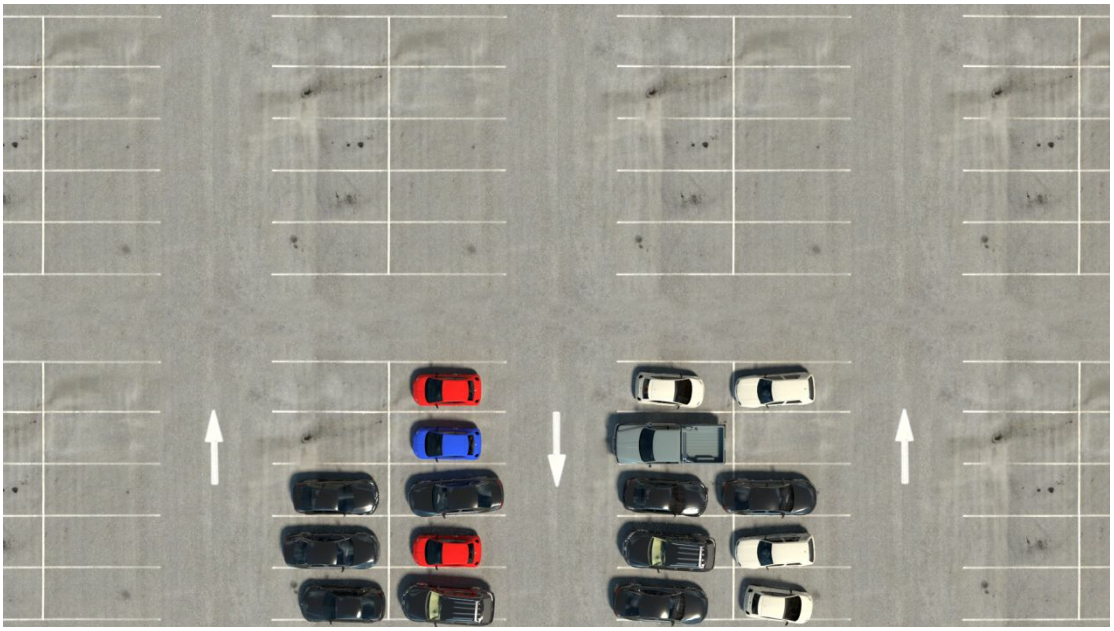
注：其他参数也如此，rotation 的 z 是摄像头旋转

如俯视图所示(bpy.data.objects['Camera'].location = [-18,0 , 75])：

1.bpy.data.objects['Camera'].rotation_euler=[math.radians(0), math.radians(0), math.radians(0)]



2.bpy.data.objects['Camera'].rotation_euler=[math.radians(0), math.radians(0), math.radians(90)]



注，要得到正视图，需要调整ｘｆ方向的视角，本次采用自动调整方式，ｘ　ｙ方向视角自动递增，人工进行挑选：代码如下：

bpy.data.objects['Camera'].rotation_euler=[math.radians(45), math.radians(0), math.radians(0)]

bpy.data.objects['Camera'].location = [-ｉ,-30-i, 42]　# x y z

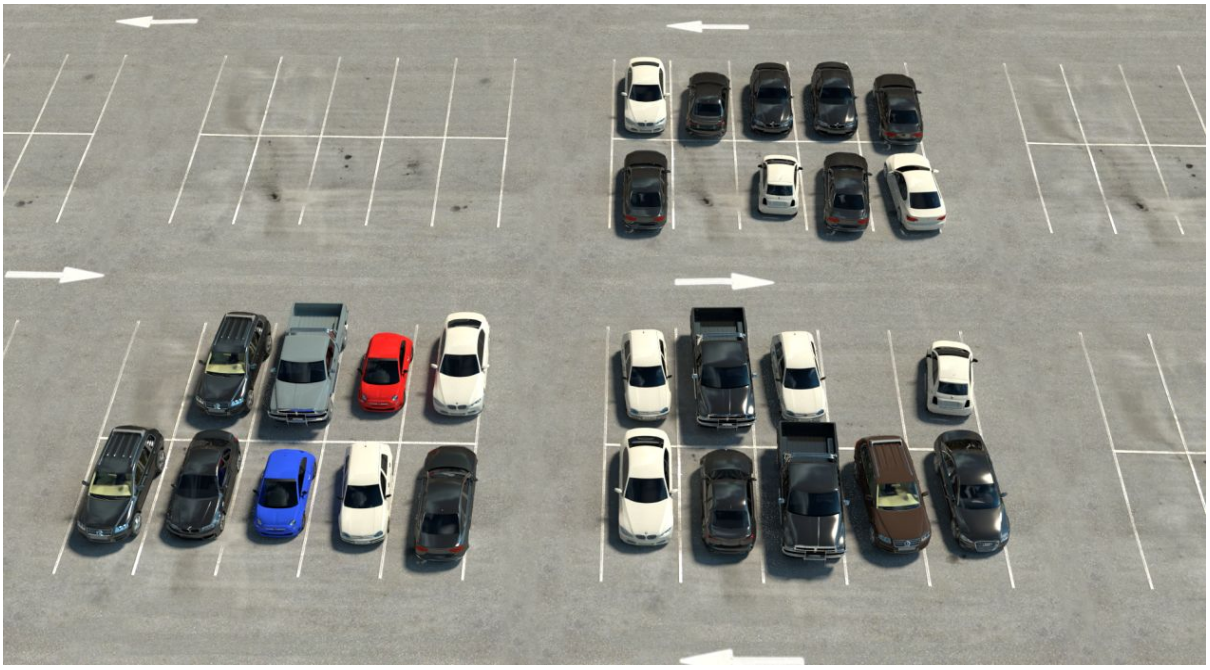
根据观察，最终参数为：

bpy.data.objects['Camera'].location = [-15,-45, 42] # x y z

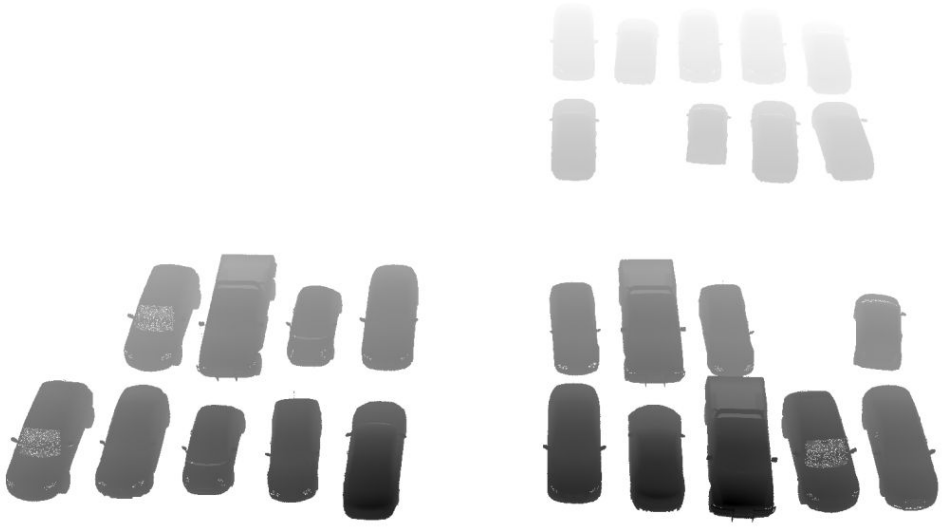
bpy.data.objects['Camera'].rotation_euler=[math.radians(45), math.radians(0), math.radians(0)]

效果如下图所示：

1.renderlayer 图



2.depthlayer 图



注：因为扩大了车库，图片生产时间加长

2.深度图片产生黑白图片，背景为黑白，车辆为全黑

作用：利用黑白图片计算出每辆车的位置信息

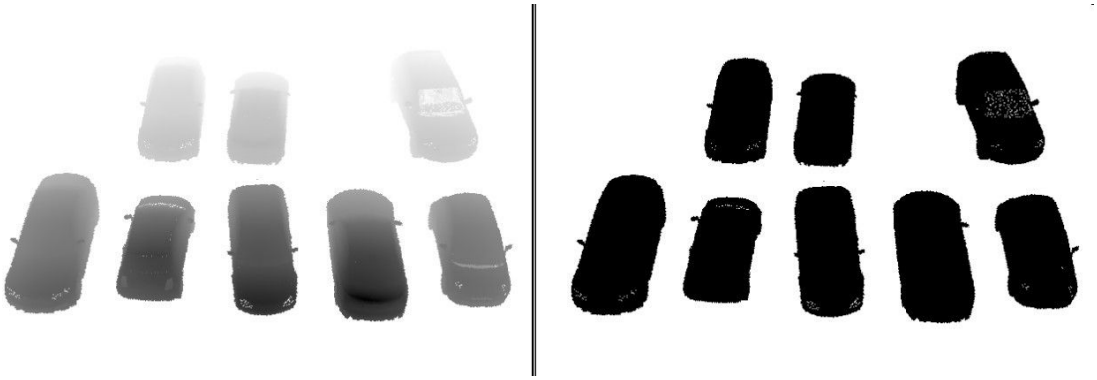
实现：使用 opencv 进行图片像素点读取，然后判断写入新的像素到新的图像中

主要实现代码(矩阵操作大大加快速度)：

```
img=cv2.imread(f_path)
img[np.where((img<[255,255,255]).all(axis=2))]=[0,0,0]
newImg_name=os.path.join(file_dir2,file_names[f_index])
if sava_img(img,newImg_name):
    print 'sava the '+str(f_index+1)+' img success'

f_index=f_index+1
print 'save total '+str(f_index)+' img'
```

最终效果如下图（左边为深度图信息，右边为黑白图信息）



3.车辆位置计算

a) 深度图像　黑白图像　和原始图像切割：

i. 将车位切割出来，计算出块的大小和坐标

ii. 保存深度和原始图像的切割块，可作为 segmentation 使用

b) 黑白图像车辆位置计算：

i. 计算出车辆位置

ii. 将位置还原至原始图像中

c) 车辆位置信息校验：

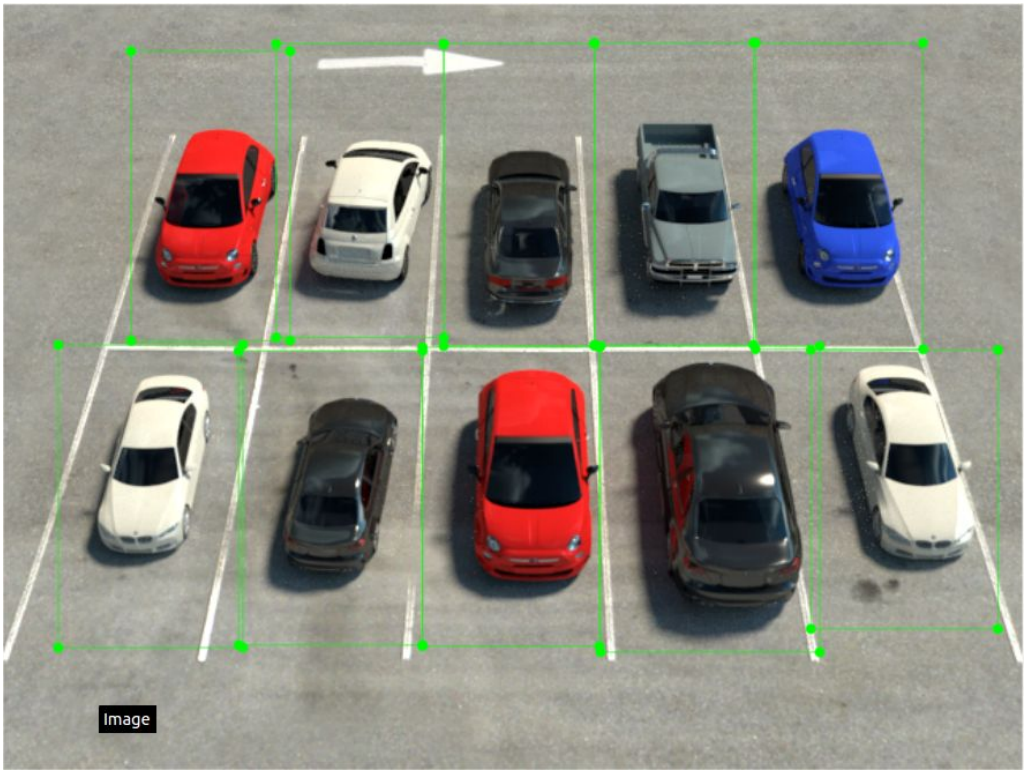
i. 原始数据中随机抽取数据组

ii. 利用已经得到的位置数据进行绘制

iii. 观察绘制结果，如有误差则需调整

a).深度图像和原始图像切割：

a） 使用 python 工具标记出车位位置，如下图所示：



得到位置信息：

```
<?xml version="1.0" ?>
<annotation>
    <folder>labelImg-master</folder>
    <filename>30</filename>
    <path>/home/gnss/Downloads/labelImg-master/30.png</path>
    <source>
        <database>Unknown</database>
    </source>
    <size>
        <width>640</width>
        <height>480</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>car</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>80</xmin>
            <ymin>29</ymin>
            <xmax>180</xmax>
            <ymax>210</ymax>
        </bndbox>
    </object>
    <object>
        <name>car</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <difficult>0</difficult>
        <bndbox>
```

进行数据转换（保存 annotation.txt）：

解析 xml 文件并将信息存入到 txt 中：

主要代码如下：

```
domTree=xml.dom.minidom.parse(fileName) #parse xml file
root=domTree.documentElement #get xml tree root
imgName=root.getElementsByTagName('filename')[0].childNodes[0].data #get img file Name
f_annotation.write(imgName+' ')
objects=root.getElementsByTagName('object')
for object in objects:
    objectName=object.getElementsByTagName('name')[0].childNodes[0].data #get img object Name
    bndBox=object.getElementsByTagName('bndbox')[0]
    xMin=bndBox.getElementsByTagName('xmin')[0].childNodes[0].data
    yMin=bndBox.getElementsByTagName('ymin')[0].childNodes[0].data
    xMax=bndBox.getElementsByTagName('xmax')[0].childNodes[0].data
    yMax=bndBox.getElementsByTagName('ymax')[0].childNodes[0].data
    print objectName+' '+xMin+' '+yMin+' '+xMax+' '+yMax
    f_annotation.write(objectName+' '+xMin+' '+yMin+' '+xMax+' '+yMax+' ')
f_annotation.write('\n')
```

最终结果贴图：

```
30.png car 80 29 180 210 car 171 24 276 208 car 276
24 371 214 car 371 24 471 213 car 472 24 577 216
car 147 217 263 401 car 263 215 374 402 car 507 216
624 391 car 375 214 512 406 car 34 213 150 403
```

进行原始图像和深度信息图像的切割

实现：使用 opencv 读取对应块的坐标，并写入新的图像中，保存图像

主要代码：

```
img_cop=img_test.copy()

#cv2.SetImageROI(img_test,(80, 29, 180, 210))
crop_img=img_cop[80:180,29:210]
cv2.imshow('test',crop_img)
```

结果贴图：

b)黑白图像车辆位置计算：

实现：计算每张图片中的车辆位置

主要代码：

```
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
    for y in range(0, zbuf_h):
        if zbuf_p[y * zbuf_w + x] < 65535:
            bounds[1] = x + 1
            flag = True
            break
    if flag == True:
        break
```

实验结果贴图：

```
2.png 45 55 90 71
```