工作记录

今天的工作安排主要有 5 点：

1.测试车辆位置设置的工作（昨天忘记测试了）

2.编写代码进行停车场打印测试

3.尝试加快打印速度的方法

a) 打印过程中只 render 要打印的数据，其他的不处理

b) 尝试变小模型，将模型中一些不必要的数据删除

c) 尝试单辆车在不同位置的的图，然后进行相关的图片拼接

4.查阅相关人群密度计算的方法和数据

5.查阅停车位检测的方法

1.测试车辆位置测试工作

a) 使用之前编写的函数即可实现

i. 添加代码：

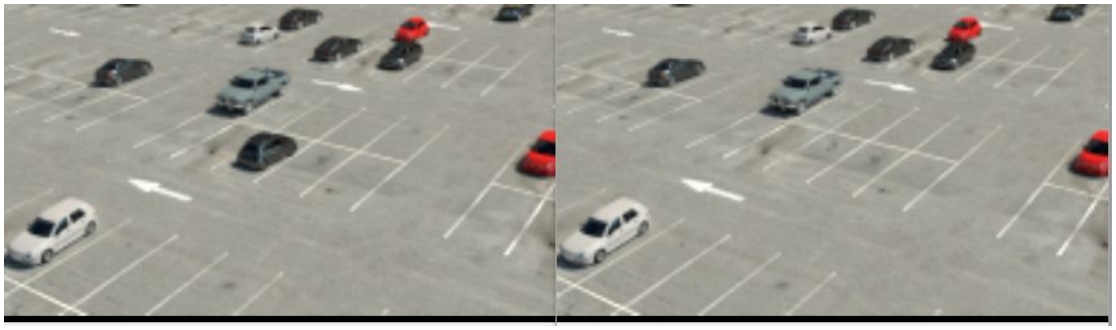
```
choose_car_position_byname('FIAT500.002',-32.02,-23.52)
```

函数实现：

```
def choose_car_position_byname(name, posi_x,posi_y):
        bpy.data.objects[name].hide_render = False;
        vec = mathutils.Vector((posi_x,posi_y, 0.3))
        inv = bpy.data.objects[name].matrix_world.copy()
        inv.invert()
        # vec aligned to local axis
        vec_rot = vec * inv
        bpy.data.objects[name].location = vec_rot
        bpy.data.objects[name].show_bounds = True
        return bpy.data.objects[name]
```

ii. 测试结果图

设置前 设置后



2.编写代码进行停车场打印测试

a) 需求：

i. 多种停车位

1. 2 排左右的停车位，10 辆车左右
2. 2 排停车位以上，30 辆车左右
3. 其他

ii. 车型比例颜色光照等，随机安排

b) 具体实现：

i. 建立车辆列表（不同车型不同颜色）

ii. 建立位置列表（车位的位置）

iii. 建立光照信息列表

iv. 设置合适的摄像头角度

v. 创建多个随机函数（随机车辆　随机光照 车位空缺随机）

vi. 调用打印测试

注（实验分析）：

1.最终效果展示：

图片可以看出位置有些漂移，实际测试打印中，将所有车辆打印在一个位 置时，车辆均在此车位的附近而并不居中，此前获取车位位置的方法其 实并不合理（使用一辆车来实验获取各个车位位置和相对关系，最终推 算得到了计算方法，但是这种方法并没有在虽有车辆上进行实际测试， 由于车型的不同，车辆在车位中的数据也是不同的，会有一定的偏移，这 个需要后面做实验来进行测试，修正误差）

2.车辆列表和位置列表光照信息列表列表

a) 车辆列表直接使用模型的车辆列表，由于车辆的颜色无法在代码中直接测试，所以只能在模型中进行设置，本次共设计了 25 辆车模型，实际使用中采用了随机的方式，从车辆中随机抽取 20 辆作为样本，具体实现如下：

i. 建立车辆模型列表

```
Car_classes=['AudiA8','AudiA8.001','AudiA8.002','AudiA8.003','AudiA8.004','BMW335i','BMW335i.001',
             'BMWM1','BMWM1.001','BMWM1.002','DodgeRamPickup','DodgeRamPickup.001','DodgeRamPickup.002',
             'FIAT500','FIAT500.001','FIAT500.002','FIAT500.003','FIAT500.004','VWGolfMK4.001','VWGolfMK4',
             'VWTouareg','VWTouareg.001','VWTouareg.002'
            ]
```

ii. 设计随机函数

```
#get random CarClasses
def getRandomCarClasses(Car_print_team,Car_print_rows,Car_print_ranks):
    return random.sample(Car_classes,Car_print_rows*Car_print_ranks*Car_print_team)
```

iii. 随机获取样本

```
car_classes_print_list=getRandomCarClasses(Car_print_team,Car_print_rows,Car_print_ranks)
```

iv. 位置列表

```
#define car space top position
Car_Space_Position_Top=[
            [[62,29.23],[49.83,29.23],[32.02,29.23],[14.2,29.23],[.67,29.23],[1.3,29.23],[9.04,29.23],[7.1,29.23]],
            [[62,11.55],[49.83,11.55],[32.02,11.55],[14.2,11.55],[.67,11.55],[1.3,11.55] ,[9.04,11.55],[7.1,11.55]],
            [[62,-5.69],[49.83,-5.69],[32.02,-5.69] ,[14.2,-5.69],[.67,-5.69],[1.3,-5.69],[9.04,-5.69],[7.1,-5.69]],
            [[62,-23.52],[49.83,-23.52],[32.02,-23.52],[14.2,-23.52],[.67,-23.52],[1.3,-23.52],[9.04,-23.52] ,[7.1,-23.52]]
            ]
```

注：所有位置信息均为每组停车位左上角的停车位位置信息，除列表第一列和最后一列为特殊的停车组外其他停车组均为 5 列 2 行 10 个车位，实际使用中，通过车位组顶角的位置来计算车位组中其他车位的位置

b) 创建多个随机函数

i. 获取车位组车位随机函数

```
#get random carSpace satrt position
def getCarSpace_Start_posi(Car_Space_Start_num):
    return random.sample(Car_space_start_posi,Car_Space_Start_num)
```

注：从摄像头可见的 4 组停车位中获取 2 组的位置信息

ii. 车位空缺随机函数

```
def setCarSpace_random_empty():
    isEmpty=random.randint(0,1)
    return isEmpty
```

注：后面引入其他概率值进行车位空缺的概率设置，目前是 0.5

iii. 车辆位置随机放置

```
#define get sapce position
def getCarSpace_Position(car_sapce_posi_top_row,car_sapce_posi_top_rank,car_num):
    if car_num<5:
        Car_print_posi=[Car_Space_Position_Top[car_sapce_posi_top_row][car_sapce_posi_top_rank][0],Car_Space_Position_Top[ca
        print(str(Car_print_posi)+'****')
        return Car_print_posi
    else:
        Car_print_posi=[Car_Space_Position_Top[car_sapce_posi_top_row][car_sapce_posi_top_rank][0]+Car_Space_up_apart_down,Ca
        print(str(Car_print_posi)+'****')
        return Car_print_posi
```

iv. 主要实现代码：

```
190     hide_all_cars(Car_classes)
191     car_classes_print_list=getRandomCarClasses(Car_print_team,Car_print_rows,Car_print_ranks)
192     car_space_posi_list=getCarSpace_Start_posi(Car_print_team)
193
194     print(str(car_classes_print_list)+'\n'+str(car_space_posi_list)+'\n')
195     print(str(car_space_posi_list)+'  '+str(Car_print_rows)+'  '+str(Car_print_ranks))
196     for car_space_posi_list_len in range(0,len(car_space_posi_list)):
197         for car_posi_print_row_len in range(0,Car_print_rows):
198             for car_posi_print_rank_len in range(0,Car_print_ranks):
199
200                 print('is is '+str(Car_print_rows*Car_print_ranks+car_posi_print_row_len*Car_print_ranks+car_posi_print_rank
201                 car_name=car_classes_print_list[Car_print_rows*Car_print_ranks+car_posi_print_row_len*Car_print_ranks+car_po
202                 car_print_posi=getCarSpace_Position(car_space_posi_list[car_space_posi_list_len][0],car_space_posi_list[car_
203                 #print(str(car_space_posi_list_len)+'  '+str(car_posi_print_row_len)+'  '+str(car_posi_print_rank_len)+'\t'+
204                 posi_x=car_print_posi[0]
205                 posi_y=car_print_posi[1]
206                 choose_car_position_byname(car_name, posi_x,posi_y)
207
208     bpy.data.scenes['Scene'].layers=[True,True,False,False,False,False,False,False,False,False,False,True,True,False,False,False,Fa
209     bpy.context.scene.render.filepath = filepath1
210     bpy.ops.render.render(write_still = True)
211     #print only car layer
212     bpy.data.scenes['Scene'].layers=[False,False,False,False,False,False,False,False,False,False,False,True,False,False,False,Fals
213     bpy.context.scene.render.filepath = filepath2
214     bpy.ops.render.render(write_still = True)
215
216     annot_file.flush()
```

3.尝试加速打印方法

a)打印过程中只 render 要打印的数据，其他的不处理

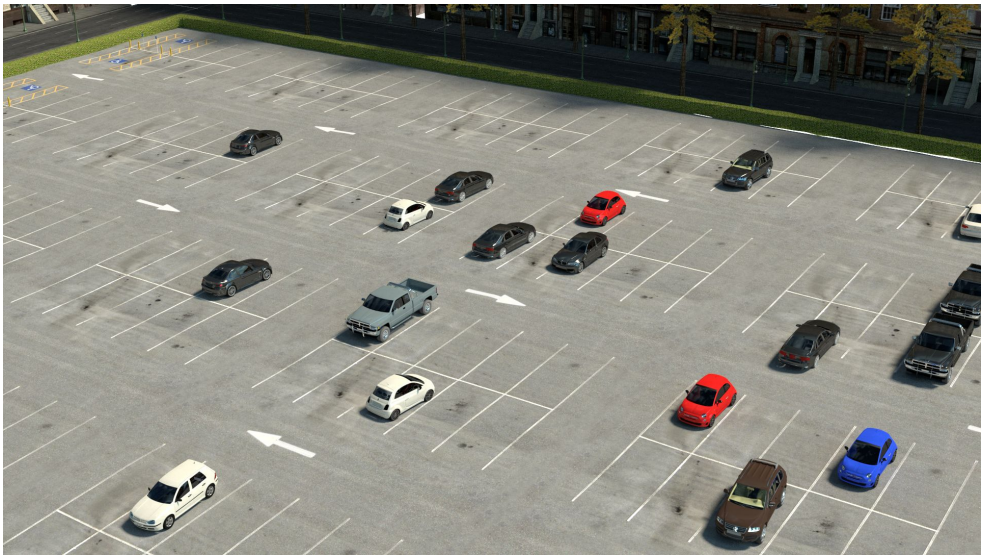
实际效果：可以加快处理速度，具体速度暂时未进行计算

b)尝试变小模型，将模型中一些不必要的数据删除

暂未尝试

c)尝试单辆车在不同位置的的图，然后进行相关的图片拼接人群密度计算方法

进行图片剪切和黏贴，可以实现



4.人群密度检测方法

A.固定场景

B.非固定场景

5.车位检测方法