深蓝学院
shenlanxueyuan.com

3D点云第一章作业分享

主讲人 张点垒

纲要

深蓝学院
shenlanxueyuan.com

- **第一部分：概述**
- 第二部分：方法

  ****

  ****
- 第三部分：问题与挑战

第一题

●Perform PCA for the 40 objects, visualize it.

●思路：从PCA原理出发，计算点云矩阵的特征值/SVD。

●注：需要注意点云数据矩阵的维度，计算出的主方向应该是3维，而不是N维。

第一题

深蓝学院
shenlanxueyuan.com

●代码实现：PCA

```
def PCA(data, correlation=False, sort=True):
    # 作业1
    # 屏蔽开始
        eigenvalues, eigenvectors = np.linalg.eig(data.T.dot(data)) # 计算矩阵X*X^T的特征值和特征向量
        # 屏蔽结束

        if sort:
         sort = eigenvalues.argsort()[::-1]
         eigenvalues = eigenvalues[sort]
         eigenvectors = eigenvectors[:, sort]

    return eigenvalues, eigenvectors
```

第一题

- 代码实现：3D可视化
- 可以创建一个o3d.geometry.LineSet()对象，并在空间中沿主方向定义3个点，实现在3维视图中主方向的可视化。

```
#三维主方向可视化
pca_vector=o3d.geometry.LineSet() #创建一个线集
lines = [[0, 1], [1, 2]]  #定义两条线(实际上是共线的)
colors = [[0, 0, 1] for i in range(len(lines))] #定义每条线的颜色
vector_points = np.array([origin_point-point_cloud_vector, origin_point, origin_point+point_cloud_vector], dtype=np.float32) #录入三个点，其均在主方向上
pca_vector.lines = o3d.utility.Vector2iVector(lines)
pca_vector.colors = o3d.utility.Vector3dVector(colors)
pca_vector.points = o3d.utility.Vector3dVector(vector_points)

vis = o3d.visualization.Visualizer()
vis.create_window(window_name='Principal direction', width=1920, height=1080, left=10, top=10, visible=True) #定义画图窗口
vis.get_render_option().point_size = 2  # 设置点的大小
vis.add_geometry(point_cloud_o3d)
vis.add_geometry(pca_vector)
vis.create_window()
vis.run() #绘图
vis.destroy_window() #关闭窗口
```

第一题

深蓝学院
shenlanxueyuan.com

- 代码实现：2D投影可视化

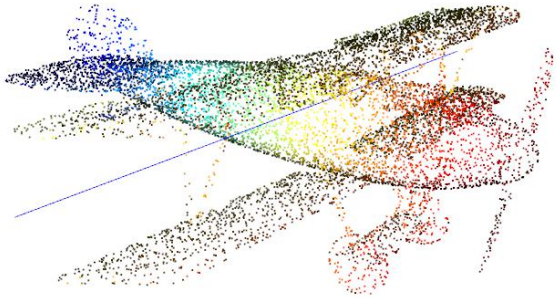
- 根据投影原理，直接将点云数据与主方向和次主方向点积可以得到投影值。增加一个空维度，使其变成三维空间中同一平面的点，便于可视化。

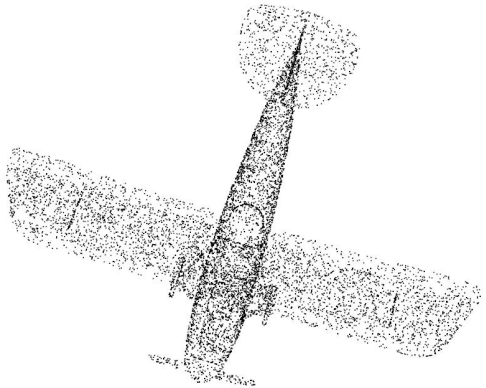
```
#在主方向和次主方向做投影，形成二维可视化图片
proj_vector=v[:, 0:2] #取前两个向量
proj_point=np.matmul(points,proj_vector)
proj_point=np.concatenate((proj_point,np.zeros([point_num,1])),axis=1) #添加一个统一的Z轴数据，形成3D点云，方便可视化
point_cloud_proj_pca = o3d.geometry.PointCloud(o3d.utility.Vector3dVector(proj_point))
vis_pca = o3d.visualization.Visualizer()
vis_pca.create_window(window_name='PCA projection', width=1920, height=1080, left=10, top=10,
                      visible=True)  # 定义画图窗口
vis_pca.get_render_option().point_size = 2  # 设置点的大小
vis_pca.add_geometry(point_cloud_proj_pca)
vis_pca.create_window()
vis_pca.run()  # 绘图
vis_pca.destroy_window()  # 关闭窗口
```

第一题

深蓝学院
shenlanxueyuan.com

●可视化结果





第一题

深蓝学院
shenlanxueyuan.com

- Perform surface normal estimation for each point of each object, visualize it.

- 思路：第一章还没有讲点云的邻域搜索算法，可以直接使用open3d给出的方法o3d.geometry.KDTreeSearchParamKNN()。

第一题

深蓝学院
shenlanxueyuan.com

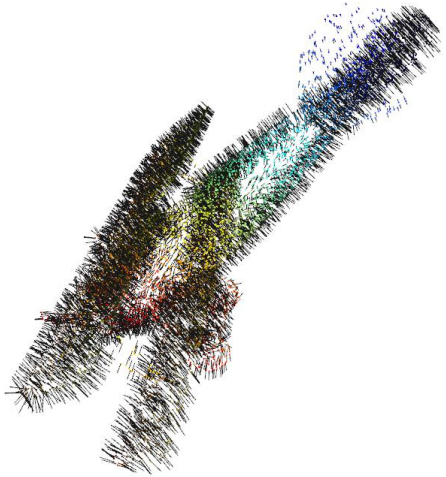
●代码实现：法方向可视化

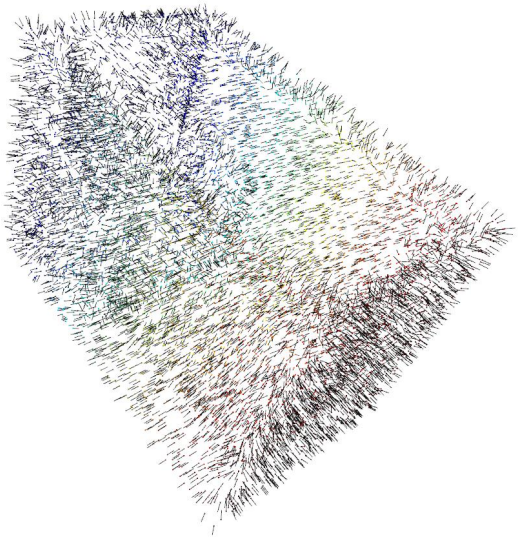
```
# 作业2
# 屏蔽开始
point_cloud_o3d.estimate_normals(
    search_param=o3d.geometry.KDTreeSearchParamKNN(knn=20)) # 计算法线，只考虑邻域内的20个点
normals = np.asarray(point_cloud_o3d.normals)
# 由于最近邻搜索是第二章的内容，所以此处允许直接调用open3d中的函数

# 屏蔽结束
normals = np.array(normals, dtype=np.float64)
#
point_cloud_o3d.normals = o3d.utility.Vector3dVector(normals) # 输入法线数据
vis = o3d.visualization.Visualizer()
vis.create_window(window_name='Normal vectors', width=1920, height=1080, left=10, top=10, visible=True)
vis.get_render_option().point_size = 2  # 设置点的大小
vis.get_render_option().point_show_normal=True #显示法线
vis.add_geometry(point_cloud_o3d)
vis.create_window()
vis.run()  #可视化
vis.destroy_window()  #关闭窗口
```

第一题

深蓝学院
shenlanxueyuan.com

●可视化结果





第二题

- Downsample each object using voxel grid downsampling(exact, both centroid & random). Visualize the results.

- **思路：根据体素降采样原理：1）划分网格；2）创建容器（hash表）；3）向容器中压入/弹出点；4）弹出容器中剩余的所有点。弹出的方法可以是centroid或random。**

第二题

深蓝学院 shenlanxueyuan.com

●代码实现：体素降采样

```
def voxel_filter(point_cloud, leaf_size, filter_type='random'):
    filtered_points = []
    try:
        assert filter_type=='random' or filter_type=='mean'
    except:
        print('Wrong filter_type input!')
        raise
    # 作业3
    # 屏蔽开始
        grid_num=[8,3,8]  #定义网格数量
        max_num_of_hash_table=leaf_size  #hash table总大小
        min_coord=np.min(point_cloud,axis=0) #读取数据中最大坐标
        max_coord=np.max(point_cloud,axis=0) #读取数据中最小坐标
        grid_size=(max_coord-min_coord)/grid_num #计算网格大小
        hash_table_point={} #创建一个字典作为hash表来存储当前hash值对应的点
        hash_table_index={} #创建一个字典作为hash表来存储当前hash值对应的index
    for point in point_cloud:
        grid_coord=np.floor((point-min_coord)/grid_size) #计算网格坐标
              index=grid_coord[0]+grid_coord[1]*grid_num[0]+grid_coord[2]*grid_num[0]*grid_num[1] #计算index
        hash_index=np.mod(index,max_num_of_hash_table) #转换为hash值
```

第二题

深蓝学院 shenlanxueyuan.com

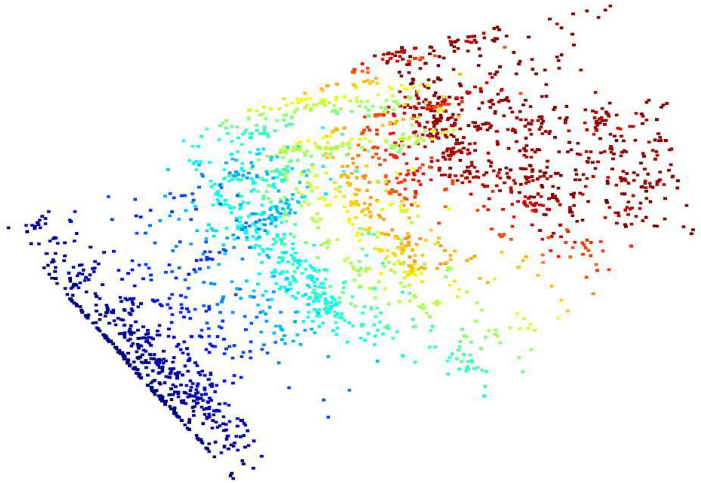
●代码实现：体素降采样

```
    if not hash_index in hash_table_index:  #若hash值不存在，则创建该值并存储该点
            hash_table_index[hash_index]=index
        hash_table_point[hash_index]=[point]
    elif index==hash_table_index[hash_index]: #如果hash值存在并且序号不冲突，则存储该点
            hash_table_point[hash_index].append(point)
    else:   #如果hash存在冲突，则清空该hash值已有的容器
            #random sample
        if filter_type=='random':  #随机采样
                filtered_points.append(random.choice(hash_table_point[hash_index]))
        #average sample
        elif filter_type=='mean':  #centroid采样
                filtered_points.append(list(np.mean(np.array(hash_table_point[hash_index]),axis=0)))
        hash_table_index[hash_index] = index
        hash_table_point[hash_index] = [point]
for hash_index in hash_table_point: #将所有hash表中剩余的数据输出
       if filter_type == 'random':
       filtered_points.append(random.choice(hash_table_point[hash_index]))
    # average sample
    elif filter_type == 'mean':
       filtered_points.append(list(np.mean(np.array(hash_table_point[hash_index]), axis=0)))
hash_table_index.clear()  #清空hash table 占用的内存
hash_table_point.clear()
# 把点云格式改成array，并对外返回
filtered_points = np.array(filtered_points, dtype=np.float64)
return filtered_points
```

第二题

深蓝学院
shenlanxueyuan.com

- 可视化结果：体素降采样



第三题

- Perform depth upsampling / completion for the validation dataset

- 思路：根据Bilateral Filter原理，对深度图进行滤波，可以使用RGB/灰度图信息作为额外的权重。

- 由于深度图是稀疏的，因此在权重分配上需要单独做一些处理。

第三题

$$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in \mathcal{S}} G_{\sigma_s}(\|\mathbf{p} - \mathbf{q}\|)\, G_{\sigma_r}(I_{\mathbf{p}} - I_{\mathbf{q}})\, I_{\mathbf{q}}$$

$$W_{\mathbf{p}} = \sum_{\mathbf{q} \in \mathcal{S}} G_{\sigma_s}(\|\mathbf{p} - \mathbf{q}\|)\, G_{\sigma_r}(I_{\mathbf{p}} - I_{\mathbf{q}})$$

- 使用RBG/灰度图的信息作为额外的权重。
- 将等式最右端的加权对象Iq（图像灰度值）更换为该点的深度值。
- 稀疏性：没有深度的点（深度为0/-1）不应该影响周围点的加权求和。

第三题

深蓝学院
shenlanxueyuan.com

- 代码实现：高斯函数计算邻域点权重

```
def Gaussian_fun(x,sigma):
    value= 1/(2*np.pi*sigma)*np.exp(-np.square(x)/(2*np.square(sigma)))
    return value
```

第三题

深蓝学院 shenlanxueyuan.com

- 代码实现：高斯函数计算邻域点权重
- 对于没有深度的点，将其权重置零

```
def Bilateral_Filter_new (image_raw,depth_raw,sigma_s,sigma_r,nr):
    #image_raw RGB图像
      #depth_raw Depth图像
      #sigma_s 颜色权重sigma 标量
      #sigma_r 深度权重sigma 标量
      #nr 邻域范围  标量  奇数
      assert np.mod(nr,2)
    depth_raw=np.asarray(depth_raw)
    image_raw=np.asarray(image_raw)
    image_raw=(image_raw[:,:,0]+image_raw[:,:,1]+image_raw[:,:,2])/3    #转换为灰度图
       img_size = depth_raw.shape
    depth_filt = np.zeros(img_size,dtype=np.float32)
    temp_index=np.int((nr-1)/2)
    Gaussian_depth_dis = np.zeros([nr, nr])
    for ii in range(nr):
        for jj in range(nr):
            Gaussian_depth_dis[ii,jj]=ii+jj-2*temp_index
    Gaussian_depth_weight=Gaussian_fun( Gaussian_depth_dis,sigma_r)
```

```
for ii in range(img_size[0]):
    for jj in range(img_size[1]):
        if ii - temp_index < 0 or ii + temp_index >= img_size[0]:  #忽略边缘点的滤波
                continue
        if jj - temp_index < 0 or jj + temp_index >= img_size[1]:
            continue
        pixel_grey = image_raw[ii, jj]
        grey_nb = image_raw[ii - temp_index:ii + temp_index + 1, jj - temp_index:jj + temp_index + 1]
        grey_weight = Gaussian_fun(grey_nb - pixel_grey, sigma_s)
        grey_weight=grey_weight/np.sum(grey_weight)
        depth_nb=depth_raw[ii-temp_index:ii+temp_index+1,jj-temp_index:jj+temp_index+1]
        depth_nb_weight=copy.copy(Gaussian_depth_weight)
        depth_nb_weight[depth_nb==-1] = 0;
        if np.sum(depth_nb_weight)==0:
            continue
        depth_nb_weight=depth_nb_weight/np.sum(depth_nb_weight)
        w=depth_nb_weight*grey_weight
        w=w/np.sum(w)
        depth_filt[ii,jj]=np.sum(w*depth_nb)
return depth_filt
```

第三题

- 代码实现：读取深度图进行双边滤波并保存结果。
- 灰度值的sigma取0.2
- 深度值的sigma取5
- 滤波的kernel大小取11

```
def depth_predict_and_save(file_name):
    #为了方便构建并行，这些固定参数没有作为函数的输入
        KITTI_depth_path = 'E:/DataSet/KITTI depth/'
    depth_path = 'val_selection_cropped/velodyne_raw/'
    image_path = 'val_selection_cropped/image/'
    gt_path = 'val_selection_cropped/groundtruth_depth/'
    save_path = './KITTI_depth_result_with_sigma_5/'
    depth_file_name = file_name
    image_file_name = file_name.replace('velodyne_raw', 'image')
    gt_file_name = file_name.replace('velodyne_raw', 'groundtruth_depth')
    image_file_path = os.path.join(KITTI_depth_path, image_path, image_file_name)
    depth_file_path = os.path.join(KITTI_depth_path, depth_path, depth_file_name)
    gt_file_path = os.path.join(KITTI_depth_path, gt_path, gt_file_name)
    # 转换为o3d的图像和深度图格式
    depth_raw = depth_read(depth_file_path).astype(np.float32)
    color_raw = plt.imread(image_file_path).astype(np.float32)
    gt_raw = depth_read(gt_file_path).astype(np.float32)
    depth_bilateral = Bilateral_Filter_new(color_raw, depth_raw, 0.2, 5, 11) # 对深度进行双边滤波
    depth_bilateral[depth_bilateral < 0] = 0;
    depth_raw[depth_raw < 0] = 0;
    gt_raw[gt_raw < 0] = 0;
    save_file_path = os.path.join(save_path, file_name)
    depth_save_data = Image.fromarray(depth_bilateral)
    depth_save_data.save(save_file_path)
    print(file_name, 'Processing finished')
```

第三题

深蓝学院
shenlanxueyuan.com

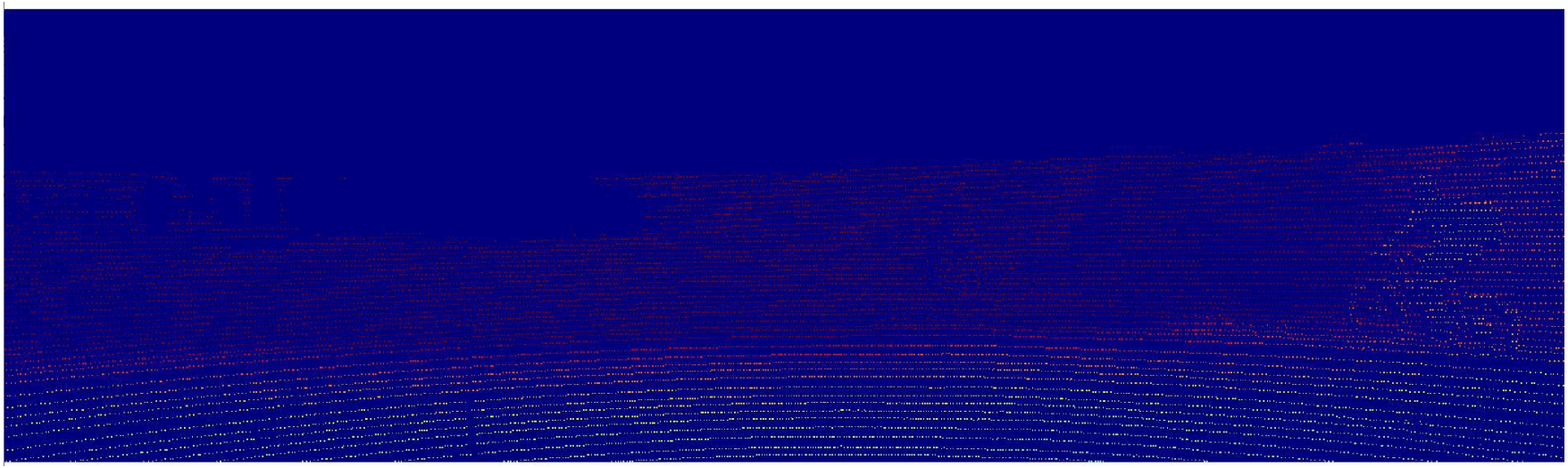
- 代码实现：主函数，对数据并行处理。

```
def main():
    KITTI_depth_path='E:/DataSet/KITTI depth/'
    depth_path='val_selection_cropped/velodyne_raw/'
    file_path=os.path.join(KITTI_depth_path,depth_path)
    file_names=os.listdir(file_path)
    pool = ThreadPool()
    pool.map(depth_predict_and_save, file_names)
    pool.close()
    pool.join()
```

第三题

深蓝学院
shenlanxueyuan.com

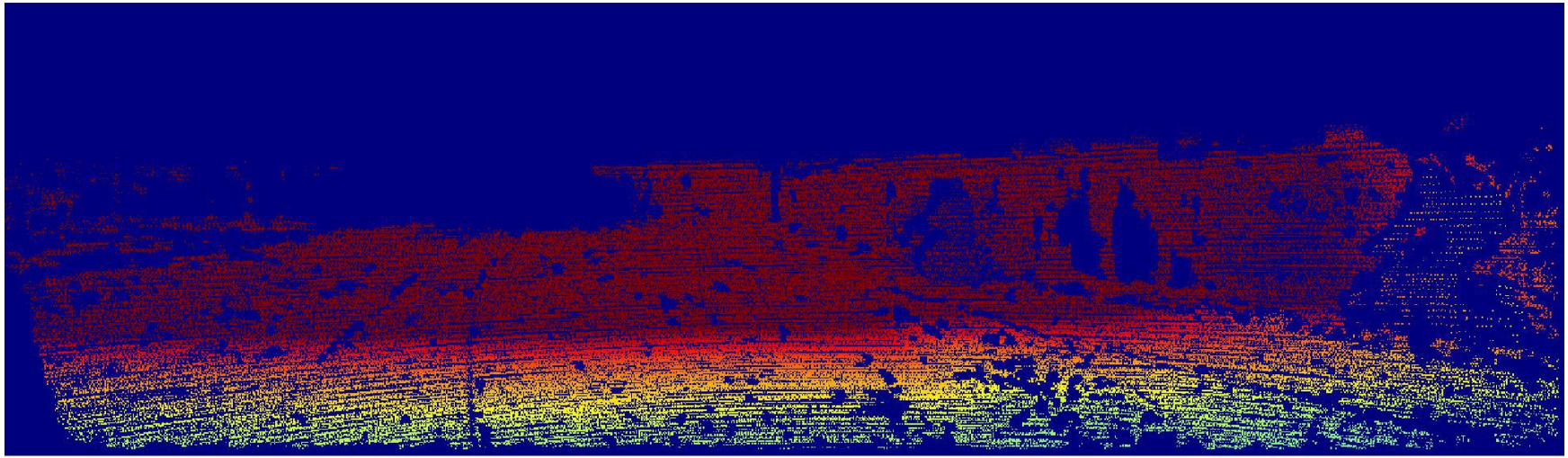
●可视化结果：输入深度图



第三题

深蓝学院
shenlanxueyuan.com

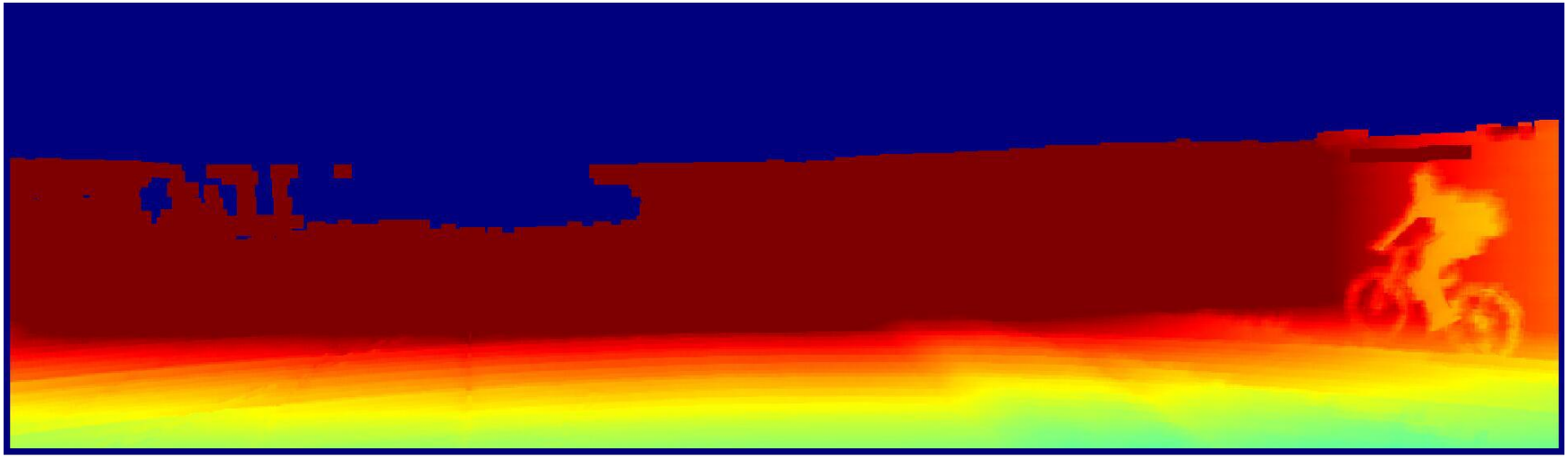
●可视化结果：深度图GT



第三题

深蓝学院
shenlanxueyuan.com

●可视化结果：深度图滤波结果



第三题

深蓝学院
shenlanxueyuan.com

●量化结果：From KITTI depth Evaluation

mean mae: 0.528272
min mae: 0.209994
max mae: 1.850485
mean rmse: 1.900804
min rmse: 0.612379
max rmse: 7.132728

总结与经验

- 第一次作业的较多时间都花在了熟悉相关的库，数据的使用以及相关的数据格式等问题上，但详细了解一下相关的函数以及对应的数据格式可以避免以后再次踩坑。
- Open3d的Visualizer好像在显示深度图的时候有距离限制，过大的距离显示不出来。可以直接使用draw_geometries()方法绘制完整的深度图。
- 双边滤波由于进行类似卷积的操作（变化权重的卷积），这种计算如果单纯按照公式来写程序，运行速度较慢。这也是我在主函数中使用并行的原因，即使如此，速度还是挺慢。
- 双边滤波当使用共享距离权重时，要注意对共享的距离权重变量进行深拷贝，否则调整权重时会影响到共享距离权重的值，导致计算错误。变量间的引用关系也应该在编程时格外注意。

在线问答

深蓝学院
shenlanxueyuan.com

Q&A

深蓝学院
shenlanxueyuan.com

感谢各位聆听！

Thanks for Listening