深蓝学院
shenlanxueyuan.com

三维点云处理第一次作业

主讲人 Lorenzo

第一题

- 对点云数据进行PCA分析

```python
def PCA(data, correlation=False, sort=True):
    # 作业1
    # 屏蔽开始

    data = data.transpose()
    data = data - data.mean(axis = 1, keepdims=True)

    # data: 3*m matrix
    # data_T: m*3 matrix
    data_T = data.transpose()

    # 3*3 matrix
    H = np.matmul(data, data_T)

    # SVD
    eigenvectors, eigenvalues, _ = np.linalg.svd(H, full_matrices=True)

    # 屏蔽结束

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

    return eigenvalues, eigenvectors
```

第一题

深蓝学院
shenlanxueyuan.com

- 将点云投影到二维平面

```python
# 从点云中获取点，只对点进行处理
points = point_cloud_pynt.points
points = points.to_numpy()

# 用PCA分析点云主方向
w, v = PCA(points)
point_cloud_vector = v[:, 0] #点云主方向对应的向量
print('the main orientation of this pointcloud is: ', point_cloud_vector)
# TODO: 此处只显示了点云，还没有显示PCA
# select the first two principle component
projected_points = np.dot(points, v[:, :2])
projected_points = np.hstack([projected_points,
    np.zeros((projected_points.shape[0],1))])

projected_point_cloud_o3d = o3d.geometry.PointCloud()
projected_point_cloud_o3d.points = o3d.utility.Vector3dVector(projected_points)

o3d.visualization.draw_geometries([projected_point_cloud_o3d])
```

第一题

●对点云数据进行PCA分析并将点云投影到二维平面

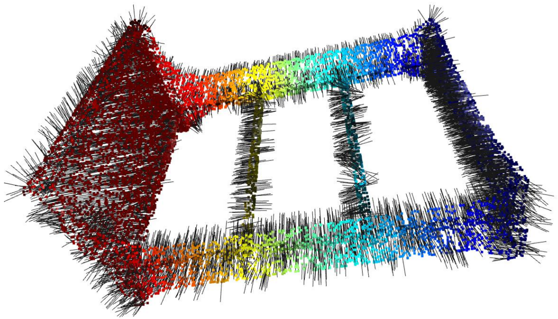
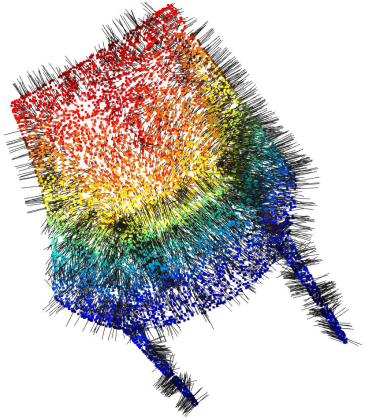
第一题

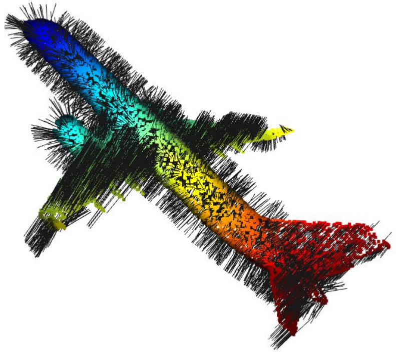
- 利用PCA分析进行法向量估计
  - 可用search_radius_vector_3d或search_knn_vector_3d
  - 如果使用search_radius_vector_3d，需挑选一个合理的搜索半径，如：点云跨度的5%

```python
# 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
# the range of the cloud in three dimensions
cloud_range = points.max(axis=0)-points.min(axis=0)
# radius: set to 5% of the cloud's max range
radius = cloud_range.max() * 0.05
for point in point_cloud_o3d.points:
    cnt, idxs, dists = pcd_tree.search_radius_vector_3d(point, radius)
    # v: 3*3 matrix
    w, v = PCA(points[idxs])
    # v[:,-1]: 3*1 matrix
    normal = v[:,-1]
    normals.append(normal)
```

第一题

- 利用PCA分析进行法向量估计



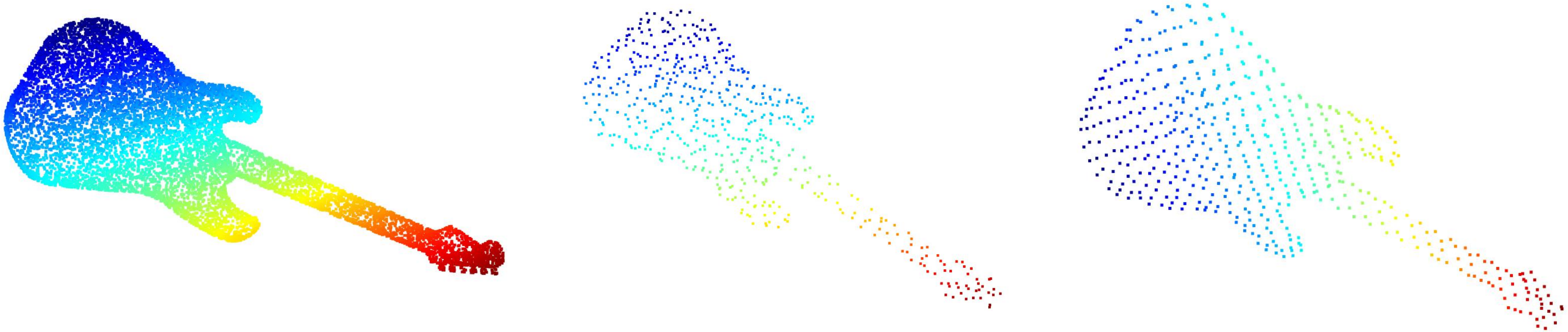
第二题

- 体素式滤波
  - 利用numpy的broadcasting操作，减少for循环的使用
  - indices记录点在x, y, z三个方向上的序号
  - h_indices记录点属于第几个voxel grid

```python
def voxel_filter(point_cloud, leaf_size, use_centroid = True):
    filtered_points = []
    # 作业3
    # 屏蔽开始

    # point_cloud: m*3 pandas DataFrame
    point_cloud = np.array(point_cloud)
    uppers = np.max(point_cloud, axis=0)
    lowers = np.min(point_cloud, axis=0)

    dims = np.ceil((uppers - lowers)/leaf_size)

    # indices: m*3
    indices = (point_cloud - lowers)//leaf_size
    # h_indices: m
    h_indices = indices[:,0] + indices[:,1]*dims[0] + indices[:,2]*dims[0]*dims[1]

    for h_index in np.unique(h_indices):
        points = point_cloud[h_indices==h_index]
        if use_centroid:
            # use centroid for each voxel
            filtered_points.append(np.mean(points, axis = 0))
        else:
            # select point in each voxel randomly
            filtered_points.append(random.choice(points))

    # 屏蔽结束

    # 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

- 体素式滤波-原始点云/random/centroid



第二题

- 体素式滤波-原始点云/random/centroid

选做题

- KITTI depth completion数据集介绍
  - https://blog.csdn.net/keineahnung2345/article/details/114369593

选做题

深蓝学院 shenlanxueyuan.com

- Bilateral Filter
  - 每个像素的新intensity由它的邻域S决定
  - 邻域中每个像素的权重由以下两点决定
    - 该像素与待更新像素的距离
    - 该像素与待更新像素intensity的差值
  - 可调参数
    - 邻域大小
    - 两个高斯函数G_{σ_s}及G_{σ_r}的标准差σ_s 及σ_r

$$G_\sigma(x) = \frac{1}{2\pi\,\sigma^2}\exp\left(-\frac{x^2}{2\,\sigma^2}\right)$$

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

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

选做题

- Bilateral Filter
  - 加速方法
    - neighborhood一般为一个正方形，可以将邻域内的intensity及距离都表示成矩阵，然后使用矩阵乘法做运算
    - 预先计算距离权重的kernel
    - 预先对各种可能的intensity差值计算其通过高斯函数后的值 @金鑫

选做题

- 只对前100笔数据做评测，左侧为原始数据，右侧为做过BF后（kernel size=5，$\sigma_s = 50$，$\sigma_r = 50$）的评测结果

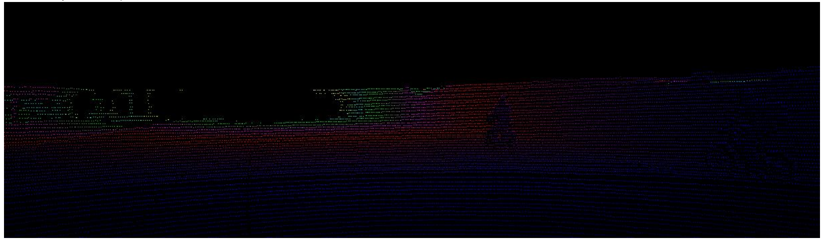
```
mean mae: 2.41848
mean rmse: 6.01417
mean inverse mae: 0.0093934
mean inverse rmse: 0.023794
mean log mae: nan
mean log rmse: -nan
mean scale invariant log: -nan
mean abs relative: 0.10948
mean squared relative: 0.0628753
```

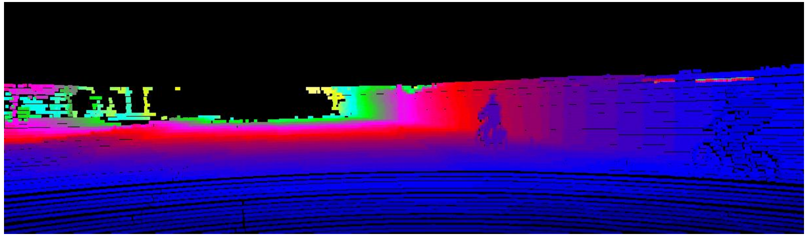
```
mean mae: 1.06399
mean rmse: 2.58188
mean inverse mae: 0.00798715
mean inverse rmse: 0.014853
mean log mae: nan
mean log rmse: -nan
mean scale invariant log: -nan
mean abs relative: 0.0749187
mean squared relative: 0.0230063
```

选做题

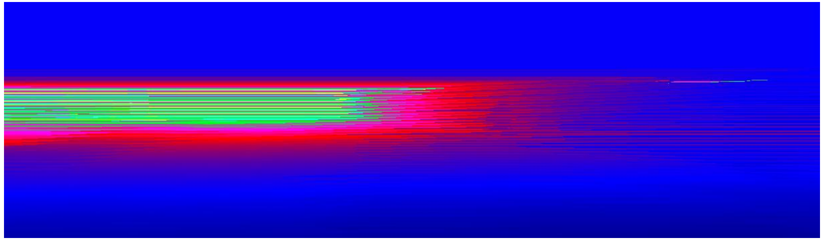
- 左：原始数据，右：BF

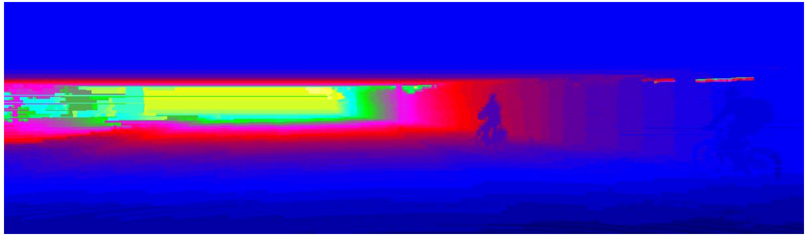
depth_orig



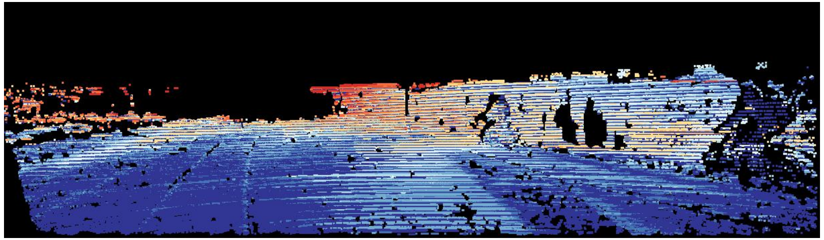


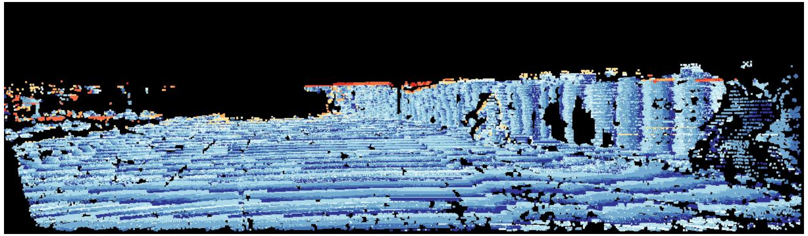
depth_ipol





errors_img





在线问答

Q&A

感谢各位聆听！

Thanks for Listening