深蓝学院
shenlanxueyuan.com

3D点云第一章作业学员分享

主讲人 Kai.

纲要

- 整体思路
- 代码分析
  - 第一题
  - 第二题

整体思路

套公式！

整体思路

- 理解公式意义
- 搞懂矩阵形状
- 进行矩阵运算

纲要

- 整体思路
- 代码分析
  - 第一题
  - 第二题

第一题

深蓝学院 shenlanxueyuan.com

```python
data = []
#name of each point cloud file
names = []
#list that stores all eigenvector matrices
U = []
#file path
path = './data/*.txt'

#load file into data ans save all names
def loadfile():
    for filename in glob(path):
        names.append(filename)
        #load data to a numpy array and ignore comma
        temp = np.genfromtxt(filename, delimiter=',', dtype=float)
        #temp = temp.reshape((-1, 3))
        data.append(temp[:, :3])

#get eigenvector matrix for X*transpose(X)
def getU(X):
    H = np.transpose(X).dot(X)
    eValues, eVectors = np.linalg.eig(H)
    idx = eValues.argsort()[::-1]
    eValues = eValues[idx]
    eVectors = eVectors[:,idx]
    return eVectors
```

40 × N × 3

40 × 3 × 3

1×1 3×N · N×3

$$\max_{z \in R^n} z^T(\underbrace{\tilde{X}\tilde{X}^T})z, \text{ s.t.:} \|z\|_2 = 1$$

1×3 · 3×3 · 3×1

ppt p28

$$(M \times N) \cdot (N \times K) = (M \times K)$$

第一题

```
def PCA():
    for pc, u, name in zip(data, U, names):
        pc_2d = pc.dot(u[:, 0:2])
        temp = np.zeros([pc_2d.shape[0], 1], dtype = float)
        pc_2d = np.append(pc_2d, temp, axis = 1)
        pc_view = o3d.geometry.PointCloud()
        pc_view.points = o3d.utility.Vector3dVector(pc_2d)
        o3d.visualization.draw_geometries([pc_view], window_name = name)
```

第一题

深蓝学院
shenlanxueyuan.com

```python
def SNE():
    for pc, name in zip(data, names):
        point_tree = spatial.cKDTree(pc)
        current_row = 0
        num_row = pc.shape[0]
        lines = []
        for p in pc:
            neighbor = point_tree.data[point_tree.query_ball_point(p, 0.05)]
            if (neighbor.shape[0] > 2):
                U_temp = getU(neighbor)
                pc = np.append(pc, np.reshape(p + 0.1 * np.transpose(U_temp[:, 2]), (1, 3)), axis = 0)
            else:
                pc = np.append(pc, np.reshape(p, (1, 3)), axis = 0)
            lines.append([current_row, current_row + num_row])
            current_row += 1
        colors = [[1, 0, 0] for i in range(len(lines))]
        pc_view = o3d.geometry.PointCloud()
        pc_view.points = o3d.utility.Vector3dVector(pc[:num_row, :])
        line_set = o3d.geometry.LineSet()
        line_set.points = o3d.utility.Vector3dVector(pc)
        line_set.lines = o3d.utility.Vector2iVector(lines)
        line_set.colors = o3d.utility.Vector3dVector(colors)
        o3d.visualization.draw_geometries([pc_view, line_set], window_name = name)
```

第一题

深蓝学院
shenlanxueyuan.com

```python
def voxel_grid(r):
    sorted_data = []
    for pc in data:
        max = np.amax(pc, axis = 0)
        max = max + r / 2
        min = np.amin(pc, axis = 0)
        min = min - r / 2
        d = (max - min) / r
        H = (pc - [min for i in range(len(pc))]) / r
        H = H.astype(int)
        h = H.dot([[1], [d[0]], [d[0] * d[1]]])
        h = h.astype(int)
        sorted_pc = np.append(pc, h, axis = 1)
        sorted_pc = np.append(sorted_pc, H, axis = 1)
        sorted_pc = sorted_pc[np.argsort(sorted_pc[:, 3])]
        sorted_data.append(sorted_pc)
    return sorted_data
```

Voxel Grid Downsampling - Exact

1. Compute the min or max of the point set $\{p_1, p_2, \cdots p_N\}$

$$x_{max} = \max(x_1, x_2, \cdots, x_N), x_{min} = \min(x_1, x_2, \cdots, x_N), y_{max} = \cdots\cdots$$

2. Determine the voxel grid size r
3. Compute the dimension of the voxel grid

$$D_x = (x_{max} - x_{min})/r$$
$$D_y = (y_{max} - y_{min})/r$$
$$D_z = (z_{max} - z_{min})/r$$

4. Compute voxel index for each point

$$h_x = \lfloor (x - x_{min})/r \rfloor$$
$$h_y = \lfloor (y - y_{min})/r \rfloor$$
$$h_z = \lfloor (z - z_{min})/r \rfloor$$
$$h = h_x + h_y * D_x + h_z * D_x * D_y$$

5. Sort the points according to the index in Step 4
6. Iterate the sorted points, select points according to Centroid / Random method
   0, 0, 0, 0, 3, 3, 3, 8, 8, 8, 8, 8, 8, 8, 8, ……

$$sorted\_data[k] = [x, y, z, h, h_x, h_y, h_z]$$

第一题

```python
def centroid_ds(r):
    sorted_data = voxel_grid(r)
    for pc, name in zip(sorted_data, names):
        pc = pc[:, :4]
        idx = np.flatnonzero(np.r_[True,pc[:-1,3] != pc[1:,3],True])
        counts = np.diff(idx)
        avg = np.add.reduceat(pc[:,:3],idx[:-1],axis=0)/counts.astype(float)[:,None]
        pc = np.c_[avg, pc[idx[:-1],3]]
        pc_view = o3d.geometry.PointCloud()
        pc_view.points = o3d.utility.Vector3dVector(pc[:, :3])
        o3d.visualization.draw_geometries([pc_view], window_name = name)
```

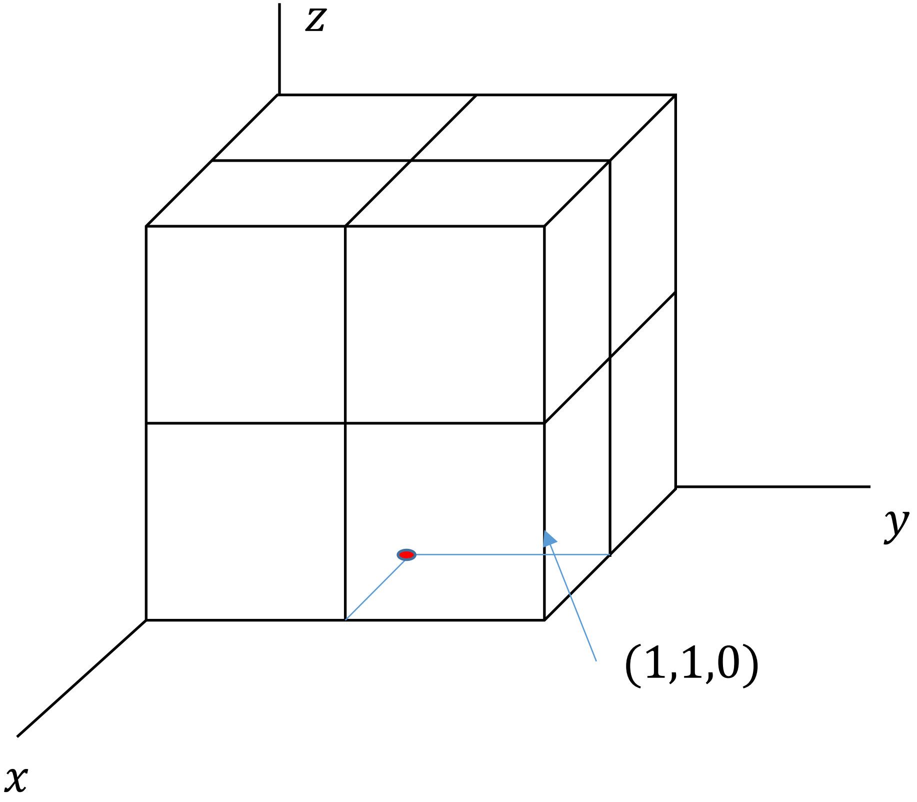
第一题

```
def random_ds(r):
    sorted_data = voxel_grid(r)
    for pc, pc_min, name in zip(sorted_data, data, names):
        min = np.amin(pc_min, axis = 0)
        pc = pc[:, 3:]
        pc = pc.astype(int)
        pc = np.unique(pc, axis = 0)
        random_pc = (pc[:, 1:] + np.random.rand(len(pc), 3)) * r + [min for i in range(len(pc))]
        pc_view = o3d.geometry.PointCloud()
        pc_view.points = o3d.utility.Vector3dVector(random_pc)
        o3d.visualization.draw_geometries([pc_view], window_name = name)
```

$$h_x = \lfloor (x - x_{min})/r \rfloor$$
$$h_y = \lfloor (y - y_{min})/r \rfloor$$
$$h_z = \lfloor (z - z_{min})/r \rfloor$$
$$h = h_x + h_y * D_x + h_z * D_x * D_y$$

$$sorted\_data[k] = [x, y, z, h, h_x, h_y, h_z]$$

第一题



第一题

```python
def random_ds(r):
    sorted_data = voxel_grid(r)
    for pc, pc_min, name in zip(sorted_data, data, names):
        min = np.amin(pc_min, axis = 0)
        pc = pc[:, 3:]
        pc = pc.astype(int)
        pc = np.unique(pc, axis = 0)
        random_pc = (pc[:, 1:] + np.random.rand(len(pc), 3)) * r + [min for i in range(len(pc))]
        pc_view = o3d.geometry.PointCloud()
        pc_view.points = o3d.utility.Vector3dVector(random_pc)
        o3d.visualization.draw_geometries([pc_view], window_name = name)
```

$$h_x = \lfloor (x - x_{min})/r \rfloor$$
$$h_y = \lfloor (y - y_{min})/r \rfloor$$
$$h_z = \lfloor (z - z_{min})/r \rfloor$$
$$h = h_x + h_y * D_x + h_z * D_x * D_y$$

$$sorted\_data[k] = [x, y, z, h, h_x, h_y, h_z]$$

纲要

- 整体思路
- 代码分析
  - 第一题
  - 第二题

第二题

深蓝学院
shenlanxueyuan.com

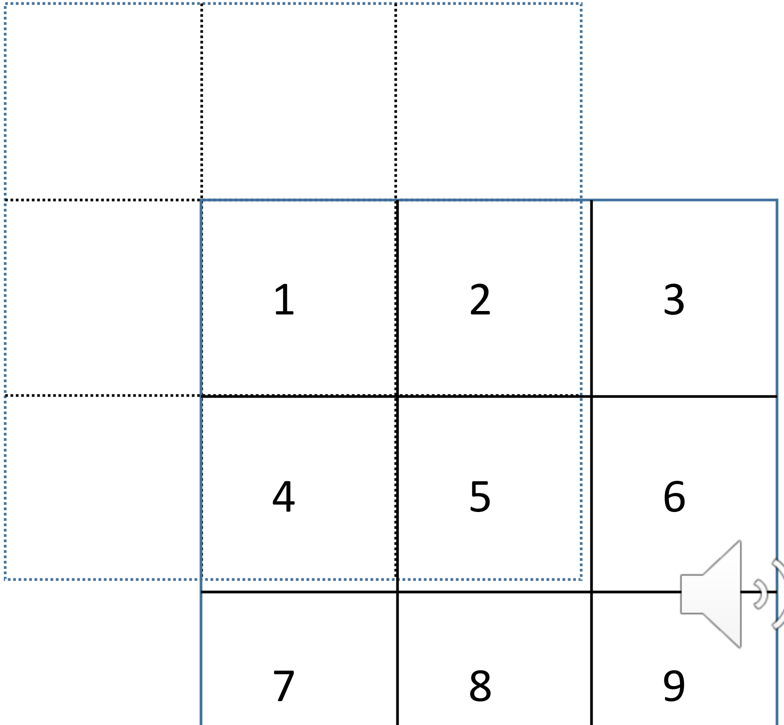
```python
def BF(image, depth, sigma_s, sigma_r):
    win_width = int( 3*sigma_s+1 )
    wgt_sum = np.zeros( depth.shape )
    result = np.zeros( depth.shape )

    for shft_x in range(-win_width,win_width+1):
        for shft_y in range(-win_width,win_width+1):
            # compute the spatial weight
            w = gaussian( shft_x**2+shft_y**2, sigma_s )

            # shift by the offsets
            image_off = np.roll(image, [shft_y, shft_x], axis=[0,1])
            depth_off = np.roll(depth, [shft_y, shft_x], axis=[0,1])

            # compute the value weight
            image_delta = image_off - image
            tw = w*gaussian( image_delta[:,:,0]**2 + image_delta[:,:,1]**2 + image_delta[:,:,2]**2, sigma_r )

            # accumulate the results
            result += depth_off*tw
            tw = tw * depth_off
            depth_off [depth_off == 0] = 1
            wgt_sum += tw / depth_off

    wgt_sum [wgt_sum == 0] = 1
    return result/wgt_sum
```

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

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

第二题

1	2	3
4	5	6
7	8	9



第二题

```
def BF(image, depth, sigma_s, sigma_r):
    win_width = int( 3*sigma_s+1 )
    wgt_sum = np.zeros( depth.shape )
    result = np.zeros( depth.shape )

    for shft_x in range(-win_width,win_width+1):
        for shft_y in range(-win_width,win_width+1):
            # compute the spatial weight
            w = gaussian( shft_x**2+shft_y**2, sigma_s )

            # shift by the offsets
            image_off = np.roll(image, [shft_y, shft_x], axis=[0,1])
            depth_off = np.roll(depth, [shft_y, shft_x], axis=[0,1])

            # compute the value weight
            image_delta = image_off - image
            tw = w*gaussian( image_delta[:,:,0]**2 + image_delta[:,:,1]**2 + image_delta[:,:,2]**2, sigma_r )

            # accumulate the results
            result += depth_off*tw
            tw = tw * depth_off
            depth_off [depth_off == 0] = 1
            wgt_sum += tw / depth_off

    wgt_sum [wgt_sum == 0] = 1
    return result/wgt_sum
```

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

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

在线问答

Q&A

深蓝学院
shenlanxueyuan.com

感谢各位聆听！

Thanks for Listening