Document

1. Introduction

This structure contains neighbors and local texture features for a point. Besides that, my method needs several ierations to minimize the energy of the whole point cloud, which is a complex computation containing millions of points' attributes, and those attributes are variable. Considering complexity, instead of using gradient descent techniques in deep learning, I choose to directly evaluate the expression for points. I separate constants from variables so that they can be used multiple times in each iteration.

2. File Structure

- **PCCTMC3Common.h**: Provides common PCC types and function definitions.

3. Main Functions

3.1 `EEDPCCPredictor`

```
struct EEDPCCPredictor {
  uint32_t EEDneighbours[kFixedEEDneighbourCount];
  float diffusiontensor[3][6];
  float* tensorcoffesforneighboursweight;
  uint32_t* neighbourindex;
  int32_t neighboursCount;
  EEDPCCPredictor()
  {
    memset(
      EEDneighbours, UINT32_MAX, sizeof(uint32_t) * kFixedEEDneighbourCount);
    memset(diffusiontensor, 0, sizeof(diffusiontensor));
    diffusiontensor[0][0] = diffusiontensor[0][1] = diffusiontensor[0][2] =
      diffusiontensor[1][0] = diffusiontensor[1][1] = diffusiontensor[1][2] =
        diffusiontensor[2][0] = diffusiontensor[2][1] = diffusiontensor[2][2] =
          1;
    neighboursCount = 0;
  }
}
```

- **Description**: Constructor initializing the structure.
- **Parameters**:
  - `EEDneighbours`:An array representing the indices of potential neighboring points. The size of this array is defined by `kFixedEEDneighbourCount` (Use 27 to make it easier to expand, 18 is actually used).
  - `diffusiontensor`: Represents the diffusion tensor for each point. It is a 3x6 matrix.
  - `tensorcoffesforneighboursweight`: Dynamic memory allocated for storing coefficients related to neighbors' weights during the computation. It is freed using the `freetensorcoffesforneighboursweight` method.
  - `neighbourindex`: Dynamic memory allocated for storing the indices of neighbors. It is used to map neighbors' indices to positions in the tensor coefficients array.
  - `neighboursCount`: Represents the count of neighbors for a point.

3.2 `freetensorcoffesforneighboursweight`

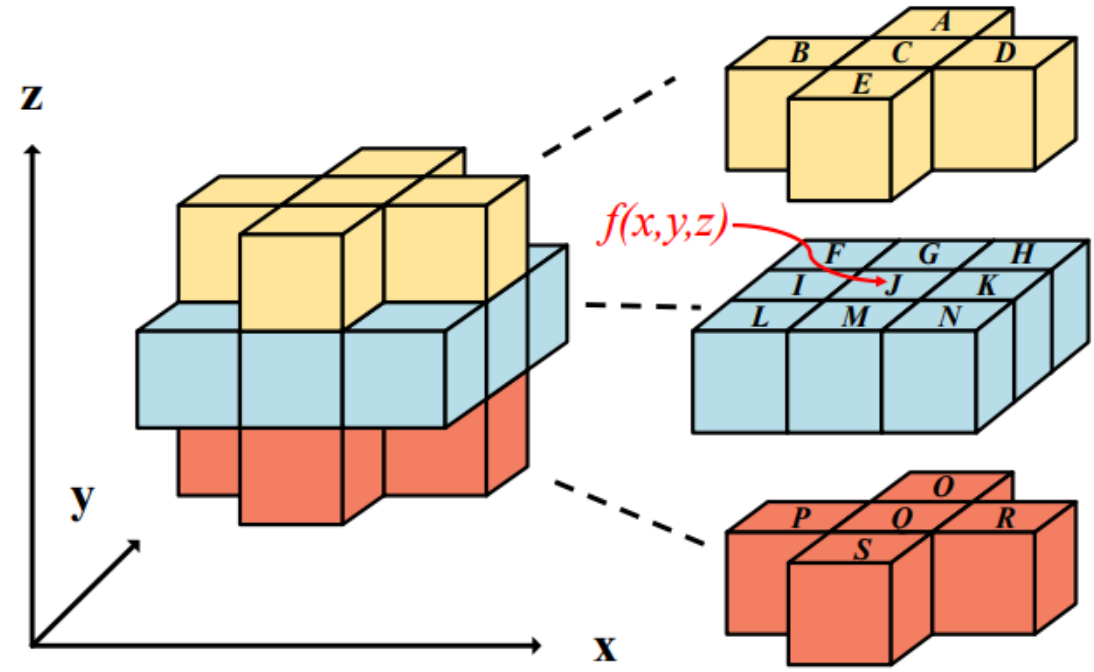
```
void freetensorcoffesforneighboursweight();
```

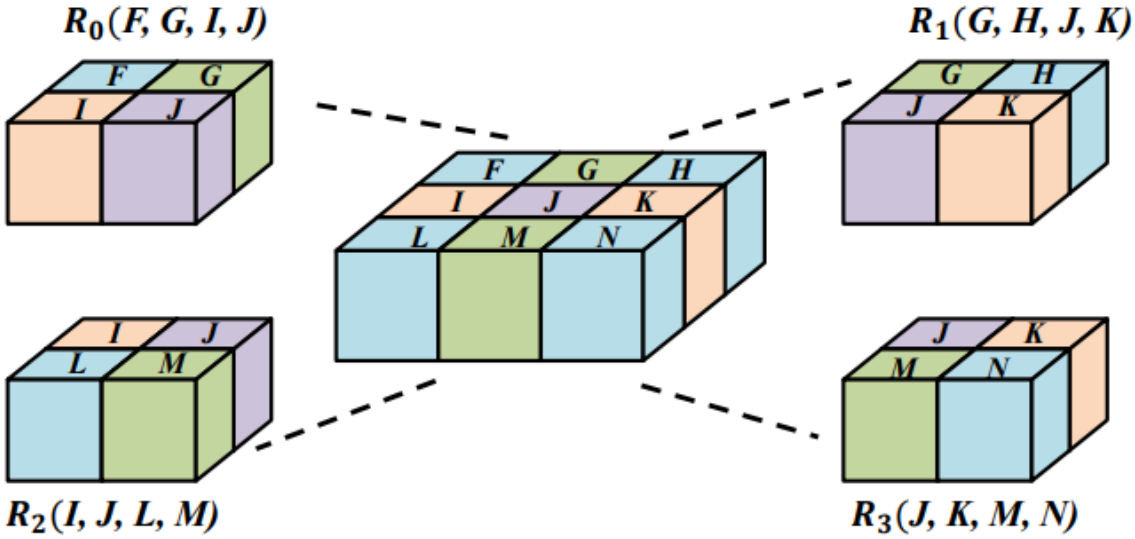
- **Description**: Frees the dynamically allocated memory for `tensorcoffesforneighboursweight`.

3.3 `CalUXYZ0`&`CalUXYZ1`&`CalUXYZ2`&`CalUXYZ3`

```
  void CalUXYZ0(
    const int32_t Utemplate[4][4],
    float distance,
    int32_t occupycode,
    size_t number,
    size_t dim1,
    size_t dim2,
    uint32_t index2pos[27]);
  void CalUXYZ1(
    const int32_t Utemplate[4][4],
    float distance,
    int32_t occupycode,
    size_t number,
    size_t dim1,
    size_t dim2,
    uint32_t index2pos[27]);
  void CalUXYZ2(
    const int32_t Utemplate[4][4],
    float distance,
    int32_t occupycode,
    size_t number,
    size_t dim1,
    size_t dim2,
    uint32_t index2pos[27]);
  void CalUXYZ3(
    const int32_t Utemplate[4][4],
    float distance,
    int32_t occupycode,
    size_t number,
    size_t dim1,
    size_t dim2,
    uint32_t index2pos[27]);
```

- **Description**: These functions analyze the neighor occupancy for each point, which will be later used to evaluate the expression for points. Due to point clouds' occupancy uncertainty, there are 2^{18} occupancy situations. So, I divide the complex situations into several small situations. I analyze only 4 points in a local region at a time.





- **Parameters**:
  - `distance`: Distance parameter for weight calculation.
  - `occupycode`: Occupancy code specifying the neighbor pattern.
  - `number`: Index used in `tensorcoffesforneighboursweight` calculations.
  - `dim1`: Dimension 1 of the tensor.
  - `dim2`: Dimension 2 of the tensor.
  - `index2pos`: Mapping of indices to positions.

3.4 `Caltensorcoffesforneighboursweight`

```
void Caltensorcoffesforneighboursweight(
  float diffX[27],
  float diffY[27],
  float diffZ[27],
  const int32_t UXUYtemplate[4][4],
  const int32_t UXUZtemplate[4][4],
  const int32_t UYUZtemplate[4][4],
  const float distance);
```

- **Description**: This function uses previously defined functions to evaluate the expression for points in a point cloud.
- **Parameters**:
  - `diffX`: Array of differences in the X-direction for neighboring points.
  - `diffY`: Array of differences in the Y-direction for neighboring points.
  - `diffZ`: Array of differences in the Z-direction for neighboring points.
  - `UXUYtemplate`: Template for neighboring points in the XY-direction.
  - `UXUZtemplate`: Template for neighboring points in the XZ-direction.
  - `UYUZtemplate`: Template for neighboring points in the YZ-direction.

3.5 `predictColorFast`

```
Vec3<attr_t> predictColorFast(
  const PCCPointSet3& EEDpointCloud,
  std::vector<EEDPCCPredictor>& EEDpredictors);
```

- **Description**: This function predicts the color of a point in the point cloud by considering neighboring points, their weights, and diffusion tensors. It is used to get the prediction of points' attributes in each iteration to minimize the energy of point cloud step by step.
- **Parameters**:
  - `EEDpointCloud`: Input point cloud.
  - `EEDpredictors`: Vector of predictors for each point in the point cloud.

3.6 `gussianfilter`

```
void gussianfilter(
  const PCCPointSet3& EEDpointCloud,
  const std::vector<float> neighboursW,
  PCCPointSet3& GussianEEDpointCloud);
```

- **Description**: Creates Gaussian weights for Gaussian filtering of the point cloud.
- **Parameters**:
  - `EEDpointCloud`: Input point cloud.
  - `neighboursW`: Vector storing Gaussian weights.
  - `GussianEEDpointCloud`: Output point cloud after applying the Gaussian filter.

3.6 `updateDiffusionTensor`

```
void updateDiffusionTensor(
  const PCCPointSet3& EEDpointCloud,
  const PCCPointSet3& GussianEEDpointCloud,
  float diffX[27],
  float diffY[27],
  float diffZ[27],
  float lambda,
  bool* isseed,
  bool* ispredicted,
  float seedweight,
  float predictedweight);
```

- **Description**: Use attributes updated in iterations to recalculate local texture features.
- **Parameters**:
  - `EEDpointCloud`: Input point cloud.
  - `GussianEEDpointCloud`: Point cloud after applying the Gaussian filter.
  - `diffX`: Array of differences in the X-direction for neighboring points.
  - `diffY`: Array of differences in the Y-direction for neighboring points.
  - `diffZ`: Array of differences in the Z-direction for neighboring points.
  - `lambda`: `Lambda` parameter for diffusion tensor calculation.
  - `isseed`: Array indicating if a point is in the first layer.
  - `ispredicted`: Array indicating if a point has been predicted.
  - `seedweight`: Weight for seed points.
  - `predictedweight`: Weight for predicted points.