Document

This project is based on the point cloud compression platform TMC13 of MPEG, which focuses on using compression techniques to eliminate redundancy in both the geometric and attribute information of 3D point clouds, allowing for efficient storage and transmission. In TMC13, there are three compression methods for point cloud attributes, corresponding to three functions: `encodeColorsTransformRaht`, `encodeColorsPred`, and `encodeColorsLift`. In this project, we implemented a prediction method based on partial differential equations, corresponding to the function `encodeColorsEED`.

Intent

On the basis of TMC13, I primarily added `LOD4EED.cpp` and `EEDpredictor.h`. Additionally, I made modifications to files such as `AttributeCommon.cpp`, `AttributeEncoder.cpp`, and `AttributeDecoder.cpp`. This resulted in the incorporation of a point cloud attribute prediction encoding method based on partial differential equations for TMC13. Under lossy conditions, this enhancement led to a 12.00% increase in encoding efficiency.

The proposed method involves partitioning the point cloud into a two-layer LOD structure. It constructs a global point cloud energy by computing the gradient of point cloud attributes. The second layer is predicted by minimizing the energy, and simultaneously, the texture features of the first layer are utilized to guide the prediction of the second layer.

`LOD4EED.cpp` contains functions about the special level of detail method designed for my prediction framework. The function `subsampleForEED` use a `MortonIndexMap3d` structure to storage positions of points in local region and update them. In this way, the complexity will be reduced from n^2 to n . This function splits the point indexes in input into "retained" and "indexes", enabling each point in retained has at least one neighbor in indexes. All other functions in the code have detailed comments.

`EEDpredictor.h` defines `EEDPCCPredictor` structure, which contains neighbors and local texture features for a point. Besides that, my method needs several iterations 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.

Input

For the encoder, the input consists of two files, namely the original point cloud (longdress_vox10_1300.ply) and the configuration file for encoding conditions (encoder.cfg).

For the decoder, the input consists of two files, namely the compressed file (longdress_vox10_1300.bin) and the configuration file for decoding conditions (decoder.cfg).

Output

For the encoder, the output is the compressed file (longdress_vox10_1300.bin).

For the decoder, the output is the reconstructed point cloud (longdress_vox10_1300_dec.ply).