CIS Background Report
Motion Compensation and Evaluation of 3D Head Reconstruction

April 5, 2022

Students:
Unnat Antani - uantani1@jhu.edu

Mentors:
Can Kocabalkanli
Ozgur Guler
Reza Seifabadi
Pediametrix Inc.

1 Instant Neural Graphics Primitives with a Multi-resolution Hash Encoding

1.1 About

The first paper we explore is titled "Instant Neural Graphics Primitives with a Multi-resolution Hash Encoding" by Muller, Thomas and Evans, Alex and Schied, Christoph and Keller, Alexander. This article was published on arXiv on January 16, 2022.

The authors here have developed and implemented Neural Network-based solution for reconstructing a 3D scene using 2D images and optimizing the training of the model using hash encoding. On a good hardware setup, the system outputs a 3D render of the scene from 2D images in a few seconds.

The traditional methods, i.e., the non-AI based methods have a lot of data processing to be done before any algorithm can be run on it. Not only that, most traditional methods work on point clouds directly, instead of 2D images. The number of points ranges in thousands, if not more. Most methods applied will take quite some time to process thousands of points, even with reduction through thresholding or outlier removal. In many cases, one might have to take advantage of complex data structures like KDTree or Octree to manipulate data

in an efficient manner. It also means, we would need a 3D scanner each time we need to gather data, which are expensive and not always readily available.

Traditional method also requires one to calculate distortions due to motion or noise and compensate analytically, which is a highly complex task. It depends largely on how the scans are obtained, the type of sensor used, the conditions of the environment when taking a scan and the amount of overlap each scan has.

The solution introduced employs use of AI to extract the features, generate registered 3D scene with just 2D images as input, which removes complexity of the data handling with point clouds. The Neural Network can be trained on number of parameters including exposure, intrinsics and distortion.

1.2 Technical Summary

In computer vision, the graphics are represented as a mathematical function. The goal is always to find a function which is fast, but also retains much of the information of the images. A Multi-layer Perceptron, MLP, serves as such a function.

The features are mapped on a higher-dimensional through encoding, which is responsible for preserving or having good approximation of the actual image. In all the images, for finer resolution, each grid is considered a hash table and mapped to lookup tables in which do not require control flow and hence can be obtained in $O(1)$ time.

The above hashing method is used in 4 representative tasks,

1. Gigapixel Image: 2D to RGB for a very high pixel image
2. Neural Signed Distance Function: Maps coordinates to distance to a surface.
3. Neural Radiance Caching: The NN learns 5D light scene
4. Neural Radiance and Density Field: NN learns the 3D density and 5D light scene from images and corresponding transforms.

For encoding, the spatio-directionally varying light field as well as density are encoded as multiresolution of $L \in N$ sine and cosine functions:

$$enc(x) = (sin(2^0 x), sin(2^1 x), \cdots, sin(2^{L-1} x), cos(2^0 x), cos(2^1 x), \cdots, cos(2^{L-1} x)$$

The given solution not only has trainable weights for the neural network, but also trainable weights for encoding, which means that the entire systems is independent of the task and trains on getting the best approximations on the fly on the given images. After training, for any input, the neural network uses the hash table of the encoding generated and interpolates to generate the approximation.

The entire solution has been implemented on CUDA. The code is avaialble

on the author's github page, linked in the reference section. The performance of the model however, depends heavily on the hardware setup, i.e., on the GPU architecture.

THE MLP/NN arhcitecture itself is quite simple. There are 2 hidden layers, with 64 neurons each and activation functions ReLU. The losses used and the ouput activation depends on the task and can be set when implementing.

For NeRF architecture, the one applicable in our case, uses 2 MLPs side by side, a density predictor, followed by color MLP. The 16 output corresponds to the 16 coffecients of the first 16 terms of the harmonic basis function.
After going through the images, the features are then stitched together, which gives the 3D scene learnt from the images.

They also have a 3D reconstruction option which uses marching cube algorithms to generate a mesh from the 3D scene learnt.

1.3 Analysis

PROS:

- The solution introduced is independent of the task and doesn't need much preprocessing of the input images.
- However, one still needs to use another software, COLMAP, to generate the camera poses from the images captured. These transformations along with the images, are the input to the MLP, which ultimately renders the 3D scene in seconds.
- This method also has option to learn distortions in the scene, which means minor distortions due to small movements can be taken care of by the interpolation of hashing.
- It also doesn't need a large dataset to be trained on. It works on as few as 15 images, though more number of images would make it more accurate.
- The use of latest AI research means that it will always have scope of expansion, and the performance will only get better.

CONS:

- The solution depends on the availability of a GPU, which will dictate the performance of the system.
- The surface reconstruction available in their solution does not give good results with custom dataset.
- Preprocessing to get the camera poses through COLMAP is a tedious process.
- The dependencies are very strict with versions of packages used.

2 Graph-Based Compression of Dynamic 3D Point Cloud Sequences

2.1 About

The second paper we explore is titled "Graph-Based Compression of Dynamic 3D Point Cloud Sequences" by Thanou, Dorina and Chou, Philip and Frossard. This article was published in IEEE Transactions on Image Processing in 2015.

The solution given in the paper, manipulates the data structure of Octree, and efficiently uses graph encoding and feature matching on graphs to register and construct the 3D scene without any distortion errors.

The implementation is directly related to the problem being solved, which is to estimate and compensate for the error generated due to motion, either by the object being scanned or motion of the camera during scanning.

2.2 Technical Summary

The point clouds are firstly decomposed into octree for better and faster manipulation of the data through representation as graphs.
Each node of the graph, is embedded with information of coordinates and color. This is done by embedding them in terms of signals by fourier transforms. Thus, the features are embedded in the graph nodes. In the end, the problem is then converted to a feature matching problem on graphs. The edges represent transformation between the points.

The similarity in the features in consecutive graphs is measured by Mahalanobis distance. After getting the best matched point, the motion is estimated with assumption that surrounding points will also go through similar motion, and hence using the graph structure, a smooth motion field is generated. This motion field is then propagated smoothly over all the points using quadratic function. In the end, we get a stitched graph, which is motion compensated.

Now, given the motion field, we use that as predictor model and compare with the target. If the color corresponds to the predicted color, it means that it is redundant and hence discarded. Only the points where the prediction is different than the target color attributes are kept. This removes any noisy data generated due to motion.

2.3 Analysis

PROS:

- Works with the 3D scan points.
- Compensates for position and color.

- Works on octree data structure which reduces the dimensionality of the data and improves efficiency.

CONS:

- Still have to preprocess thousands of points to generate octree.
- Can lose information while converting to graph.
- Not guarantee to give water-tight meshes.
- Still takes some time, which only increases with increase in number of input points.
- Depends on the quality of input data and the overlap.

3 References

References

[1] Müller, Thomas and Evans, Alex and Schied, Christoph and Keller, Alexander (2022) *Instant Neural Graphics Primitives with a Multiresolution Hash Encoding*, [paper].

[2] Thanou, Dorina and Chou, Philip and Frossard, Pascal (2015) *Graph-Based Compression of Dynamic 3D Point Cloud Sequences*, IEEE Transactions on Image Processing, [paper]