High-Performance Compression of Scientific Volume Data Using Learned 3D Gaussian Models

Landon Dyken[1,2], PhD Student; Nathan DeBardeleben[2], PhD. | [1]University of Illinois Chicago; [2]Los Alamos National Laboratory, HPC-DES

Introduction

Why is compression needed?

As HPC systems continue to increase in scale, the datasets output by scientific simulations are becoming larger and more complex. Effectively using these datasets for end user tasks presents a challenge, as they regularly exceed the memory of user systems. To address this, compression methods are needed that can reduce data's memory footprint and support interactivity.

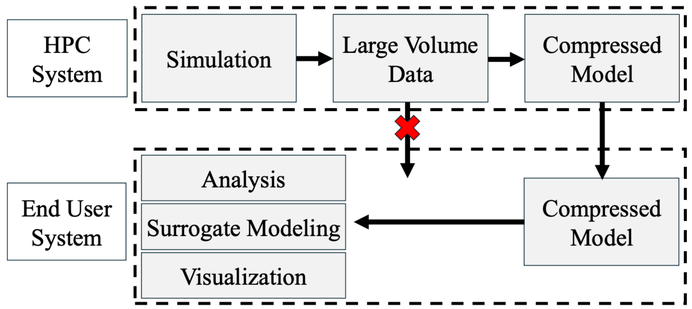


Figure 1. Compression of volume data

Method

What are 3D Gaussians?

In this project, we explore the potential for compressing scientific volumes by approximating them as collections of 3-dimensional Gaussians. Each Gaussian is defined as a scaled and rotated distribution around a point in 3D space. By using fewer Gaussians than data points in the ground truth, and removing the need to store connectivity information, a 3D Gaussian model can represent a dataset using a fraction of the original memory.

How is a 3D Gaussian model created?

The first step of creating a 3D Gaussian model is to initialize a collection of Gaussians from the original volume. To do this, we first discard the volume's connectivity information, leaving only its data points with attached scalar values. These data points are extended to become 3D Gaussians with a spherical distribution and scaled depending on the density of surrounding points. This process is shown in Figure 2. In practice, we discard some fraction of data points before initializing, depending on the desired compression ratio for the trained model.

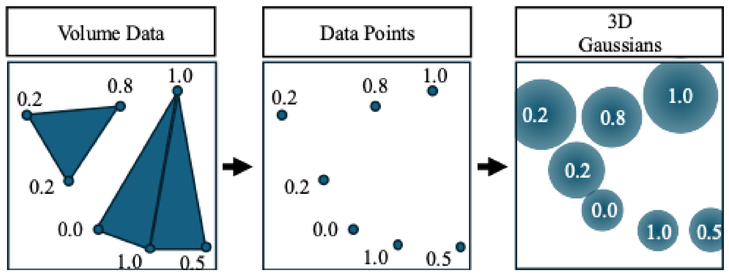


Figure 2. 3D Gaussian initialization

How is a 3D Gaussian model trained?

Once a collection of 3D Gaussians is initialized, they can be trained to reconstruct a volume using gradient descent. This is done by sampling the 3D Gaussian model and the ground truth, computing the reconstruction loss between the two sets of sampled points, then backpropagating gradients to the individual 3D Gaussians. An example of this process is shown in Figure 3. At figure right, a depiction of how the Gaussians may change after gradients are applied is shown. Note how the Gaussians scale and rotate to better match the geometry of the ground truth.

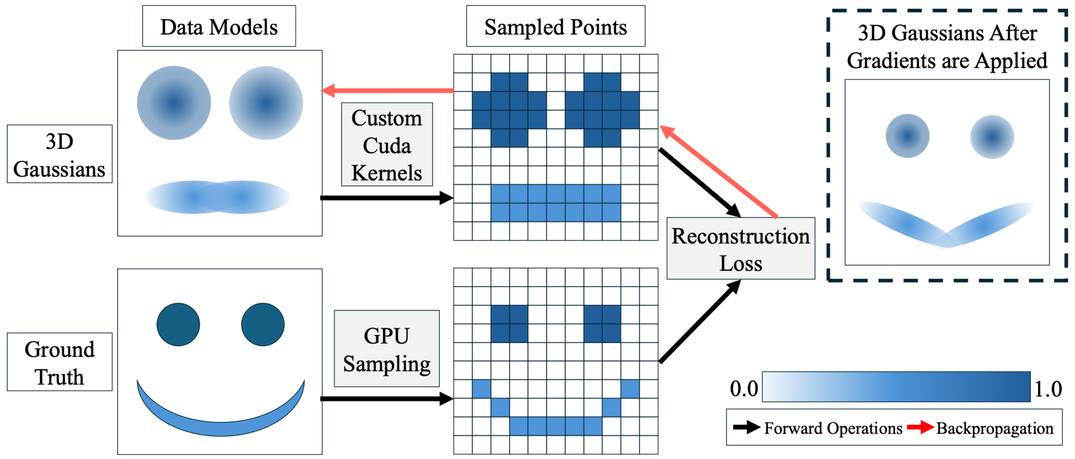


Figure 3. 3D Gaussian training

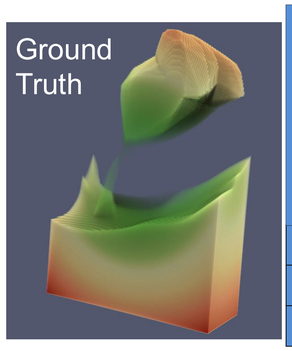
How is training implemented?

For a 3D Gaussian model to accurately represent a dataset, many iterations of training are required, necessitating optimized GPU performance to keep training times low. To accomplish this, we implement 3D Gaussian sampling and backwards pass gradient computation using custom Cuda kernels, and hook these into our PyTorch training context, allowing our method to proceed efficiently on state-of-the-art Nvidia GPUs.

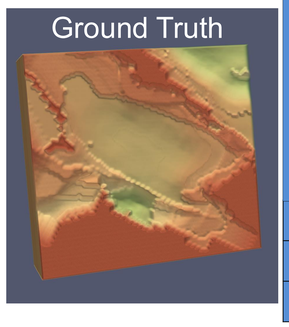
Results

How do 3D Gaussian models perform for compression?

We present results for compression factor, reconstruction quality (PSNR, higher is better), training time, and rendering for 3D Gaussian models trained on two separate datasets. All models were trained for 16,000 iterations on an Nvidia A40 GPU. Testing was done using 8 million ground truth samples not seen during training.



Mito Dataset 235 MB 5.54M Cells 972K Points						
Compression	5x	10x	20x	40x	80x	160x
PSNR	26.96	26.88	26.57	26.10	24.66	14.99
Training Time	5 min 23 sec	3 min 7 sec	2 min 3 sec	1 min 38 sec	1 min 21 sec	1 min 19 sec



SF1 Dataset 594 MB 14.0M Cells 2.46M Points						
Compression	5x	10x	20x	40x	80x	160x
PSNR	20.21	20.22	20.12	19.72	18.88	15.81
Training Time	16 min 14 sec	8 min 44 sec	5 min 13 sec	3 min 23 sec	2 min 32 sec	2 min 8 sec

Los Alamos NATIONAL LABORATORY
NNSA National Nuclear Security Administration
Managed by Triad National Security, LLC, for the U.S. Department of Energy's NNSA.
ROSY#: 81f072d2 LA-UR-25-28269