3D Gaussian Point Encoders: Supplementary Material

Jim James
Georgia Tech
jimjames@gatech.edu

Benjamin Wilson
Georgia Tech

Simon Lucey
University of Adelaide

James Hays
Georgia Tech

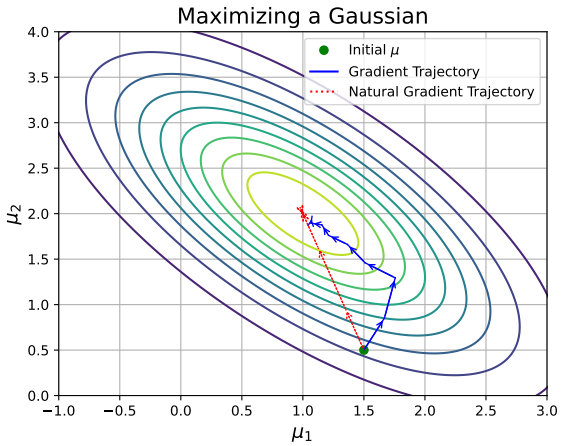


Figure 1. **Effect of Natural Gradients.** We apply Mahalanobis natural gradients on a simple optimization problem (finding the mean to maximize a given Gaussian's likelihood). The Mahalanobis natural gradient steps directly towards the optimum, leveraging the geometry of the Gaussian, while the standard gradient takes a longer, less stable trajectory.

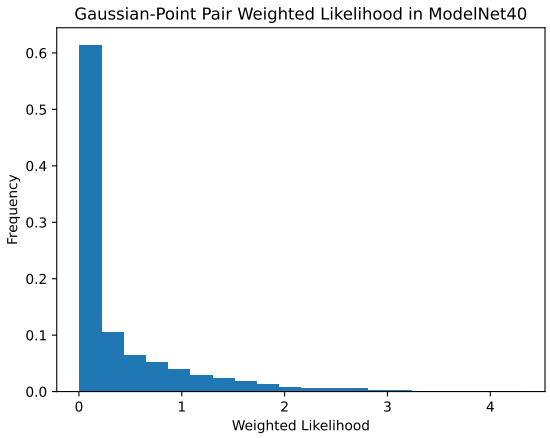


Figure 2. **Weighted Likelihood Distribution on ModelNet40's Training Set.** Approximately 60% of weighted Gaussian-point pair likelihoods are near zero, motivating filtering.

1. Example Problem with Mahalanobis Natural Gradients

We design a simple optimization problem to demonstrate how the Mahalanobis natural gradient differs from the standard gradient. Consider the following optimization problem:

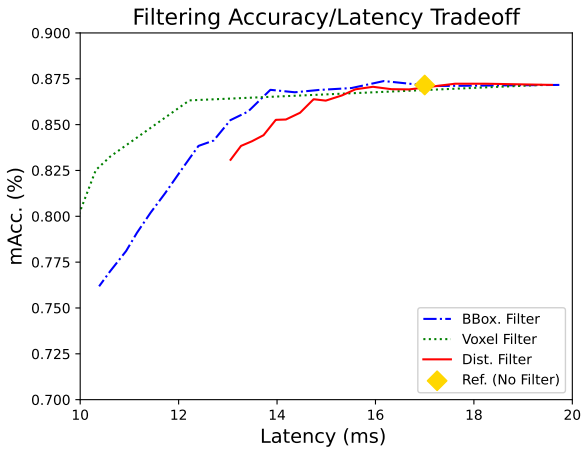
$$\arg\max_{\mu \in \mathbb{R}^2} \exp\left(-(x-\mu)^\top \Sigma^{-1}(x-\mu)\right)$$

In essence, we are trying to find the mean of a Gaussian that maximizes the likelihood at a given point. Intuitively, its solution is simply $\mu^* = x$. We attempt to solve this via gradient ascent from a random starting point in Fig. 1 with both Mahalanobis natural gradients and standard gradients. The Mahalanobis natural gradient finds the optimal solution in fewer iterations with the same learning rate, with each update pointing directly to the optimum. On the other hand, the standard gradient will always be orthogonal to contours in the graph, which for an anisotropic Gaussian do not necessarily point to its center. Accordingly, the standard gradient converges slower, taking a more jagged trajectory.

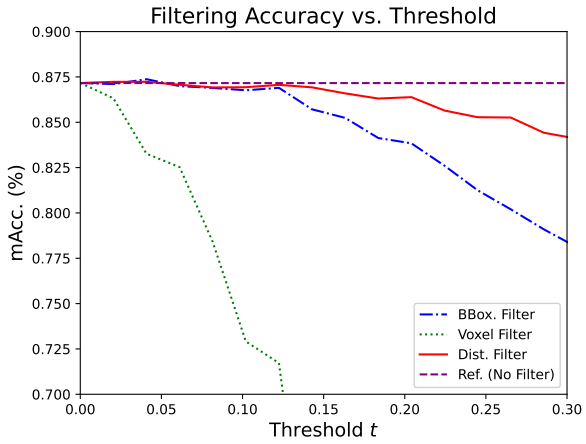
2. Ablation on Accuracy and Runtime with Filtering Thresholds

We measure how adjusting the filtering thresholds t_{distance}, t_{bbox}, and t_{voxel} impact accuracy and runtime on our 3DGPE model distilled from PointNet. Fig. 3a demonstrates how class-averaged accuracy varies with the latency for each filtering method, while Fig. 3b indicates how sensitive each filtering method's threshold is. Voxel filtering is the most effective, being able to reduce runtime the most without a significant loss in accuracy. Both bounding-box filtering and distance filtering result in larger accuracy tradeoffs in order to reduce runtime. However, both bounding-box and distance filtering are significantly less sensitive to the threshold value than voxel filtering. We hypothesize this is a consequence of voxel filtering most closely bounding each Gaussian, making it filter larger likelihood values than the other methods at the same threshold.

Overall, we find approximately a **30%** reduction in like-



(a) **Filtering Performance vs. Latency.**



(b) **Filtering Threshold Sensitivity.**

Figure 3. **Ablations on Filtering.** Voxel filtering is the most effective at reducing runtime while minimizing accuracy loss, followed by bounding-box filtering, and finally distance filtering. However, voxel filtering is much more sensitive to the value of the threshold compared to both distance and bounding-box filtering.

lihood computation latency without significant performance impacts using voxel filtering, and a **20%** reduction for bounding-box filtering. However, we consider bounding-box filtering to be the best option, as it is far less sensitive to its threshold value than voxel filtering, while also providing a reasonable speedup. Bounding-box filtering also scales better with input dimension compared to voxel filtering, as the bounding boxes can be computed in linear time.

3. Additional Implementation Details

We implement additional optimizations on our encoder to enhance inference throughput. At test time, we pre-compute the precision matrices from their respective Cholesky factors to avoid unnecessarily recomputing them for every point cloud. Furthermore, after computing the transformation matrix from the 3D Gaussian T-Net in our PointNet experiments, we apply the inverse transform to the Gaussian parameters rather than apply the transform to the points themselves. This reduces computational costs as there are significantly fewer Gaussians (at most 64) than points per sample (1024 to 2048). However, this optimization is incompatible with voxel filtering, as it would require re-computing the voxel grid for every sample.

We train all of our models on an NVIDIA GeForce RTX 5090 GPU. Because of the minimum CUDA version supported by Blackwell GPUs, we use PyTorch version 2.8.0 on CUDA 13.0 and newer versions of Mamba-SSM and Causal-Conv1D compared to the original Mamba3D [1] paper. We observe a minor degradation in performance metrics from the paper's original results, even from the publicly available checkpoints, which we attribute to these major library versioning differences.

References

[1] Xu Han, Yuan Tang, Zhaoxuan Wang, and Xianzhi Li. Mamba3d: Enhancing local features for 3d point cloud analysis via state space model. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 4995–5004, 2024. 2