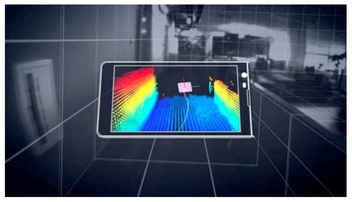
GPU-Accelerated 3D Surface Reconstruction using Gaussian Mixture Sampling and Sparse Voxel Lists

Benjamin Eckart[†], Alejandro Troccoli[‡], Kihwan Kim[‡], Jan Kautz[‡]
[†] The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA USA
[‡] NVIDIA Research, Santa Clara, CA USA

Introduction

From self-driving cars to mixed reality on cell phones, there is a fast growing need to understand how to build advanced spatial awareness into our smart devices.

SoftKinetic's Time-of-Flight Sensor



Project Tango's Structured Light Sensor

Typical 3D sensors, such as the SoftKinetic or Google's Project Tango (pictured above), operate by generating vast amounts of samples of solid objects many times per second. The collection of these samples is referred to as a point cloud, and this data structure forms the basis of many spatial perception algorithms. It is therefore important to develop efficient and robust techniques to manage and process point cloud data.

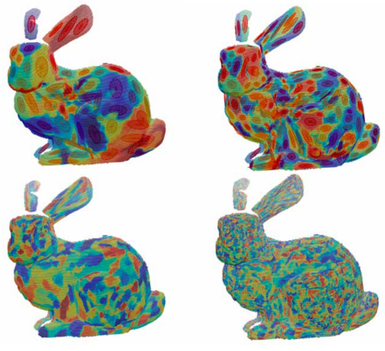
Contributions

A common operation of particular importance is **3D Surface Reconstruction**: the process of deriving solid 3D geometry from unorganized sets of points.

In this poster, we describe a parallel method to hierarchically process and compress 3D point data into a statistical parametric form from which a 3D triangle mesh can be quickly constructed. We leverage our model through sparse, parallelized importance sampling to facilitate a stochastic marching cubes surface extraction.

Parametric Model: Gaussian Trees

We build our parametric model by recursively creating a tree of 8-component Gaussian Mixture Models in a top-down fashion.

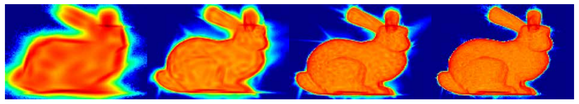


Stanford bunny models from different levels in the GMM tree. Each color denotes the area of support of a single Gaussian, and the ellipsoids indicate their 1 sigma extent.

Our spatial decomposition can be compared to an octree, but instead of recursively subdividing space into regular octants, we use parallelized Expectation Maximization (EM) to allow the subdivisions to be probabilistic and data driven.

Stochastic Surface Extraction

Since we can produce a valid probability distribution function from our Gaussian Tree (a slice through the tree produces a regular Gaussian Mixture Model), any isovalue of this PDF will form a probabilistic "shell" around our data according to a given confidence value. Thus, the isosurface extraction process forms the basis of our surface reconstruction.



The graphics above show sample PDF from 4 different levels in the Gaussian tree, from lowest to highest fidelity. Any isocontour of these PDF's forms a valid 3D isosurface associated with some confidence value.

Sparse Sampling for Meshification

A straightforward method for extracting this isosurface from the Gaussian hierarchy would be to sample densely in a regular grid and triangulate over isovalue crossings using the Marching Cubes algorithm, but this is quite inefficient since most of the 3D volume is sparse.

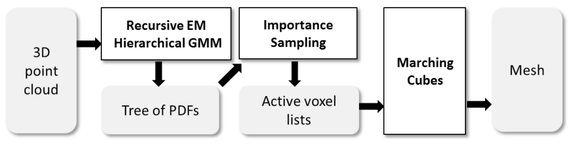
Thus, we use Importance Sampling (IS) in order to quickly and sparsely find regions of high 3D spatial probability. Given our model, Θ, the probability of a particular voxel V_k is related to the Gaussian Mixture parameters $\pi_j, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j$ for J mixtures and N samples,

$$p(V_k|\Theta) \approx \sum_{j=1}^{J+1} \frac{\pi_j}{N} \sum_{i=1}^{i=N} I_{V_k}(x_i)$$

where $x_i \sim \mathcal{N}(\boldsymbol{\mu}_j|\boldsymbol{\Sigma}_j)$ and I is an indicator function for whether x_i falls within the bounds of V_k.

Note that each sample can be tested independently and in parallel, leading us to a very fast CUDA implementation. We can then send these samples to a CUDA-accelerated Marching Cubes routine for the final meshification.

Pipeline Overview



We first build a hierarchical GMM by recursively applying EM. Secondly, we run stochastic importance sampling on the highest detail PDF to build a sparse list of active voxels that we feed to marching cubes for the final surface reconstruction.

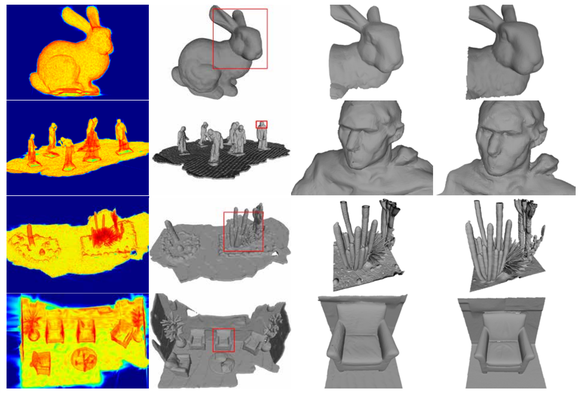
Testing Datasets

We tested our method over several standard datasets. From left to right:

- Bunny - 35,947 points (0.41MB for points/1.21MB for mesh)
- Burghers of Calais - 3.47M points (39.68MB/118.17MB) [4]
- Cactus Garden - 1.9M points (21.71MB/64.70MB) [4]
- Lounge - 1.62M points (18.50MB/54.56MB) [4]

Procedural Surface Reconstruction

Using these datasets, we first constructed our model and then sampled from the model in order to procedurally generate the mesh. Below we compare the original models to our PDFs and reconstructed surfaces.



(1st column): Heat maps represent the areas of high data probability (PDFs) for each model, (2nd column): The reconstructed scenes with our hierarchical GMM and stochastic marching cubes in low voxel resolution (256^3). The red rectangle denotes the region of selection where we reconstruct with higher resolution. (3rd column): Our reconstruction with higher quality. (4th column): Original meshes from [4]. One can see from these results that our method is able to reconstruct surfaces with high fidelity.

Space Efficiency

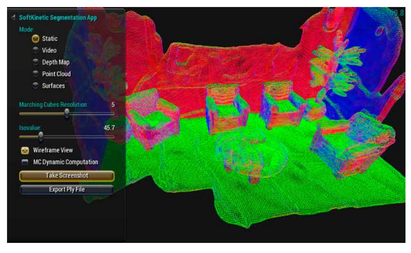
To test our space efficiency, we compared the fidelity of our model with that obtained by a simpler strategy: random subsampling.

Size (kB)	Ours (PSNR)	Subsampling (PSNR)
0.31	62.3	28.1
2.5	65.7	33.8
20	68.1	42.5
159	70.3	51.1
1270	72.3	59.9

We subsampled the point cloud data at varying levels to match the model sizes and compare PSNRs. We use Peak Signal to Noise Ratio (PSNR) as our quality metric, which is calculated from the RMSE according to [2].

Compared to subsampling, the smallest GMM doubles the PSNR. Similarly, the largest GMM has 10 more PSNR than a subsampled point cloud of equivalent size.

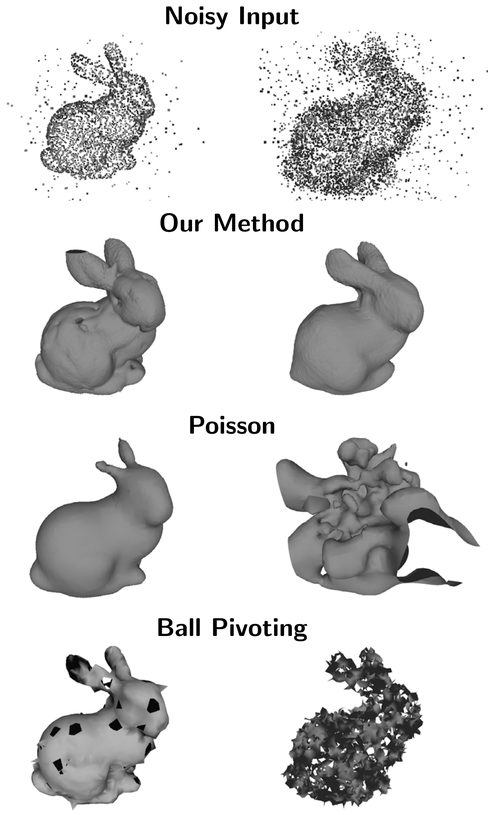
Native Android CUDA Application



To demonstrate our computational efficiency, we deployed this pipeline onto an NVIDIA Shield Tablet. Our hierarchical segmentation is able to stream SoftKinetic data and segment in greater than real-time (356 FPS for a coarse level 1 segmentation and 124 FPS for a high fidelity level 5 segmentation). Our sparse sampling strategy for a 256x256x256 Marching Cubes grid takes about 2.5 ms (400 FPS) for a level 4 segmentation in CUDA.

Reconstruction Comparison under Noise

The probabilistic representation of 3D structure allows us to implicitly model noise by maintaining surface covariances, and perform outlier rejection with the inclusion of a uniform mixture.



(1st row): Noisy model where the noise was randomly added (Normal distribution), (2nd row): *our results*, (3rd row): Poisson reconstruction [3], (4th row): Ball-pivoting [1]. Note that even with more aggressive noise (2nd column), our approach can effectively handle the noise, providing better overall model quality.

Conclusions

We have shown that our proposed model is good for compression, handling noise, and reconstructing surfaces with high fidelity. Furthermore, given the parallel and recursive construction of our model, it can be efficiently implemented in CUDA, allowing for real-time rates on a mobile device.

References

[1] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin.
The ball-pivoting algorithm for surface reconstruction.
IEEE Transactions on Visualization and Computer Graphics, 5(4):349–359, 1999.

[2] Paolo Cignoni, Claudio Rocchini, and Roberto Scopigno.
Metro: Measuring error on simplified surfaces.
Computer Graphics Forum, 17(2):167–174, 1998.

[3] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe.
Poisson surface reconstruction.
In *Eurographics Symposium on Geometry Processing*, pages 61–70, 2006.

[4] Qian-Yi Zhou and Vladlen Koltun.
Dense scene reconstruction with points of interest.
ACM Transactions on Graphics, 32(4):112, 2013.