Massively Parallelized Point Spread Function Noise Filtering for Geiger Mode LiDAR

HARRIS

Nicholas Shorter, O'Neil Smith, Robert Stark, Philip Smith, Randall St Romain, Steve Blask

Harris Corporation, P.O Box 9800, Melbourne, FL, USA 32902-9800; Dept. of Image Processing Core Technology Center
Email: nick.shorter@harris.com

ABSTRACT
Light Detection And Ranging (LiDAR) platforms utilizing Geiger mode avalanche photodiode (GmAPD) detector arrays are able to realize a higher area of coverage rate (ACR) because of their detectors' single photon sensitivity. The single photon sensitivity comes at the expense of requiring computationally intensive noise filtering techniques. In this work we have implemented a GPGPU accelerated Point Spread Function (PSF) noise filtering algorithm which associates a local likelihood measure with every input point and thus calculates the probability that that point is true target-return photon versus noise. We realize a speed up factor of 30 compared to sequential CPU implementations.

Keywords: Noise Filtering, Geiger Mode, Foliage Penetration, FOPEN, LiDAR, Light Detection and Ranging Data, Laser Radar, LADAR

Problem statement
What is Geiger Mode LiDAR?
Geiger mode avalanche photodiode (GmAPD) detectors are able to operate above critical voltage, and a single photoelectron can initiate the current surge, making the device very sensitive.
What are the advantages?
The sensitivity of the GmAPD array allows for dense feature rich point clouds with direct application to: foliage penetration; urban planning; wide area mapping; flood analysis; feature detection
What are the disadvantages?
- Come at the expense of requiring computationally intensive noise filtering techniques.
- Noise is a problem which affects the imaging system and reduces the capability.
- The sensitivity aggravates the sensor noise characteristics, giving on the order of hundreds of millions measurements with approximately a 1-to-10 signal-to-noise ratio in a typical airborne collect.

What is the solution?
A scalable, fast noise filtering solution is necessary to make GmAPD LiDAR feasible for real-time applications and to alleviate the big data problem of noise filtering wide area mapping collections.

Harris Solution
Harris' Point Spread Function (PSF) filter algorithm is a local spatial measure that has been GPGPU accelerated.
- We apply a kernel density estimation technique for point clustering.
- We associate a local likelihood measure with every point of the input data capturing the probability that a 3D point is true target-return photons or noise (background photons, dark-current).
- Our application is designed for GPU hardware and allows a much higher sampling intensity while decreasing runtime by ~80x.

Spatial Weight Calculation
Point Spread Function (PSF3D) convolution
- Given point cloud data set and 3D Point Spread Function
- Calculate an integrated probability field
- Sample at regular spatial interval
- Determine the elevation where the field is strongest

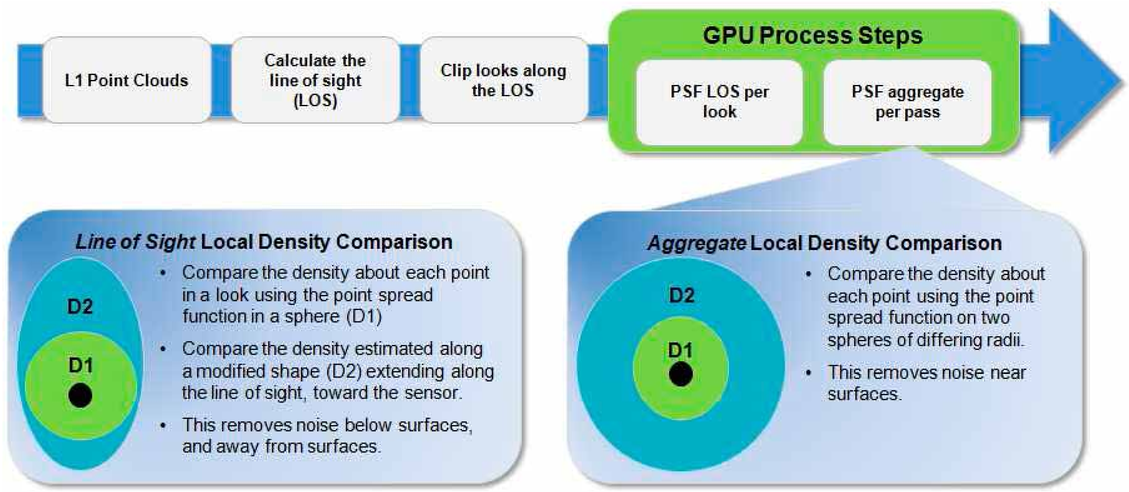
(x, y) : Spatial Interval $\quad \{(x, y, z) : (x_i, y_i, z_i), 1 \le i \le N\}$
z : Elevation
F : Probability Field $\quad F(x, y, z) = \sum_{i=1}^{N} PSF_{3D}(x - x_i, y - y_i, z - z_i)$

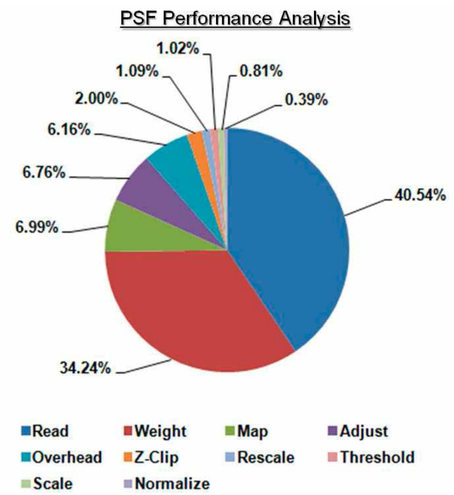
$$PSF_{3D}(x, y, z) = \frac{H(\Delta t, p)}{(2\Pi)^{\frac{3}{2}} \alpha\beta\gamma} e^{-\frac{x^2}{2\alpha^2}} e^{-\frac{y^2}{2\beta^2}} e^{-\frac{z^2}{2\gamma^2}}$$

$$\{(x, y, z) : x_{\min} \le x \le x_{\max}, y_{\min} \le y \le y_{\max}, z : F(x, y, z) = \max F(x, y, \zeta), z_{\min} \le \zeta \le z_{\max}\}$$

Processing Steps
Look level filtering
- Initial step is to remove obvious noise caused by atmospheric backscatter, background radiation, and all End-of-Range (EoR) noise by utilizing a clipping scheme.
- Line of sight transformation
- Open air noise filter signal detection
- GPGPU memory organization
- Primary spatial weight calculation
- Determine optimal threshold

Pass level filtering
- Aggregate looks
- Secondary spatial weight calculation





Look Level Filtering
- Applied to all the looks within a particular pass.
- Assign two weights to every point by statistical calculations of local neighborhood.
- A weight "PSF weight" is estimated by a three – dimensional Gaussian approximation.
- A "stretch weight" considers the spread of points within the local neighborhood.

Table of Signal-to-Noise

Look #	Number of Raw Points	Signal Points	SNR (Raw/S)
1	15749344	3756182	0.153
2	15799257	3867669	0.075
3	15853390	4051528	0.044
4	16007695	4158681	0.044
5	16157085	4310637	0.157
6	16298459	4375405	0.047
7	16307731	4400346	0.175
8	16434910	4441436	0.113
9	16570346	4463870	0.049
10	16565881	4357997	0.124

PSF weight: $w_i = \ln\left(1 + \sum_{p_j \in N_i} \exp\left\{-\frac{1}{2}\left[\left(\frac{x_j - x_i}{\alpha}\right)^2 + \left(\frac{y_j - y_i}{\beta}\right)^2 + \left(\frac{z_j - z_i}{\gamma}\right)^2\right]\right\}\right)$

Stretch weight: $\tilde{w}_i = \sum_{p_j \in N_i} \exp\left\{-\frac{1}{2}\left[\left(\frac{x_j - x_i}{\alpha}\right)^2 + \left(\frac{y_j - y_i}{\beta}\right)^2 + \left(\frac{\Delta z}{\gamma}\right)^2\right]\right\}$

$$\Delta z = \begin{cases} \frac{1}{10}(z_i - z_j), & (z_i - z_j) < 0 \ \text{(the neighborhood point is } \textit{below} \text{ the reference)} \\ 10(z_i - z_j), & (z_i - z_j) \ge 0 \ \text{(the neighborhood point is } \textit{above} \text{ the reference)} \end{cases}$$

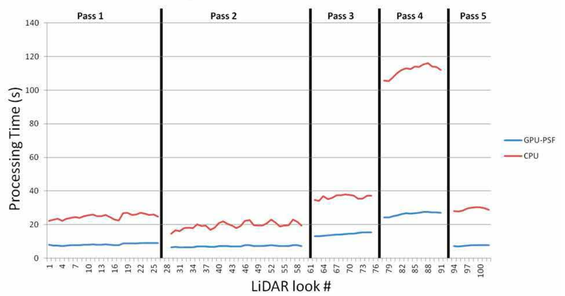
Aggregate filtering
- Merged individual filtered looks to create a pass
- Each point is assigned an additional weight called the "aggregate weight".
- Each point is then projected onto the look's normalization map.
- Maximum aggregate weight of all points is then determined.
- The Gaussian kernel calculation is computed

Table of overall signal-to-noise ratio on a selection of consolidated passed

Pass #	#Looks	SNR Mean	SNR Standard Deviation
1	26	0.107	0.134
2	31	0.113	0.311
3	14	0.109	0.221
4	14	0.910	0.091
5	9	0.577	0.090

Aggregate Weight: $w_i' \rightarrow \frac{w_i'}{\sqrt{1 + \frac{\overline{w}_i}{\overline{w}^{\max}}}}$

PSF Performance Analysis
- 5 passes selected for analysis with a variation in the number of looks.
- This results in a mixture of the number of input raw points which allows us to measure the performance of the algorithm as a function of number of input points.
- The number of output points will be determined by sensor characteristics and collection factors which contribute to the noise and density.
- The goal is to improve the end-to-end process latency of the PSF algorithm.
- Timing analysis was performed on the different stages of the process.
  - **Note:** times were measured using *clock()* in standard C.



GPU Technology in Defense And Space
- Modern Graphics Processing Units (GPGPUs) technology has proven to be a powerful and efficient computational platform.
- Many applications demand faster processing and less power at a lower cost.
- To achieve high performance throughput for our LiDAR filtering algorithm, software and hardware optimization techniques were employed to reduce computational complexity.
- For better optimization it is importation to design software with the subsequent hardware parallelization in mind.
- The hardware parallelization is implemented using the CUDA enabled GPU programming language.

Summary
- More recently, massive volumes of LiDAR point data have become available due to the improvement of laser scanning technology.
- This has posed an unprecedented challenge to the community to provide filtering algorithms to process large volumes of point cloud data.
- We have presented a massively parallel GPU solution for dynamic point cloud data sets capable of achieving near real-time performance. The results shown outperform a similar CPU implementation.
- There are several directions for improvement and future investigation.
  - Provide multi-GPU support in order to utilize the full processing power of available GPUs, and not being limited to just one.
  - Improve the ability to more efficiently handle large data sets, through more efficient memory usage or the use of stream computing.
  - Apply our solution to different problems, such as point cloud modeling and target detection.