

Review of Computer Engineering Research

2024 Vol. 11, No. 2, pp. 73-84

ISSN(e): 2410-9142

ISSN(p): 2412-4281


DOI: 10.18488/76.v11i2.3778


© 2024 Conscientia Beam. All Rights Reserved.



Time optimization for simulation of PMD 3D camera

 **Sangita Gautam Lade^{1*}**

 **Sanjesh Pawale²**

 **Aniket Patil³**

^{1,2}Department of Computer Engineering, Vishwakarma University, Pune, Maharashtra, India.

¹Email: sangita.lade-966@vupune.ac.in

²Email: sanjesh.pawale@vupune.ac.in

³ifm engineering pvt ltd, Maharashtra, India.

³Email: Aniket.patil@ifm.com



(+ Corresponding author)

ABSTRACT

Article History

Received: 13 December 2023

Revised: 29 April 2024

Accepted: 16 May 2024

Published: 20 June 2024

Keywords

3D camera
Graphics processing unit
Power calculation
Ray tracing
Simulation
Speed up
Radiance
Irradiance.

This research aims to enhance the efficiency of simulating 3D cameras utilizing Photonic Mixer Devices (PMD) technology, crucial for applications in computer vision, robotics, and augmented reality. Despite their significance, the computational demands of simulating PMD 3D cameras present substantial challenges in time and resource management. This study proposes a novel approach to optimizing simulation time without sacrificing accuracy, achieved through advanced algorithms and parallel computing techniques. Through a comprehensive analysis of existing simulation methodologies, bottlenecks are identified, and tailored optimization techniques are implemented. The system is designed to simulate PMD sensors, wherein ray tracing precedes power calculation, essential for determining pixel radiance and irradiance. However, the inherent computational intensity of the sequential power calculation algorithm presents a challenge of speed, particularly for PMD sensor simulation reliant on fast-imaging technology. To address this issue, a parallel algorithm leveraging General Purpose Graphics Processing Units (GP GPUs) is proposed and implemented. Experimentation is carried out on Volta (GV100) Graphics Processing Unit (GPU) with varying block sizes from 32 to 1024 in the multiples of 32. Experimental results demonstrate significant speed enhancements, with a maximum speed up of 78% utilizing Volta GPU with a block size of 1024, thereby showcasing the efficacy of the proposed methodology.

Contribution/Originality: In power calculation, a hybrid strategy is implemented, integrating aspects of both the divide-and-conquer model and the pipeline model of the parallel algorithm's design. By combining these two approaches, the algorithm achieves improved efficiency and scalability in parallel processing, effectively leveraging the strengths of each model.

1. INTRODUCTION

The introduction of three-dimensional (3D) imaging technologies has sparked a profound transformation across multiple domains, including computer vision, robotics, and augmented reality. Within this landscape, Photonic Mixer Device (PMD) 3D cameras have risen as indispensable tools, offering unprecedented accuracy and speed in capturing depth information. Yet, the widespread adoption of PMD cameras across various applications underscores the need for precise simulation tools for both development and testing purposes. A significant obstacle in this endeavor lies in the computational demands associated with simulating PMD 3D cameras, frequently impeding real-time or time-critical applications.

This research addresses the imperative task of optimizing the simulation time for PMD 3D cameras, aiming to strike a balance between computational efficiency and high-fidelity results. The significance of this endeavor lies in the broad spectrum of applications relying on 3D sensing technologies, where the ability to rapidly prototype and simulate PMD-based systems is crucial for innovation and deployment.

The introduction delves into the evolving landscape of 3D imaging technologies, highlighting the indispensability of PMD 3D cameras and the challenges posed by their computationally intensive simulations. It sets the stage for the subsequent sections, outlining the objectives, methodologies, and expected contributions of the research in optimizing the simulation time for PMD 3D cameras. As the digital world continues to demand faster and more accurate simulations, the outcomes of this study hold promise for accelerating the development and integration of PMD-based systems across various domains.

Today, many fields need image processing systems. The intensity of the optical image in each pixel is detected by a charged coupled device (CCD) camera, while Complementary Metal-Oxide-Semiconductor (CMOS) camera detects the projection of real scenery. Many control and navigation applications require knowledge of three-dimensional data. Obtaining in-depth information has become a main requirement for fast and reliable developments in the industrial and automotive environments in the future [1]. There are many 3D sensors and cameras available on the market. Photonic Mixer Device (PMD) sensors are also used to generate the 3D data. These sensors utilize Time of Flight (TOF) technology. This TOF technology delivers speedy images faster with the depth information needed for real-time applications. This new technology works without complex electronics. It detects both the intensity and the distance in each PMD pixel or voxel, respectively [1].

The manufacturing of a new 3D camera is an expensive process. So, before manufacturing, a software simulator can be designed and implemented. Building prototypes is very expensive, so they can be avoided using simulation. The simulator can fix the sensor characteristics without building and testing the prototypes [2].

Simulation software is employed to generate a prototype for a tangible product, replicating the fundamental traits of the sensor. This tool proves highly beneficial across diverse scenarios. It enables easy modification of sensor parameters, allowing for flexibility and adaptability. The simulation environment allows for the consistent conduct of a range of experiments, thereby simplifying the creation of dynamic scenes. Consequently, the simulation yields results that accurately mirror real sensor data, offering a reliable basis for comparison [3].

2. LITERATURE REVIEW

Photonic Mixer Devices (PMD) 3D cameras have gained prominence in recent years for their ability to capture depth information with remarkable accuracy and speed. As these cameras become integral components in various technological applications, the demand for efficient simulation tools has intensified. The simulation of PMD 3D cameras involves intricate computational processes, often requiring substantial time and resources. This literature review explores the current state of research in the simulation of PMD 3D cameras, with a specific focus on efforts to optimize simulation time while preserving accuracy.

2.1. PMD 3D Camera Technology

Understanding the fundamental principles of PMD 3D camera technology is crucial for effective simulation. The literature offers comprehensive insights into the working mechanisms, sensor architectures, and the unique advantages of PMD cameras in capturing, in-depth information. These foundational studies provide the basis for subsequent simulation efforts.

2.2. Simulation Techniques and Challenges

Researchers have proposed a wide range of simulation techniques for PMD 3D cameras, from ray tracing to numerical simulations. Existing literature highlights the complexities involved in accurately replicating the

behavior of PMD sensors, taking into account factors such as light propagation, sensor response, and environmental conditions. However, the consensus is that these simulations often come at a significant computational cost.

2.3. Computational Optimization in Simulation

Recognizing the computational challenges, recent research has explored various strategies for optimizing the simulation of PMD 3D cameras. Parallel processing, Graphics Processing Unit (GPU) acceleration, and algorithmic enhancements are among the approaches investigated to reduce simulation time without compromising precision. The literature reveals a growing interest in harnessing the capabilities of modern computing architectures to enhance the efficiency of simulation workflows.

2.4. Real Time and Time sensitive Applications

The urgency surrounding real-time or time-critical applications like robotics and augmented reality highlights the necessity for swift simulation of PMD 3D cameras. Research in this field explores the impact of simulation speed on the viability and efficiency of applications, emphasizing the crucial role of optimizing simulation time for seamless deployment in practical scenarios.

2.5. Benchmarking and Evaluation Metrics

Evaluating the performance of simulation techniques requires standardized benchmarks and metrics. Existing literature discusses methodologies for benchmarking PMD 3D camera simulations, considering factors such as accuracy, computational efficiency, and scalability. These benchmarks serve as reference points for assessing the efficacy of time optimization strategies.

2.6. Future Directions and Challenges

While advancing camera features, “Michal Kucis “presented simulation of camera imperfections to get computer-generated images with features that are close to the features of images captured by real cameras. Current version of the simulator has been for PMD 3D cameras. The literature acknowledges ongoing challenges and identifies avenues for future research. This includes addressing the trade-off between simulation speed and accuracy, exploring emerging technologies, and adapting optimization, techniques to evolving hardware architectures.

Head of BU Systems, Dr.-Ing. Thorsten Ringbeck, described the design of PMD cameras in [Ringbeck \[2\]](#). These cameras provide 3D information about the captured scene. Many customized camera designs are presented in the paper. But power calculation is not used for reproducing a 3D data. So, simulation of such 3D camera is required to reproduce essential sensor characteristics.

“Simulation is capable of processing only 0.9 frames per second. The test was performed with use of a dual-core Intel Core with 2GHz frequency on an image with a resolution 640x480 [\[4\]](#). This process of simulation needs optimization. Also, only few features are used, and power calculation is not done in the simulation.

Paper [Peters, et al. \[5\]](#) present a software-based simulation approach for a 3D camera. This system allows modifying many static and dynamic parameters of the sensor, but it doesn't consider the power calculation for each pixel.

The accuracy of a smartphone camera simulation is assessed by [Zheng, et al. \[6\]](#). End-to-end analysis is done in this simulation. The analysis starts with a high dynamic range three-dimensional scene physical description. We compare the data with the actual physical scene. The method is found to be very effective and accurate, as the results match many parameters of the sensor.

In their paper, [Zhang, et al. \[7\]](#) have simulated three-dimensional technique on GPU. They achieved 70x to 75x speedup using Nvidia GTX260+ GPU over parallel version of the Intel Q6600 CPU.

Remo Gygax et al. have provided a parallel approach for numerical simulation on GPUs. The NVIDIA Compute Unified Device Architecture (CUDA) GPU was used for parallel processing which has increased the speed of the simulation and the accuracy of the prediction. They wrote an easily adaptable and short code using CUDA and achieved higher resolution and better processing speed [8].

Zhang, et al. [9] have developed a three-dimensional simulation program for the core shooting process. They used a graphics processing unit (GPU) to improve the calculation efficiency. CUDA was used for parallel processing, and the accuracy was tested with high-speed camera. The speed-up achieved was 95% without affecting the simulation results.

3. RESEARCH PROBLEM & SYSTEM ARCHITECTURE

This paper presents a parallel algorithm that mimics PMD 3D cameras. The experimentation phase focuses on utilizing a Picoflex camera with a resolution of 38K.

The simulation task at hand poses a significant computational challenge when employing a sequential algorithm, especially when executed on a single processing unit. We have introduced a parallel algorithm as an alternative approach to better handle the computational demands and address this issue.

3.1. Sequential Algorithm Intrinsically Computationally Intensive

The term "intrinsically computationally intensive" indicates that when executed sequentially, the nature of simulation task necessitates significant computational resources and time. This may be due to the complexity of the simulation, which involves intricate calculations or large datasets.

3.2. Motivation for Parallelization –Computational Intensity

The decision to propose a parallel algorithm is motivated by the recognition that the computational demands of the simulation are high. In many cases, traditional sequential algorithms struggle to handle such intensity efficiently within a reasonable timeframe.

3.3. Parallel Algorithm Proposal –Addressing Computational Bottlenecks

Parallel algorithms are designed to address computational bottlenecks by distributing the workload across multiple processing units simultaneously. Tasks with inherent parallelism, where different parts of the computation can perform independently, particularly benefit from this.

3.4. Parallelization Strategies

- a. Dividing the Workload: Typically, parallel algorithms divide the overall computation into smaller, independent tasks that can run concurrently.
- b. Utilizing Multiple Processing Units: The proposed parallel algorithm likely leverages the capabilities of multi-core processors or, as indicated in a previous context, GPUs for parallel execution.
- c. Optimizing Resource Utilization: By parallelizing the algorithm, it becomes possible to make more efficient use of available computational resources, reducing the overall time required to complete the simulation.

3.5. Trade-Offs and Considerations –Concurrency Vs Overhead

While parallelization can significantly improve computational speed, it introduces overhead related to managing parallel threads and communication. The design of the parallel algorithm should carefully balance concurrency benefits against this overhead.

3.6. Performance Improvement

The primary goal of proposing a parallel algorithm is likely to reduce the simulation's overall execution time. By efficiently utilizing multiple processing units, the parallel algorithm aims to achieve better performance compared to its sequential counterpart.

The simulation of 3D camera is demonstrated in [Figure 1](#). It comprises steps like scene generation using ray tracing, power calculation, raw data generation and raw data processing. The algorithm for scene generation is already presented in [Lade, et al. \[10\]](#). The sensor grid used for the picoflex camera is represented by a 3D matrix of $172 \times 224 \times 3$. So, all the calculations involve the 3D matrix operations.

The output of ray tracing [\[10\]](#) is scene generation, which is given as an input to the next step, i.e., power calculation. The input to the ray tracing is the number of shapes (object list), their corresponding equation values, origin, speed, reflectivity, and sensor pixel coordinates, while the output is distance matrix generated for given object list, reflectivity matrix generated for the object list, intersection matrix, and normal matrix.



Figure 1. Simulation process of PMD sensor.

4. POWER CALCULATION

Power calculation is critical when simulating 3D Photonic Mixer Device (PMD) cameras, as it directly influences the computational resources required for accurate and efficient simulations. Accurate simulations are essential for the development, testing, and optimization of PMD camera systems. Understanding and managing the power requirements of these simulations are pivotal for ensuring that the computational demands align with available resources. The following aspects contribute to the power calculation considerations in the simulation of 3D PMD cameras.

4.1. Algorithmic Complexity

The choice of simulation algorithms significantly impacts the computational workload. Complex algorithms, such as those involving detailed ray tracing or sophisticated signal processing techniques for PMD cameras, may demand substantial computational power. Power calculations need to account for the algorithmic intricacies and evaluate potential optimizations to strike a balance between accuracy and efficiency.

4.2. Parallelization and Multi-Core Processing

Power-efficient parallelization and multi-core processing play a crucial role in optimizing simulation performance. Parallelizing computations across multiple cores or utilizing graphics processing units (GPUs) can enhance simulation speed while potentially reducing overall power consumption. Power calculations should consider the trade-offs between the computational gains achieved through parallelization and the additional power required by parallel processing units.

4.3. Memory Requirements

Simulation of PMD cameras often involves the manipulation and storage of large datasets, including 3D point clouds and sensor response matrices. Efficient memory management strategies are essential to avoid unnecessary power consumption related to data transfer and storage. Power calculations should take into account the memory requirements and assess strategies such as data compression or streaming to minimize energy usage.

4.4. Hardware Acceleration

Leveraging specialized hardware, such as Field-Programmable Gate Arrays (FPGAs) or application-specific integrated circuits (ASICs), can provide significant power advantages for specific simulation tasks. Power calculations should consider the feasibility and benefits of utilizing hardware acceleration for critical components of the simulation workflow.

4.5. Dynamic Resource Allocation

At various stages, dynamic resource allocation entails adjusting computational resources based on the simulation's specific needs. Power calculations should explore adaptive algorithms that dynamically allocate resources based on the simulation's current workload, potentially enabling power savings during less computationally intensive phases.

4.6. Energy Efficient Simulation Environment

Implementing energy-efficient computing environments and adopting power-aware programming practices can contribute to minimizing the overall power consumption during simulation. Power calculations should account for the overhead associated with creating and maintaining an energy-efficient simulation environment.

4.7. Real Time Simulation Requirements

If the simulation is intended for real-time applications, power calculations should consider the feasibility of meeting real-time constraints while maintaining an acceptable level of accuracy. This involves understanding the power consumption patterns associated with achieving real-time simulation rates.

So, power calculation for simulating 3D PMD cameras is a multifaceted consideration that involves a careful balance between computational accuracy and efficiency. Optimizing power consumption in simulations is pivotal for facilitating practical and sustainable implementation of PMD camera technologies across various applications.

Power calculation is an application of the Phong Lighting Model that provides sensors to compute the energy radiated by objects in a scene when rays of light fall upon them. Ray tracing begins with sensor data and a single light ray which traces its straight-line path and goes through successive collisions with objects to give a layout of what is present in the scene. The primary aim is to find intersections of the ray with the objects in the scene and have a numerical layout with which the sensors learn automatically about the position of the objects in the scene.

Power calculation is the successor to ray tracing, which gives intermittent values such as proximity and radiance from the scene in consideration. Lights and objects in the scene create several types of three-dimensional effects since illumination is necessary. Disposing of the light is required for tracing a raw image of realistic three-dimensional objects. Phong model works similar to that of realistic lighting model [4]. In this paper, Phong model is used to compute power, radiance, irradiance, and light source length, and a parallel algorithm is implemented to speed up calculations.

Radiance is the power that a focal point transmits. When the ray hits the object, some power gets lost, which is proportional to $\cos(\alpha)$. This power received by an object is called irradiance. Now the ray will be reflected and hit the sensor grid. So, the power received at object becomes radiance and when it reaches the sensor grid, it becomes irradiance. As shown in Figure 2, a ray can be sent from a light source, and it can hit an object 2 in the scene. Irradiance is the power that objects 2 receives. When this ray is reflected, on its way to sensor grid there can be other objects placed, like object 1, so the ray gets hit to object 1 and then reflected back to sensor grid. The sensor grid calculates the power for each pixel.

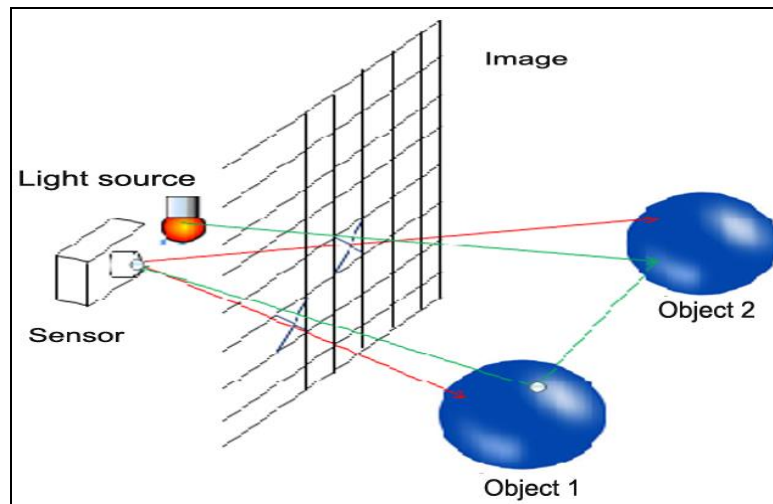


Figure 2. Power calculation.

5. PARALLEL ALGORITHM FOR POWER CALCULATION

The algorithm is tested on picoflex camera with a 38K resolution. Now there are 172 X 224 pixels for calculation, so this task of calculation can be done parallelly with the help of GPU. So a parallel algorithm for the same is proposed and shown in Figure 3.

```

Function PowerCalculation(parameterList):
  Define Light_Origin = (15.47, 0, 0)
  Define offsetX = 42
  Define offsetY = 43
  Define nx = 172
  Define ny = 168
  Define ff = parameterList.FillFactor
  Define PixelHeight = parameterList.PixelHeight
  Define PixelWidth = parameterList.PixelWidth
  Define FNumber = parameterList.FNumber

  Define Light_Vector
  Define Inclination
  Define Azimuth
  Define xScale
  Define yScale

  Define intensity_profile_matrix[ny][nx]
  Define L
  Define E
  Define P

  For each pixel in parameterList.Camera.Pixels:
    Light_Vector = Normalize(Light_Origin - pixel.RayObjectIntersectionPoint)
    Inclination = arctan(Light_Vector.z / sqrt(Light_Vector.x^2 + Light_Vector.y^2))
    Azimuth = arctan(Light_Vector.x / Light_Vector.y)
    xScale = nx / (2 * offsetX)
    yScale = ny / (2 * offsetY)
    rowIndex = (Inclination * Sin(Azimuth) * 180 * yScale) / Pi
    colIndex = (Inclination * Cos(Azimuth) * 180 * xScale) / Pi
    intensity = intensity_profile_matrix[rowIndex][colIndex]

    L = (parameterList.Reflectivity * intensity * Cos(theta)) / (Pi * r^2)
    E = (L * Pi * (1 / FNumber)^2 * Cos(alpha)^4) / (4 * f^2)
    P += E * PixelHeight * PixelWidth * ff

  Return P

```

Figure 3. Algorithm for power calculation.

The algorithm in the form of flowchart for power calculation is presented in Figure 4. As Compute Unified Device Architecture (CUDA) is multi-threaded architecture, we can schedule many threads on the same number of cores to perform the same task. So, the tasks that can be scheduled in parallel are shown in the Figure 4. We write a kernel specifically for this purpose. After power calculation, raw data generation takes place, followed by raw data processing.

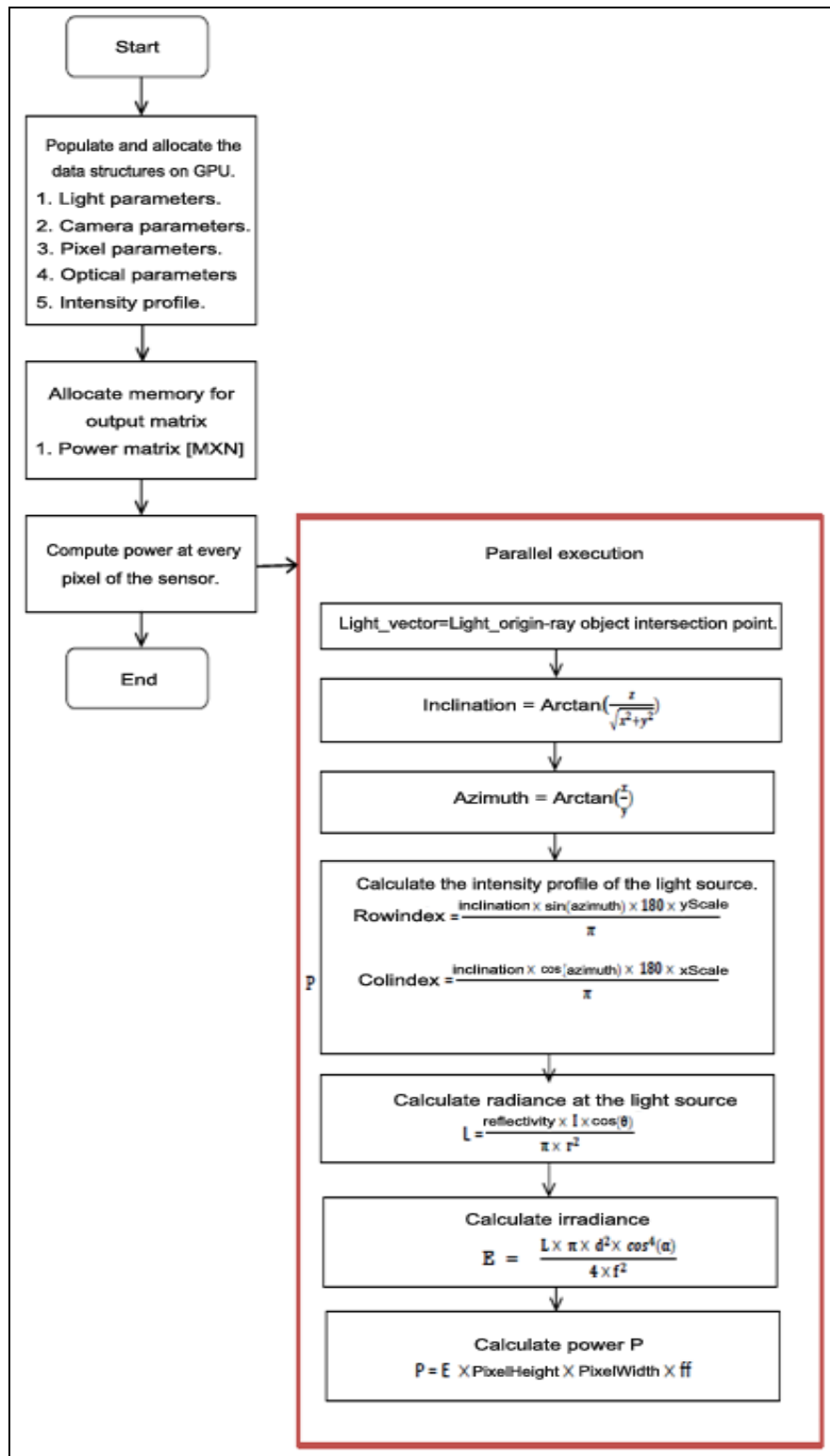


Figure 4. Flowchart for parallel algorithm for power calculation.

6. RAW DATA GENERATION & CORELATION

To simulate the performance of a 3D Photonic Mixer Device (PMD) camera, you need to create raw data that mirrors how the sensors reacts to light and then use correlation processes to get accurate depth data. This process is crucial for testing and refining the functionality of PMD cameras in diverse applications. Here, we delve into the methodologies of raw data generation and correlation in the context of simulating a 3D PMD camera.

6.1. Raw Data Generation

- a. **Light Propagation Simulation:** Simulating the interaction of light with the scene involves modeling the propagation of light rays from the source to the PMD sensor. Algorithms, such as ray tracing or Monte Carlo simulations, are commonly employed to generate realistic light paths and interactions.
- b. **Sensor Response Modeling:** Accurate raw data generation requires modeling the response of the PMD sensor to incident light. This involves simulating the sensor's characteristics, such as pixel sensitivity, quantum efficiency, and temporal response. Detailed sensor models ensure that the raw data closely resembles the real-world response of the PMD camera.
- c. **Time of Flight Simulation:** PMD cameras frequently function using the time-of-flight principle, which measure the duration of time travel from the source to the object and back. Raw data generation includes simulating this time-of-flight information, considering factors like signal noise, jitter, and temporal resolution.
- d. **Depth Map Generation:** The raw data should be processed to generate a depth map, representing the distance of each pixel from the camera. Algorithms such as phase-shift or amplitude modulation are employed to correlate the time-of-flight information and convert it into accurate depth measurements.

6.2. Correlation Techniques

- a. **Time of Flight Correlation:** Correlating the time-of-flight information is essential for deriving accurate depth maps. This involves analyzing the phase or amplitude information of the received signals and correlating it with the transmitted signals. Sophisticated correlation algorithms help extract precise depth information from the raw data.
- b. **Noise Reduction and Filtering:** Raw data from PMD cameras can be prone to noise and artifacts. Correlation techniques include using filtering algorithms to reduce noise and improve the accuracy of depth maps. People commonly employ techniques like Gaussian filtering algorithms to reduce noise and improve the accuracy of depth maps.
- c. **Calibration and Registration:** Correlating raw data also involves calibrating the simulated PMD camera to match real-world calibration parameters. This includes accounting for intrinsic and extrinsic camera parameters, lens distortions, and geometric transformations. Calibration ensures that the simulated data correlates accurately with physical measurements.
- d. **Cross Validation with Real Data:** To enhance the reliability of the simulated results, correlation techniques may involve cross-validation with real-world data. This step helps validate the accuracy of the simulated PMD camera against empirical measurements, refining the simulation parameters if necessary.

Ambient light refers to general illumination which is not directly caused by light source but is present in the environment. The presence of ambience is the result of multiple reflections and re-reflections of rays. Hence, we take into consideration the ambient light in the environment, as it can't be ignored completely.

Correlation is calculating the mutual relationship between two waves, whether they are in phase or out of phase. In this case, the correlation is calculated between 2 different sets of waves. The common wave in these 2 sets is the light wave, which is bounced back to the PMD chip's sensor from the scene. One set has a modulated signal that reaches channel A of the PMD chip, the other set has a modulated signal that reaches channel B of the chip. Both of these waves have a phase difference of 180 degrees. Each wave, being a square wave can be represented in the form of a Fourier transform [11].

There are various methods used for interpolation. Cubic spline is most widely used [12]. The earlier implementation of spline was showing a large deviation from the expected output. We found out that the reason for such a large deviation was different approaches used for interpolation. The pre-existing method of interpolation did not consider boundary conditions; hence, we implemented not-a-knot conditions. You can find the output in the Results section. Earlier, there was large deviation in the final values of ambient electrons, which was resolved to a

great extent by rectifying the units of various input parameters that are being used in the calculation and by modifying the interpolation algorithm to include boundary conditions.

7. ALGORITHM ANALYSIS

The simulation process of 3D camera is parallelized using GPU. The parallel algorithm for power calculation is implemented on GeForce Giga Texel Shader eXtreme (GTX) 950M and Volta GV100. Both of these architectures are based on Fermi architecture [13]. Table 1 presents the specifications of both the GPUs, GeForce GTX 950M and Volta GV100, including their architecture, performance, CUDA cores, memory capacity, power consumption, etc.

Table 1. GPU specifications.

Parameter	GeForce GTX 950M	Volta V100
Architecture	The GTX 950M is based on the Maxwell architecture. Maxwell was NVIDIA's architecture released in 2014 and focused on power efficiency and performance-per-watt	The V100, on the other hand, is based on the Volta architecture. Volta is a more recent and advanced architecture, designed for high-performance computing and artificial intelligence workloads
Performance	The GTX 950M is a mid-range mobile GPU intended for laptops. It is suitable for gaming and multimedia tasks but is not designed for high-performance computing	The V100 is a high-end GPU designed for data center and professional applications. It is part of NVIDIA's Tesla lineup and is known for its exceptional performance in scientific computing, deep learning and AI workloads
CUDA cores	The GTX 950M has a limited number of CUDA cores compared to GPUs designed for high-performance computing	The V100 has a significantly higher number of CUDA cores, reflecting its design for parallel processing and handling complex computational tasks
No. of CUDA cores	640	5120
Memory	The GTX 950M typically comes with GDDR5 memory with a smaller capacity suitable for gaming and multimedia	The V100 is equipped with HBM2 (High bandwidth memory) with a significantly larger memory capacity. This type of memory is designed to provide high bandwidth and is well-suited for memory-intensive workloads
Memory capacity	2 GB	16 GB
Manufacturing process	The GTX 950M is manufactured using a 28nm process technology	The V100 is built on a more advanced 12nm process technology, contributing to improved energy efficiency and performance
Memory bus	128 bit	4096 bit
Power consumption	60W	300 W

7.1. GPU Selection

The first mobile GPU based on the Maxwell architecture, which is commonly found in laptops and is suitable for mid-range gaming and multimedia tasks, is selected for experimentation. Another GPU that belongs to NVIDIA's high-end Tesla lineup, designed for data center and high-performance computing applications. It features the Volta architecture, known for its advancements in parallel processing, and AI workloads are selected for experimentation.

7.2. Experimentation with Block Size

a. Block Size variation: The block size is a crucial parameter in parallel processing, often associated with GPU programming using technologies like CUDA. Experimenting with block sizes from 32 to 1024 indicates an exploration of the optimal configuration for parallel execution.

b. Parallel Processing: GPUs excel in parallel tasks, dividing computational workloads into smaller blocks for simultaneous processing. Varying the block size allows for finding the most efficient configuration for the specific power calculation task.

7.3. Objects Considered

1. Sphere, box, plane and triangle: These objects represent different geometries that might be encountered in 3D simulations or visualizations. Considering various objects is important for assessing the generalizability and performance of the power calculation algorithm across different scenarios.

2. Reduction in Time Taken: Using the GPU significantly reduces the time required for power calculation compared to the CPU. GPUs excel at parallel processing, allowing for faster execution of tasks that can be parallelized.

3. Implications of Results: The GPU's shorter power calculation time may make real-time or near-real-time simulations more feasible. This efficiency gain has implications for the overall performance and responsiveness of the simulation, especially in scenarios involving complex 3D objects.

4. Practical Applications: The reduced time for power calculation on the GPU may make real-time or near-real-time simulations more feasible. Alternatively, it could enable the processing of larger batches of data in a shorter timeframe, depending on the specific application requirements.

5. The speedup of GPU over CPU is depicted in Figure 5. Also, the frame generation time including all 9 phases is 0.1569 seconds using GPU which is very small as compared to frame generation time of 6 seconds as recorded by Pasinetti, et al. [14].

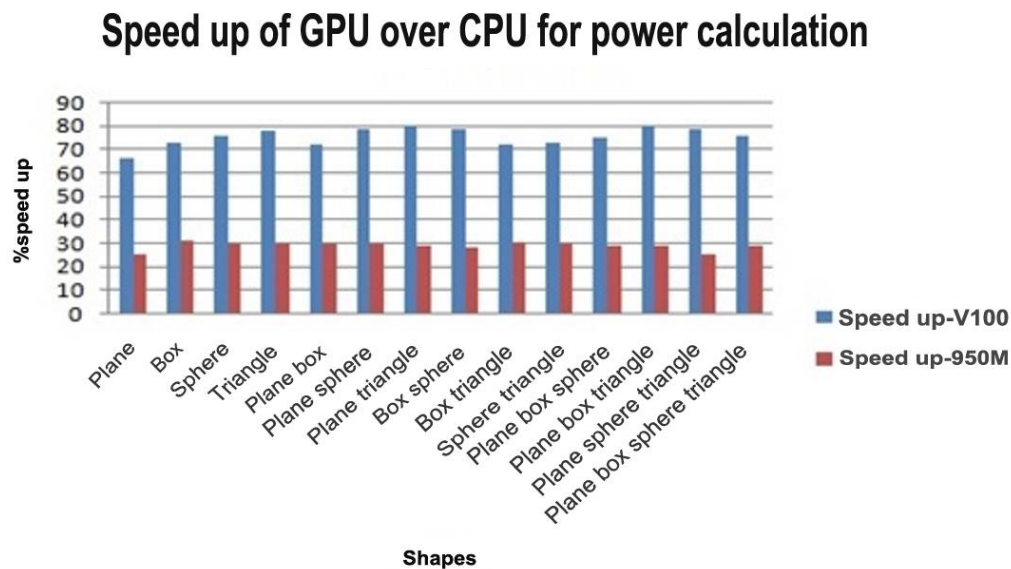


Figure 5. Speed up of GPU over CPU.

In summary, the experimentation on the GeForce GTX 950M and Volta V100 involves optimizing power calculation through the exploration of different block sizes. The utilization of GPUs, known for their parallel processing capabilities, results in a significant reduction in the time required for power calculation compared to using the CPU. This finding has implications for the overall efficiency and performance of simulations involving 3D objects, showcasing the advantages of leveraging GPU acceleration for such computational tasks.

8. CONCLUSION & FUTURE WORK

Power calculation is very crucial while simulating PMD sensors. It is independent of the placement of objects in the scene. Using Volta GPU, a minimum speed increase of 75 % and a maximum increase of 78% are achieved by using a block size of 1024 for power calculation. The average frame generation time, including all 9 phases, is 0.1569 seconds. The hybrid models of parallel algorithm, i.e., combination of divide and conquer and pipeline, work best for power calculation. The ray tracing algorithm is used to generate the scenes using primitive objects whose equations are known. The power calculation is also done for the same scenes. This algorithm can be modified to

generate more complex and dynamic scenes whose equations are unknown. Also, the various lighting effects can be simulated, like diffused light, specular light, etc. Various colour effects can also be simulated.

Funding: This study received no specific financial support.

Institutional Review Board Statement: Not applicable.

Transparency: The authors state that the manuscript is honest, truthful, and transparent, that no key aspects of the investigation have been omitted, and that any differences from the study as planned have been clarified. This study followed all writing ethics.

Competing Interests: The authors declare that they have no competing interests.

Authors' Contributions: All authors contributed equally to the conception and design of the study. All authors have read and agreed to the published version of the manuscript.

REFERENCES

- [1] H. Kraft *et al.*, *3D-camera of high 3D-frame rate, depth-resolution and background light elimination based on improved PMD (photonic mixer device)-technologies*. Nuernberg: OPTO, 2004.
- [2] T. Ringbeck, "A 3D time of flight camera for object detection," presented at the 8th Conference on Optical 3D Measurement Techniques, 2007.
- [3] M. Keller, J. Orthmann, A. Kolb, and V. Peters, "A simulation framework for time-of-flight sensors," presented at the 2007 International Symposium on Signals, Circuits and Systems, IEEE, 2007.
- [4] K. Michal, "Simulation of camera features," in *Proceedings of CESC 2012: The 16th Central European Seminar on Computer Graphics*, 2012.
- [5] V. Peters, O. Loffeld, K. Hartmann, and S. Knedlik, "Modeling and bistatic simulation of a high resolution 3D PMD-camera," in *Proceedings Congress on Modelling and Simulation (EUROSIM)*, 2007.
- [6] L. Zheng, G. Thomas, W. Brian, and F. Joyce, "Accurate smartphone camera simulation using 3D scenes," *arXiv preprint arXiv:2201.07411*, 2022.
- [7] F. Zhang, G. Wang, X. Liu, and J. Liu, "An improved parallel implementation of 3D DRIE simulation on GPU," presented at the 2009 10th International Symposium on Pervasive Systems, Algorithms, and Networks, IEEE, 2009.
- [8] G. Remo, R. Ludovic, O. Samuel, P. Yuri, and J. Michel, "GPU accelerated numerical simulation of viscous flow down a slope," *Geophysical Research Abstracts*, vol. 16, pp. EGU2014-15299, 2014.
- [9] Y.-z. Zhang, G.-c. Lu, C.-j. Ni, T. Jing, L.-l. Yang, and Q.-f. Wu, "GPU based numerical simulation of core shooting process," *China Foundry*, vol. 14, no. 5, pp. 392-397, 2017.
- [10] S. Lade, M. Kulkarni, and A. Patil, *Ray tracing algorithm for scene generation in simulation of photonic mixer device sensors*. In: Swain D., Pattnaik P.K., Atharwale T. (Eds.), *machine learning and information processing, advances in intelligent systems and computing*. Singapore: Springer. https://doi.org/10.1007/978-981-33-4859-2_25, 2021.
- [11] V. C. Tsai, "Understanding the amplitudes of noise correlation measurements," *Journal of Geophysical Research: Solid Earth*, vol. 116, no. B9, p. B09311, 2011. <https://doi.org/10.1029/2011jb008483>
- [12] G. Wolberg, *Cubic spline interpolation: A review*. Department of Computer Science, Columbia University. <https://doi.org/10.7916/D82Z1DMQ>, 1988.
- [13] C. M. Wittenbrink, E. Kilgariff, and A. Prabhu, "Fermi GF100 GPU architecture," *In IEEE Micro*, vol. 31, no. 2, pp. 50-59, 2011. <https://doi.org/10.1109/MM.2011.24>
- [14] S. Pasinetti, M. M. Hassan, J. Eberhardt, M. Lancini, F. Docchio, and G. Sansoni, "Performance analysis of the PMD camboard picoflexx time-of-flight camera for markerless motion capture applications," *IEEE Transactions on Instrumentation and Measurement*, vol. 68, no. 11, pp. 4456-4471, 2019. <https://doi.org/10.1109/tim.2018.2889233>

Views and opinions expressed in this article are the views and opinions of the author(s), Review of Computer Engineering Research shall not be responsible or answerable for any loss, damage or liability etc. caused in relation to/arising out of the use of the content.