Accelerated Impulse Response Calculation for Indoor Optical Communication Channels

M. Rahaim, J. Carruthers, and T.D.C. Little
Department of Electrical and Computer Engineering
Boston University, Boston, MA 02215
{mrahaim, jbc, tdcl}@bu.edu

MCL Technical Report No. 09-01-2012

Abstract

In the modern era of communication systems, wireless data transfer is essential for appealing to the growing user demand for ubiquitous high data rate network connectivity. Increases in the complexity of wireless applications along with the diversification of wireless devices has led to forecasts indicating continued gains in wireless consumer IP traffic [1] and concerns regarding congestion in the RF spectrum. This has opened the door for other means of wireless data transfer including optical media such as Infrared (IR) and Visible Light Communication (VLC). Much of the recent work in optical communications has focused on appropriate modulation techniques. A fast and accurate optical channel model is important when estimating achievable data rate in the analysis of these techniques. Such a model should observe received power as well as the impulse response between transmitter and receiver in order to estimate inter-symbol and inter-channel interference.

In this work, we observe the channel modeling software "Communication and Lighting Environmental Simulation" (CandLES) [2] and more specifically the impulse response simulator carried over from previous work by Carruthers et al. for IR channels – IRSIM [3]. We accelerate the algorithm for complex environments by adding parallel processing and show a performance increase between 5x and 20x, depending on the simulation settings. In the following, we overview the algorithm and original implementation of IRSIM, discuss the accelerated GPU implementation, and compare the performance of each.

Keywords -- Visible Light Communication, Impulse Response, GPU, Parallel Programming.

In *IEEE International Conference on Wireless Information Technology and Systems (ICWITS 2012)*, Maui, HI, United States, November 2012. This material is based upon work support by the National Science Foundation under Grant No. EEC-0812056.

1. Introduction

In the modern era of communication systems, wireless data transfer is essential for appealing to the growing user demand for ubiquitous high data rate network connectivity. Increases in the complexity of wireless applications along with the diversification of wireless devices has led to forecasts indicating continued gains in wireless consumer IP traffic [1] and concerns regarding congestion in the RF spectrum. This has opened the door for other means of wireless data transfer including optical media such as Infrared (IR) and Visible Light Communication (VLC). Much of the recent work in optical communications has focused on appropriate modulation techniques. A fast and accurate optical channel model is important when estimating achievable data rate in the analysis of these techniques. Such a model should observe received power as well as the impulse response between transmitter and receiver in order to estimate inter-symbol and inter-channel interference.

In this work, we observe the channel modeling software "Communication and Lighting Environmental Simulation" (CandLES) [2] and more specifically the impulse response simulator carried over from previous work by Carruthers et al. for IR channels – IRSIM [3]. We accelerate the algorithm for complex environments by adding parallel processing and show a performance increase between 5x and 20x, depending on the simulation settings. In the following, we overview the algorithm and original implementation of IRSIM, discuss the accelerated GPU implementation, and compare the performance of each.

2. Indoor Optical Channel Model

In this section, we review the optical channel model and core algorithm behind the IRSIM software. Due to directionality, path loss in an optical channel is modeled differently than that of an RF channel. Angle of transmission as well as angle of acceptance become a factor in the received electrical signal, therefore additional parameters get added to the system. As a general model, the line of sight (LOS) received power, P_r, at destination j from source i is modeled as

$$P_r = P_t T(\phi_{ij}) A_r g(\theta_{ij}) / {D_{ij}}^2 \qquad\qquad T(\phi) = \frac{n+1}{2\pi} cos^n(\phi)$$

where P_t is the transmitted optical power, A_r is the area of the destination, D_{ij} is the distance between source and destination, ϕ_{ij} and θ_{ij} represent angles of emission and acceptance, and the functions $T(\phi)$ and $g(\theta)$ represent the sources intensity pattern and the destinations optical gain function, respectively. The intensity pattern of an LED is modeled as Lambertian with order n and the optical gain function for a bare photodiode is observed as $g(\theta) = cos(\theta)$. In addition, the delay from source to destination is observed as $T_{ij} = D_{ij}/c$, where c denotes the speed of light. In the model, all transmitters, receivers and reflectors are observed as point sources.

Rather than observing a time step simulation and maintaining every path – increasing overhead and complexity with each reflection – the algorithm developed in [3] uses an event driven simulation to model the received signal depending on the number of reflections in a path from transmitter to receiver. This method benefits from known memory overhead and a calculation time which increases at a constant rate per reflection. As a high level overview, the algorithm follows the five steps listed below:

1. *Zero Bounce (LOS) Calculation*: The zero bounce calculation determines the LOS impulse response (delay and attenuation) at each receiver from the set of transmitters. Note that the attenuation function observes objects in the room and returns 0 if the path is obstructed.
2. *Room Partitioning*: The environment is next partitioned into reflectors – or source / destination pairs. This is done by observing each object and walls of the room as a set of planes. Each plane is divided into a set of reflectors and added to a list. Throughout the remainder of the paper, we refer to the list of reflector received signals as the matrix. This is an m X n matrix where m represents the maximum number of time slots of an impulse response array and n represents the number of reflectors in the environment. Note that m increases with smaller time steps or an increased number of reflections.
3. *Matrix Initialization*: This step of the algorithm calculates delay and attenuation from each transmitter to each reflector and stores the results to the appropriate time slot in the matrix.
4. *Received Power Update*: Given the matrix described above, delay and attenuation from each reflector to each receiver is calculated. The m X 1 reflector array is convolved with the delayed attenuation impulse and added to the appropriate time slots in the receiver impulse response array.
5. *Matrix Update*: The final stage of the algorithm updates the matrix for the next reflection. Attenuation and delay is calculated between each pair of reflectors and the received signal is stored in a temporary matrix. Once completed, the new matrix is used to determine the receiver signals from the next reflection by repeating step 4. These two steps are repeated so that the receivers have the sum of the LOS response and multipath responses from each of the desired reflections.

3. GPU Accelerated Impulse Response

High complexity implementations of the basic IRSIM algorithm, including those with small time step, a large number of reflectors, or multiple reflections, are potentially very time consuming. Each iteration (or reflection) relies on the previously calculated matrix, therefore is dependent on the previous and should be run sequentially; however each step in the algorithm allows for parallelism and an opportunity for improvement in the overall performance.

- *Zero Bounce (LOS) Calculation*: This step can be parallelized by allowing each thread to observe a specific transmitter, receiver, or pair. Due to write dependencies, selecting a transmitter or pair per thread requires synchronization across threads; however allowing each thread to observe a receiver allows threads to run independently. The GPU function first determines the receiver associated with the thread and loops through the necessary calculations from all sources to the specific receiver.
- *Matrix Initialization*: Here we follow a similar method to the LOS calculation, now observing transmitter and reflector pairs. Again, due to write dependencies at the reflectors, we parallelize by associating threads with a reflector and having each thread loop through all transmitters. The reflectors from different planes are associated with different GPU blocks and each block is padded so that the total number of threads per block is equal to the number of reflectors in the largest plane. If the largest plane has more reflectors than the maximum threads per block, we divide each plane amongst multiple blocks. Each thread observes its location relevant to the associated plane and is considered invalid if it does not map to a reflector. If valid, the thread loops through transmitters and calls the same attenuation / delay function from the LOS calculation.

- *Received Power Update*: In this step each receiver is given a thread (as with the LOS calculation) which determines the signal from each reflector to the receiver. The parallelism here is dependent on the number of receivers rather than the complexity of the environment.
- *Matrix Update*: Here, each reflector must determine the received signal from each other reflector, leading to a complexity on the order of n^2m in the basic implementation. To parallelize this, we setup the GPU grid and blocks in the same way as the matrix initialization. Every valid thread determines its associated reflector and calculates received signal from all other reflectors – allowing the GPU implementation to potentially run with a number of execution steps on the order of nm.

4. Results and Conclusions

In order to compare the accelerated implementation with the original software, we observe performance and accuracy. Both sets of simulations are run on the same workstation – an HP Z400 with an Intel Xeon W3505 processor. The GPU used for the accelerated simulations is an Nvidia GeForce 285 with CUDA version 2.3. Table I offers an overview of the performance improvements provided by the accelerated implementation in multiple scenarios and Figures 1-4 show detailed analysis of the performance across a range of complexity metrics for the Cubicle scenario discussed in [2].

Table 1: Accelerated IRSIM speedup for various room scenarios

Scenario	Room Size	Time Resolution	Spatial Resolution	Reflections	Speedup
Small Empty Room	4m x 4m x 4m	2.5 ns	8 divisions / m	2	19.10x
Large Empty Room	8m x 8m x 8m	5 ns	10 divisions / m	4	20.73x
Cubicles / Office	6m x 6m x 3.5m	5 ns	6 divisions / m	4	12.98x

In Figures 1 - 4, we compare the execution time of the GPU accelerated implementation to the original IRSIM CPU implementation and observe the speedup. Figure 1 shows performance as spatial resolution ranges from 2 to 10 divisions / m with a time resolution of 2.5ns. Similarly, Figure 2 compares performance as time resolution ranges from 0.5ns to 5ns with a spatial resolution of 6 divisions / m. Both sets simulate 4 reflections. In general, the accelerated implementation offers an 8x to 13x performance increase for these scenarios. Higher spatial resolution scenarios offer better improvement due to the larger number of reflectors, n, which leads to a larger matrix. Similarly, larger time step scenarios have shorter impulse response arrays, m, associated with each reflector. The smaller ratio m/n leads to a higher speedup when comparing the GPU implementation to the basic CPU implementation. Speedup begins to decline at higher spatial resolution (very large n) due to growing number of invalid

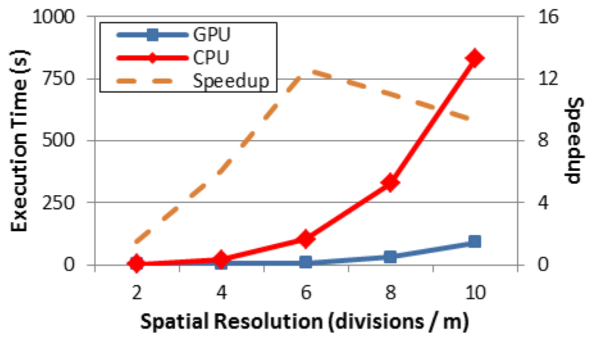

Figure 1: Performance versus Number of Reflectors

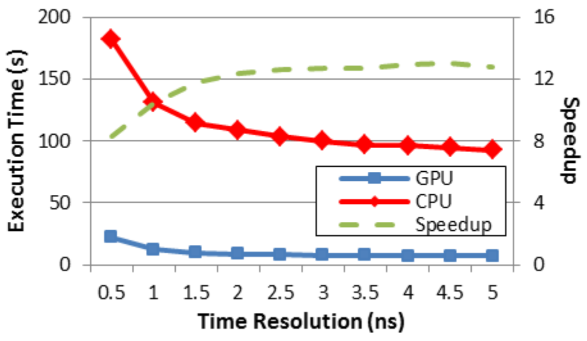

Figure 1: Performance versus Time Resolution

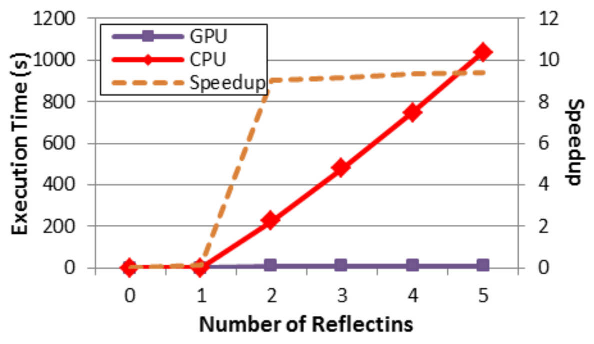

Figure 3: Performance versus Reflections
Time Resolution: 5ns

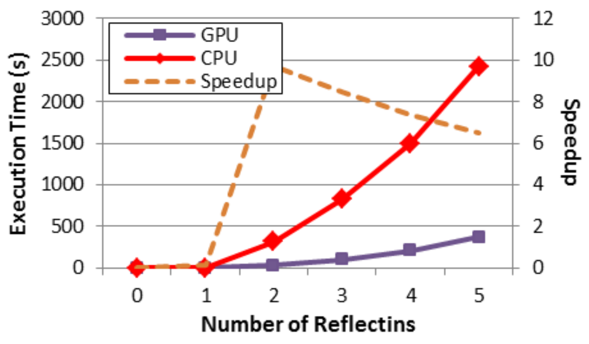

Figure 3: Performance versus Reflections
Time Resolution: 0.5ns

threads from block padding. This can be mitigated in future work by adjusting the mapping between threads and reflectors.

Figures 3 and 4 show performance as the number of reflections ranges from 0 to 5 with a spatial resolution of 10 divisions / m. Note that the CPU provides better results for LOS simulations due to the overhead of memory transfer in the accelerated implementation; however, multiple reflection scenarios perform better with the GPU implementation due to the parallelization of the matrix update step. In general, speedup remains constant as the number of reflections increases (as shown in Figure 3); however scenarios with a small time step show a drop off in speedup after the second reflection (as shown in Figure 4). Since the length of the impulse response array increases with the number of reflections, the ratio m/n also increases leading to this decrease in speedup.

In addition to performance metrics, we have also compared the output from the CPU and GPU implementations in order to validate the accuracy of the results. There was an average difference of 2nW in the calculated signal power at the receivers in the simulation, which equates to an average of 0.6% error from the basic IRSIM results. This is likely due to the precision of 64 bit floating point operations on the CPU versus 32 bit floating point operations on the GPU. Normalized impulse responses are nearly exact matches, with a 0.5% average difference.

In conclusion, we have shown excellent performance increases in the IRSIM simulation software by taking advantage of the parallelism offered by a GPU implementation. For simulations of typical resolution, we have shown speedup between 5x and 20x when compared to the basic implementation. This improvement allows simulations to complete at a much faster rate which will be beneficial for future observation and analysis of indoor optical communication systems.

5. References

[1] Cisco Visual Networking Index. "The Zettabyte Era." San Jose, CA, May 30, 2012. www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/VNI_Hyperconnectivity_WP.html

[2] M. Rahaim, T. Borogovac, and J.B. Carruthers. "CandLES – Communication and Lighting Emulation Software." *WiNTECH '10*, Chicago, IL, September 2010.

[3] J.B. Carruthers, S.M. Carroll, and P. Kannan. "Propogation modeling for indoor optical wireless communications using fast multi-receiver channel estimation." *Optoelectronics, IEEE Proceedings*, pages 473-481, 2003.