An OpenCL Implementation of Pinhole Image Reconstruction

Matthew R. Dimmock, Dmitri A. Nikulin, John E. Gillam, and Chuong V. Nguyen

***Abstract*—A C++/OpenCL software platform for emission image reconstruction of data from pinhole cameras has been developed. The software incorporates a new, accurate but computationally costly, probability distribution function for operating on list-mode data from detector stacks. The platform architecture is more general than previous works, supporting advanced models such as arbitrary probability distribution, collimation geometry and detector stack geometry. The software was implemented such that all performance-critical operations occur on OpenCL devices, generally GPUs. The performance of the software is tested on several commodity CPU and GPU devices.**

***Index Terms*—Collimator, GPU, OpenCL, pinhole.**

I. Introduction

THIS paper reports the implementation of the line of response (LoR) reconstruction aspect of a larger OpenCL image reconstruction software project, *Conan the Reconstructor*. The software has been developed to allow single-step line and cone (from Compton scattered events [1]) back projection of list-mode data in an iterative reconstruction framework for near-field small animal imaging [2].

The reconstruction platform was developed for use with the Pixilated Emission Detector for RadiOisotopes (PEDRO) [2]. PEDRO will utilize a hybrid collimation technique [3] which requires the combination of high resolution pinhole imaging data [4] with modulated Compton cone of response data [1] from larger open apertures in the collimator. Hybrid collimation aims to provide optimal detection characteristics in the energy range 30 keV $\leq E_\gamma \leq$ 511 keV. It is expected that this will result in a reduction in the sensitivity-resolution trade-off, inherent in conventional mechanically collimated configurations.

The work presented herein focuses on the implementation and performance characteristics of LoR based reconstruction appropriate for single photon emission imaging. The term LoR will refer to a conical or cylindrical volume, with a non-uniform intensity through its cross-section, representing the probability of measurement from possible emission positions.

Manuscript received May 30, 2011; revised October 05, 2011; accepted April 22, 2012. Date of publication June 14, 2012; date of current version August 14, 2012. This work was supported by the Cooperative Research Center for Biomedical Imaging Development Ltd. (CRC-BID), established and supported under the Australian Government's Cooperative Research Centers program.

M. R. Dimmock, D. A. Nikulin, and C. V. Nguyen are with the School of Physics, Monash University, Melbourne, VIC 3800, Australia (e-mail: matthew.dimmock@synchrotron.org.au; dnikulin@gmail.com; chuong.v.nguyen@gmail.com).

J. E. Gillam is with the Instituto de Fisica Corpuscular (IFIC), Universidad de Valencia–CSIC, Valencia, Spain (e-mail: john.gillam@gmail.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNS.2012.2197760

The remainder of this section will summarize the relevant aspects of medical emission imaging and existing work in GPU-accelerated image reconstruction. Section II will describe LoR reconstruction in the list-mode iterative framework. Section III will present the considerations specific to OpenCL implementation of the reconstruction. Section IV will outline low-level optimizations performed. Section V will evaluate the performance of the reconstruction on several common CPU and GPU devices. A derivation of the new probability density function (PDF) for the LoR formulation used in this framework is presented in Appendix A.

A. Single Photon Emission Imaging

Hybrid-collimation, along with techniques such as tomosynthesis [5], synthetic collimation [6] and stationary SPECT [7], exploits the use of large area detectors with a sophisticated mechanical collimator to form an image without the need for rotation. These techniques have significant benefits for geometries where rotation around the field-of-view (FoV) is limited.

As almost identical reconstruction algorithms can be utilized for tomographic and non-tomographic reconstruction, with only a simple transformation of coordinates to account for rotation, the broader term of single photon emission imaging (SPEI) can be used to encompass all of the above.

The simplest geometry for SPEI is the pinhole camera [8], where a single pixelated energy resolving semiconductor detector [9] is positioned behind a pinhole collimator. This geometry enables the acquisition of high resolution images, however the system sensitivity is limited by the collimator. Ideally, each incident photon is photoelectrically absorbed in the detector, enabling a single LoR to be formed as a back-projection from the interaction location (single pixel), through the pinhole and into the imaging volume. If the photon Compton scatters, event-ordering techniques [10] can be used to identify the primary interaction.

The reconstructed image volume is a 3D matrix of "voxels" that represent the FoV. The intensity counted in each voxel is an estimated emission total, factoring in the probability distribution of any given position having been the origin of the photon emission [11], [12].

When many LoRs are reconstructed, an estimate of the source distribution (object) is formed. To improve the system sensitivity, the number and density of pinholes can be increased. However, multiplexing can give rise to overlapping source projections that reduce the average information of each event.

In the reconstruction, forward projection (FP) of the object gives the expected counts in each measurement bin. Back-projection (BP) is then defined as $BP = FP^T$, where T refers

to the matrix transpose. In nearly all instances of image reconstruction, the projection operator is not Hermitian and therefore $BP \neq FP^{-1}$. Necessarily, the back-projection operator alone will give a poor estimate of the source distribution. In order to account for discrepancies, iterative statistical image reconstruction is widely used.

The relevant aspects of LoR formation, iterative image reconstruction and the use of GPU acceleration will now be discussed.

B. LoR Formation

The cross-section (σ) of the intensity distribution of the LoR depends mostly on the pinhole transmission and detector pixel sensitivity:

1) Pinhole Transmission: In order to analytically calculate the pinhole transmission $(T_{pin}(\theta))$, Metzler, *et al.* [13] proposed an approximation that was composed of two terms, geometric and penetrative. In their work the total transmission of the pinhole, which varies as a function of the incident angle of the photon, θ, was calculated from equations 37 and 62, where the latter was used to fit the former. This formalism gives excellent agreement with experimental data for apertures with large opening angles. However, the approximation does not include the collimator thickness, and also the magnitude of x in equation 62 is undetermined for small radii, e.g., < 0.32 mm for a Tungsten collimator and a source that emits 140 keV photons.

A more comprehensive analytic expression was later proposed in [11]. However, these formulae ((5) and (6)) do not account for the finite collimator thickness, and most importantly for the PEDRO detector, the geometry of the detector pixels.

2) Detector Pixelation: The PEDRO prototype detector is designed as a stack of five Silicon double sided strip detectors (Si-DSSDs) with a CdTe absorber detector, positioned behind an aperture array. A stack detector was utilized for two primary reasons:

- To maximize the system sensitivity, whilst minimizing the uncertainties associated with depth of interaction.
- To explore the application of PEDRO to hybrid and synthetic collimation in a static configuration.

The Si-DSSDs, manufactured by Centro Nacional de Microelectronica (CNM), each have an active volume of $32.0 \cdot 32.0 \cdot 0.8$ mm^3. The interaction positions of incident photons can be localized to the center of $0.4 \cdot 0.4 \cdot 0.8$ mm^3 voxels. The CdTe detector [14] consists of a $51.2 \cdot 51.2 \cdot 2.0$ mm^3 CdTe crystal with a pixelated anode. The interaction positions of incident photons can be localized to the center of $0.2 \cdot 0.2 \cdot 2.0$ mm^3 voxels, however it is likely that this will be up-sampled to $0.4 \cdot 0.4 \cdot 2.0$ mm^3 voxels due to charge sharing [15].

The detector pixelation can be accounted for by either back-projecting the pinhole transmission function from many locations within a single detector pixel, or convolution of the pinhole and pixel response functions followed by a single back-projection.

As the detectors are aligned in a stack geometry and are therefore different distances from the collimator, the size of the detector pixels in different layers has a variable effect on the response function at the back-projection stage. It was therefore decided that a single step LoR formation routine would be derived as it would offer a consistent result, regardless of the detector position in the stack.

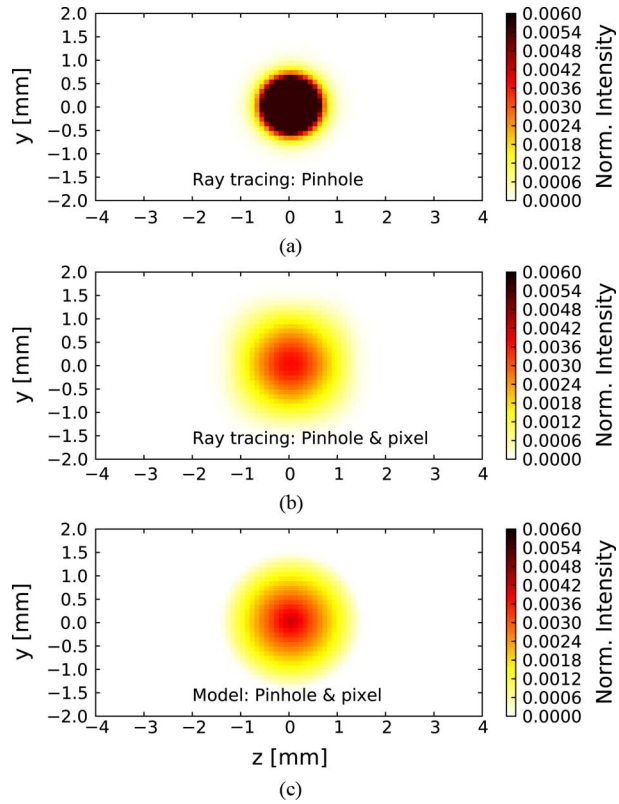

Fig. 1. A comparison of pinhole LoR cross-sections. (a) Ray tracing with a single origin at the center of the pixel (representative of the pinhole PSF). (b) The sum of ray tracing distributions from a grid of 21×21 equi-spaced origin positions within the detector pixel area. (c) An analytic model of the convolution of the pixel and pinhole cross-sections.

In order to demonstrate the effect of the detector pixelation on LoR formation, a ray tracing exercise has been performed for a representative experimental geometry. The geometry chosen is typical of that intended for operational use of PEDRO. The pinhole has inner and outer diameters of 0.3 mm and 1.55 mm, respectively. The collimator thickness is 5.0 mm, resulting in an effective diameter [11] of 0.363 mm at 140 keV. The pixel size is $0.4 \cdot 0.4$ mm^2. The distances from the source to the collimator and the collimator to the pixel were 60.0 mm and 20.0 mm, respectively. Fig. 1 shows a comparison of the cross sections of an LoR for three cases for this geometry. The cases are: ray tracing of the pinhole from a single point in the center of the pixel (see Fig. 1(a)), the same as in Fig. 1(a) plus the inclusion of the pixel area (see Fig. 1(b)), and an analytic model which includes the pinhole and pixel contributions (see Fig. 1(c)).

The model represents a convolution of the pinhole transmission and the pixel sensitivity and is derived in Appendix A. It should also be noted that the model includes an attenuation correction for the consecutive detector layers in PEDRO. The model produces a close match with the ray-traced example in Fig. 1(b), with a root mean squared deviation of 0.004.

C. Iterative Statistical Image Reconstruction

The continuous radio-tracer distribution can be discretized into a volume of J voxels $(n_x \cdot n_y \cdot n_z)$, represented as a vector

with elements $\lambda(j)$ for which $j = 1 \ldots J$. The detected emission data is typically represented as a vector of length I, where $I = M \cdot N$ is the total number of sinogram bins (possible LoRs) corresponding to the product of the M projection angles and N measurement bins.

For detector systems where discretization results in an unacceptable loss of data precision, or the histogram contains many zero value elements, or additional event attributes may improve image reconstruction, it can be beneficial to store each measurement separately in a time ordered list, referred to as list-mode data. List-mode processing allows the data to reflect the measurement geometry and so provides a highly flexible input to the image reconstruction algorithm.

In list-mode, the detected data is now a vector of length K for which $k = 1 \ldots K$ are the individual events. Now, i_k refers to the LoR along which the kth list mode event is detected. This list of measured data is related to the true image estimate through a set of linear equations. The most appropriate technique for solving for the source distribution is the form of iterative statistical reconstruction—Maximum Likelihood Expectation Maximization (ML-EM) [16], [17]. The list-mode (LM) form of the ML-EM equation is given as [18]

$$\lambda_j^{l+1} = \frac{\lambda_j^l}{\sum_{i=1}^{I} t_{i,j}} \sum_{k=1}^{K} t_{i_k,j} \frac{1}{\sum_{b=1}^{J} \cdot t_{i_k,b} \lambda_b^l} \tag{1}$$

where λ_j^l is the intensity in the jth voxel at the lth iteration, $t_{i,j}$ is the probability that an emission from voxel j is detected along LoR i (termed the system matrix), and $\sum_{i=1}^{I} t_{i,j} = s_j$ is the probability an emission from voxel j is detected by the system (termed the sensitivity matrix). The calculation of the contribution to each imaging voxel from constructing an LoR is the process of "LoR formation". The calculation $FP = \sum_{b=1}^{J} \cdot t_{i_k,b} \lambda_b^l$ is often just termed "projection". The calculation $BP = \sum_{k=1}^{K} t_{i_k,j} 1 / \sum_{b=1}^{J} t_{i_k,b} \lambda_b^l$ produces a correction matrix which updates the given image estimate. The initial image-estimate is uniform, i.e., $\lambda_j^0 = 1$.

D. OpenCL for Accelerated Emission Imaging

List-mode back-projections of distribution functions such as that detailed in Appendix A require intensive computation. Such demands have been often met in other fields with the use of commodity graphical processing units (GPUs), using standard as well as proprietary programming interfaces.

Open Compute Language (OpenCL) [19], [20], an open standard for high-performance computing, is becoming an important alternative to the proprietary Compute Unified Device Architecture (CUDA) interface. As it is supported by a wide range of computing devices from all categories, OpenCL was deemed the most appropriate acceleration platform for this project.

Previous works have experienced varying results on the relative performance of OpenCL and CUDA on the same devices, but rarely does OpenCL meet or exceed the efficiency of CUDA [21], [22]. As this has not been shown to be inherent to the specification of OpenCL itself, and rather appears to be due to the immaturity of its implementations, it is reasonable to expect that the performance difference will be reduced in the future. In return, OpenCL offers a far greater choice of implementations, including middle ware such as MOSIX Virtual OpenCL [23] and Hybrid OpenCL [24].

E. Related Work

Recently, Nguyen *et al.* [25] developed a CUDA software package for Compton cameras. As a compromise to simplify processing, at some acceptable cost to reconstruction accuracy, cones of response are modelled as sets of rays, reducing the problem to a similar conceptual complexity as this work. Although the ray-tracing method is described in their work, the details of the CUDA realization are only implied.

The work by Pratx *et al.* [26] is very different in requirements and implementation, in particular because it is aimed at positron emission tomography (PET) [27]. The software utilizes the OpenGL framebuffer object (FBO) extension to enable shaders to write directly to texture for projection.

The work presented in [26], was superseded [28] through the use of CUDA to overcome the device specific limitation of OpenGL. We consider the work of Cui *et al.* to be the most direct comparison in reconstruction workload. However, in the context of the requirements of this project, we have pursued a greater level of flexibility and generality. This has enabled the use of arbitrary experimental geometries to influence the back-projection and iterative image reconstruction processes, and for the framework as a whole to include other forms of back-projection such as cone of response (CoR).

II. LM-ML-EM Implementation for LoRs

A. LoR Sub-Volume Reconstruction

With line back-projection through pinholes, there is a simple ray-tracing calculation to determine the set of voxels that receive non-zero intensity along the path of the center-line. This is achieved with the assumption that the conical LoRs are cylindrical, with the cross section of the cylinder being slightly larger than that of the biggest cone. The pre-calculation is performed once for the entire event vector and retained for all event reconstructions. The collection of sub-slices, hence sub-volume as a whole, denoted as J_{sub}, along a pinhole LoR is shown in Fig. 2.

B. Generalized LM-ML-EM Algorithm

For a practical SPEI device, (1) must be implemented for an arbitrary number of pinholes. This implementation is summarized in Algorithm 1. In its most general form, where no attempt is made to determine the pinhole through which an incident photon may have originated, the LoR back-projection must be performed through all apertures for each event (lines 5–8 in Algorithm 1). In practice, an extra analytic step is applied to geometrically determine the most likely pinhole(s).

For the PEDRO implementation of Algorithm 1, the sensitivity was calculated both analytically and through modelling with the Monte-Carlo radiation transport code GEANT4 [29]. Analytic calculation of the sensitivity was evaluated to have a mean uncertainty of within 5% of that modelled by stepping a point source through the FoV. Having validated the method, it was deemed desirable to analytically compute the sensitivity matrix through the back-projection of all combinations of de-

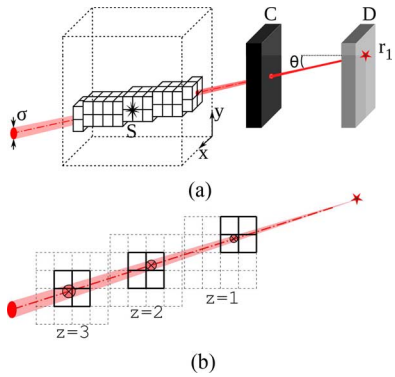

Fig. 2. (a) A photon emitted from a point source (S) passes through a pinhole in the collimator (C) and interacts in the detector (D) at location r_1 . An LoR is subsequently back-projected from r_1 , through the pinhole and into the imaging volume. (b) Fixed-size sub-slice selection, also shown in (a), for a given LoR. Each shaded circle represents the non-zero slice of the LoR at each depth plane. Each square (slice) represents the sub-slice that will be allocated to a single OpenCL work group.

Algorithm 1 LM-ML-EM algorithm

Set sensitivity matrix s_j.
Set initial image estimate λ_j^1 to 1.
for iteration $l = 1$ to L **do**
  for event $k = 1$ to K **do**
    Form merged LoRs:
    for pinhole $p = 1$ to P **do**
      $t_{i_k,a} = t_{i_k,a} + \mathrm{LoR}(i_k, a, p)$
      where $a = 1...J_{sub}$
    end for
    Forward-project: $FP_k^l = \sum_{b=1}^{J} t_{i_k,b} \cdot \lambda_b^l$
    Back-project: $BP_j^l = BP_j^l + \frac{t_{i_k,j}}{FP_k^l}$
  end for
  Update estimate: $\lambda_j^{l+1} = \frac{\lambda_j^l}{s_j} BP_j^l$
end for
where $j = 1...n_x \cdot n_y \cdot n_z$

tector pixel-pinhole combinations and store the probabilities in memory. This removes the need for the sensitivity matrix to be remodelled if the geometry, e.g., detector spacing, is modified.

III. OpenCL LM-ML-EM Implementation

The simplest interface to the platform's functionality is a command line application, intended to be used as a back-end by other software such as graphical user interfaces. Two modes of iterative reconstruction have been implemented, LM-ML-EM and OPL-EM [30].

The software has several distinct stages. Before the iterative computation stage, data files are loaded that describe the list-mode event data, and the physical objects implementing the detector geometry in the data acquisition. After the iterative computation, simple filters such as Gaussian blur may be applied and finally the reconstructed volume is recorded to an output stream or an image file.

A. World and Data Model

The image reconstruction requires knowledge of the world geometry e.g., pinhole and detectors locations. To achieve this, a *world model* is made available via C-style structures that OpenCL can access. The use of pointers is restricted in OpenCL, as the memory models of devices may not be known or portable, and one of the consequences is that pointers cannot be passed as input data to kernels [20].

As any realistic world geometry is unlikely to exceed a well-bound level of complexity, it is simple to represent the entire geometry as a single hierarchical C structure. Despite generally leaving unused gaps in the overall structure, this has several key strengths.

The set of headers describing the C structure can be included from C, C++ and OpenCL source, and be interpreted identically because of the deliberate commonalities between the standards, requiring some discipline in the use of `cl_` prefixed types in the headers and appropriate host C/C++ code.

Although the experimental geometry has a reasonably limited complexity, there may be millions of list-mode events fed to devices, and so they must be delivered to devices in buffered chunks. Each buffer contains a contiguous array of event structures, portioned to at most a fixed number.

The intention is to keep both buffers—the world data and the event data—all within a device's very small `constant` memory pool. On many available GPU devices, constant memory is as small as 16384 bytes, though modern devices extend this to as much as 65536 bytes. Unlike `global` and `local` memory, `constant` memory is aggressively cached and extremely efficient to access, even in random access patterns. So while `constant` memory is desirable for as much of the input memory as possible, it must be carefully rationed.

It is important to note that because the data types are lists of structured records of mixed sizes and types, we have chosen to use `constant` memory instead of image objects. Also, as every thread processes a voxel, and is required to access *every* list-mode event record plus the world geometry, `constant` memory provides even for highly irregular data structures and access patterns. It is difficult to achieve the same memory access efficiency with image objects alone, though an even more efficient design would combine both types of memory to draw upon their strengths as applicable.

This is one of the key differences between histogram and list-mode reconstruction on GPU devices. Histogram reconstruction requires dense input and output data, with computations of a low complexity and high mathematical density. List-mode reconstructions involve irregular data access and computational patterns, and are far more difficult to optimize.

B. Kernel Selection—Runtime-Memory Trade-off

Fig. 3 shows a data flow that could be implemented for Algorithm 1, where all sub-slice information is stored for both the forward and back projection Kernels. The details of this flow can be tuned in favour of runtime or memory, but this is not a simple switch. A trade-off investigated in this work was that the FP can be updated from the merged LoRs, whereas the BP requires them to be kept separate. Therefore, the sub-slice origins

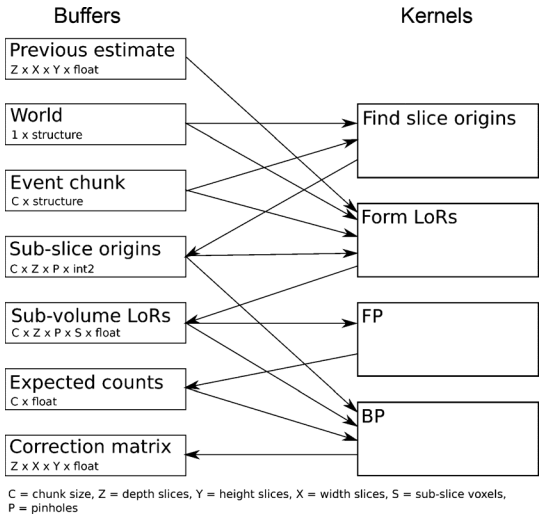

Fig. 3. Example kernel data flow for a single EM iteration, if origins and merged LoRs are stored.

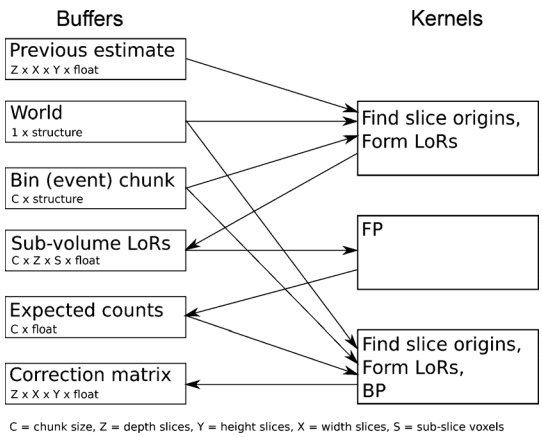

Fig. 4. Example kernel data flow for a single EM iteration, if origins and merged LoRs are recalculated as shown in Algorithm 2.

and the P dimensionality in the sub-volume LoRs must be either stored in the buffer or recalculated at the BP stage so that the sub-volume slices can be realigned to their correct position in the global memory.

As the sub-volume buffer is the largest in use, and affects the maximum event chunk size, this is a primary target for memory optimization. The tests performed with these two complementary decisions (store or recalculate origins, store or recalculate merged LoRs), showed that the code complexity increased considerably without a significant change in performance. The simplest mode was selected, where origins and LoRs are recalculated (see Fig. 4). The resulting three OpenCL kernels are specified in Algorithm 2.

C. Reconstructing in Sub-Slices of Fixed Size

Optimization of Algorithm 2 for highly parallel devices such as GPUs has produced a very simple and generally effective specialization of this algorithm.

The software allocates one OpenCL "work group" (conceptually similar to a CUDA "block", and essentially isolated from other groups) to a depth slice of the volume. The OpenCL platform is responsible for scheduling the work groups, and the threads within them, to execute on the available physical threads. In general, each compute unit of the GPU will receive one work group at a time, so it is important that the work group be sized to use as many of the compute unit's number of threads as possible.

Algorithm 2 Optimized kernels for LoR formation with sub-volume reconstruction

1: Kernel I : $t_{i_k,a} = \sum_{p=1}^{P} \mathrm{LoR}(i_k, a, p) \cdot \lambda_a^l$
2: Kernel II : $FP_k^l = \sum_{b=1}^{J} t_{i_k,b}$
3: Kernel III: $BP_j^l = BP_j^l + \frac{\sum_{p=1}^{P} \mathrm{LoR}_{(i_k,j,p)}}{FP_i^l}$
4: where $a = 1...J_{sub}$
5: where $j = 1...n_x \cdot n_y \cdot n_z$

The number of threads in the work group is the square of the maximum width[1] of *any* known line in voxels (predictable from the camera geometry), so that each thread processes exactly one voxel.[2] For each work group, the centre of the line in the slice is calculated using linear interpolation, tracing from the interaction in the detector stack, through the pinhole and into the imaging volume at the given depth. This determines the actual voxel index, within the 2D slice, that each thread will reconstruct. This grouped calculation is repeated once per depth slice, per pinhole, per event.

Note also that CPU devices do benefit from branched short cuts in the computation, whenever it is clear that the rest of the per-voxel computation cannot result in positive intensity contribution. This has the effect of relatively penalizing GPU devices for their constraints with regard to branch divergence, and partly accounts for the performance seen in Section V-C.

D. Transient and Final Data

For the initial back-projection, the voxels are written to a single volume in OpenCL global memory. Each work group operates on a slice, separate from any other work group, and every thread operates on a voxel within its group's sub-slice, separate from any other voxel.

For iterative reconstruction (EM), the contributions must be reduced to a per-event total, representing the forward projection to a single detector interaction. To that end, the per-event-per-sub-slice-voxel total is recorded in a separate global memory buffer. A forward projection kernel collects these subtotals using a parallel reduction (see Section III-E).

The same sub-volume can be used for the forward projection and to update the image in Algorithm 2. The merged LoR volume is still a dense matrix in implementation, but represents the non-zero sub-volume of the full back-projection volume. All three kernels are reduced in runtime and memory cost using this sub-volume approach, with significant enhancement over a total volume search ("brute force") approach (see Section V-C).

[1]As sliced by any imaging volume plane.

[2]This is also a limitation in cases with a very wide line of response, however rare in practice. This case would need to be handled by a generalization to more than one voxel per thread.

E. Estimated Forward Projection

The forward projection kernel collects per-event-per-sub-slice-voxel totals into only per-event totals. This is the *only* kernel that is specialized between CPU and GPU, because CPUs have far fewer threads and can achieve full thread saturation with one thread per event. Even the most modest GPUs have more threads than events in our event chunks. The GPU kernel performs a parallel reduction in local memory, a very common idiom for fast data reductions. In our kernel, a single work group handles the per-voxel totals for a single event.

Each thread in the work group collects per-voxel totals in a coalesced memory access pattern, then stores its own total in local memory, exactly one 32-bit floating point number per thread. A parallel reduction kernel reduces the per-thread totals to a single total, which is then stored in global memory. At every iteration of the reduction, the number of threads is halved, until one thread completes the final 32-bit floating point value. Therefore, the total runtime is logarithmic with respect to the number of parallel threads in the work group, *independent* of the number of events and voxels originally reconstructed (as they were already accumulated by threads in the previous kernel).

For a single event, on CPU and GPU alike, there is only one global memory write (the final total) and only one global memory read per voxel (and only those voxels included in sub-slices, see Section III-D). Local memory on the GPU is sized only as per the number of threads, and read/written only logarithmically with respect to the number of threads. These qualities make the forward projection kernel very economical overall.

IV. Low-Level Optimizations

A. Runtime Specialization

As OpenCL source code is issued to the platform at runtime, the source code can be arbitrarily generated or processed before being compiled, and this may be performed per device if desired.

The application mastering the OpenCL devices has user preferences, and may partially predict the data set. Instead of passing user preferences in a global structure to static code along with the world model memory (see Section III-A), many preferences and runtime decisions can be passed to the C preprocessor that runs as part of OpenCL compilation.

In *Conan the Reconstructor*, several user preferences result in source code specialization, such as the values of constants defining the imaging volume, the choice of whether to perform EM iterations, and whether to use a CPU-optimized kernel for sum reduction (see Section III-E).

B. Vector Types

OpenCL natively supports fixed-size numerical vector types, complete with operators and functions defined over appropriate sizes and types of vectors. These vector types support concise and efficient arithmetic expressions, and are used extensively within the OpenCL portion of the software.

V. Performance Evaluation

A. Reconstruction Data

The experimental hardware for the PEDRO is still under construction. Therefore, the detector geometry and interaction list were simulated with GEANT4. The source was a planar Derenzo [27] resolution phantom (see Fig. 5(a) positioned 60.0 mm from the collimator along the z-axis. The three-pinhole test collimator and detector configuration are described in detail in [3]. The position and energy resolution parameters for the detectors were included in the model [3].

An example reconstruction for the PEDRO is presented in Fig. 5. The image estimate λ_j^2 after the first iteration $(l = 1)$, which can be considered as the initial back-projection of the detected list-mode data at the slice of the imaging volume $z = 60.0$ mm, is shown in Fig. 5(b). The same slice at $z = 60.0$ mm after 10 iterations $(l = 10)$ of the LM-ML-EM algorithm is shown in Fig. 5(c). Fig. 5(d) shows the root mean squared difference (RMSD) as a function of iteration number, calculated as the pixel-by-pixel difference[3] between λ_j^{l+1} and the distribution presented in Fig. 5(a). The estimate is shown to converge towards the known phantom even within 10 iterations.

Although this work is primarily concerned with the OpenCL implementation of the algorithm, two key points regarding image quality observed at the 10th iteration (see Fig. 5(c)) will briefly be discussed. Firstly, blurring from the multiplexed data is still observable outside the object boundary. The magnitude of this blurring diminishes with increasing iteration. Secondly, there are ring artefacts around the larger, 1.5 mm radius, features. This is attributed to a small overestimation of the point-spread function of the system (see Fig. 2 in [31]). This is demonstrated in Fig. 6, which shows 1D slices through the distributions shown in Fig. 1.

The remainder of this section will discuss the performance of the OpenCL implementation of the algorithm.

B. Devices Evaluated

Six devices were chosen, two per vendor out of three major vendors. The OpenCL devices tested are aliased to single letters. Table I lists relevant details for each device.

The Intel Core i7 930 is the baseline (alias A), as it is a CPU with a very high performance-to-cost ratio, and is common in desktop PCs in 2011. For professional workstations, the Intel Xeon E5520 (alias B) is taken as a common choice for a dual-socket configuration. Though architecturally similar, the E5520 has a lower clock speed (2.26 GHz) than the i7 930 (2.80 GHz).

An AMD Mobility Radeon HD 5650 (alias C) was chosen to represent a very low-end portable GPU. An AMD Radeon HD 5850 (alias D) represents a low-end desktop GPU.

An NVIDIA GeForce GTX 460 (alias E) represents a low-end GPU from the NVIDIA side, though more expensive and powerful than the AMD Radeon HD 5850. The NVIDIA Tesla C2050 (alias F) is the only device that would not be considered "consumer-grade" and is chosen to represent a common choice for high-performance workstations and clusters.

[3]This is not a comparison against another reconstruction, it is a comparison against the theoretically perfect reconstruction that no statistical system is able to achieve.

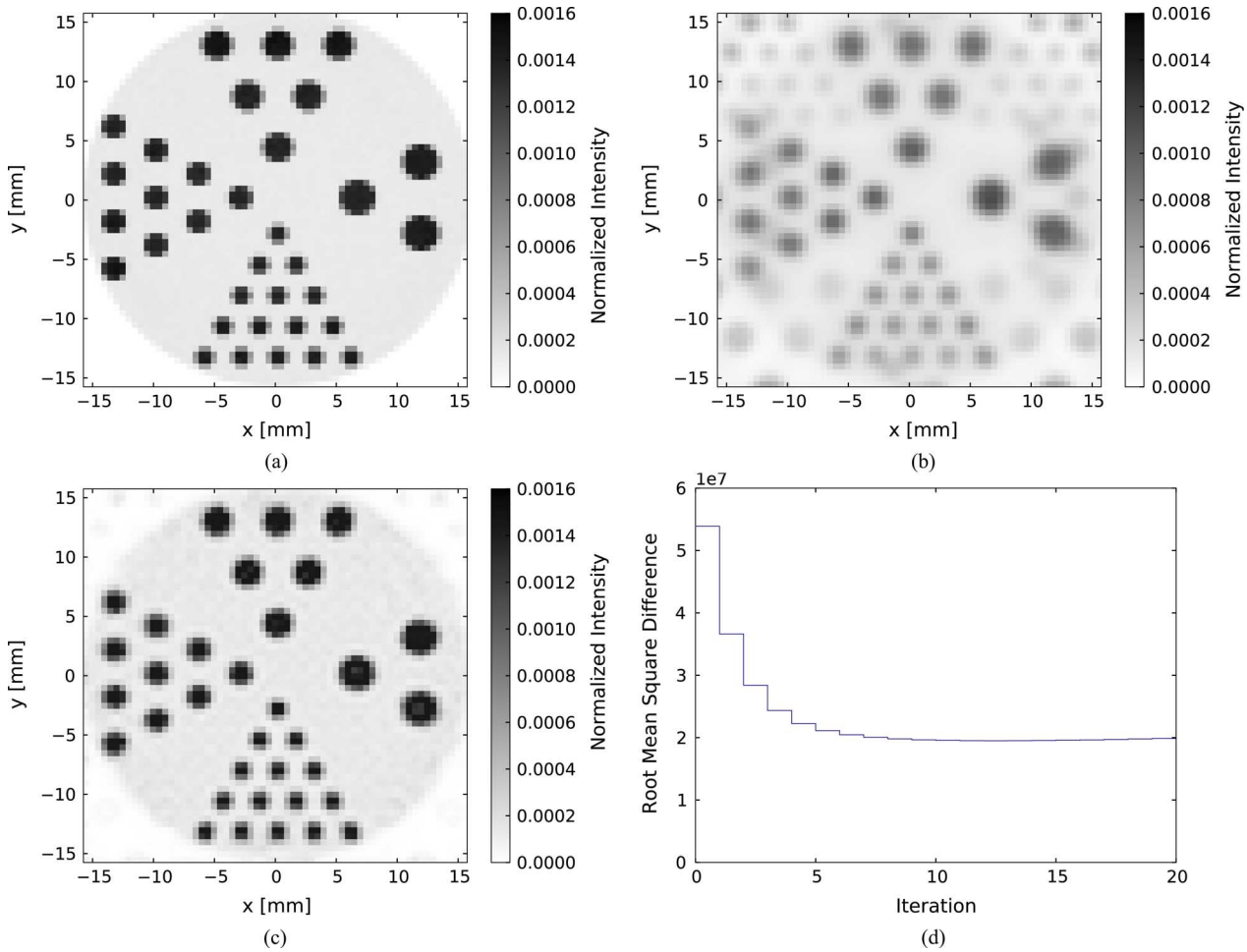

Fig. 5. (a) A histogram of the number of emitted photons as a function of position from the planar non-attenuating Derenzo phantom ($z = 60.0$ mm). (b) Image estimate after the first iteration ($l = 1$), λ_j^2 at imaging volume slice $z = 60.0$ mm—also considered as the initial back-projection of the detected list-mode data. (c) Image estimate after the 10th iteration, λ_j^{11}. (d) The root mean squared difference (RMSD), calculated as the pixel-by-pixel difference between λ_j and the original known distribution presented in (a), as a function of iteration.

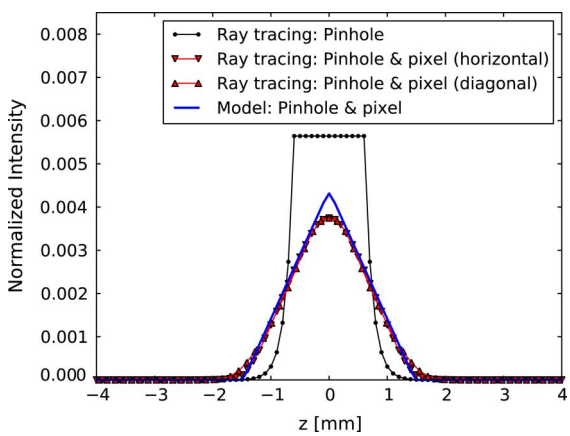

Fig. 6. 1D slices through the distributions shown in Fig. 1. From the cross sections presented, the small overestimate at the center of the distribution for the model can be observed.

Ubuntu Linux 10.10 64-bit is used for all experiments. The AMD APP SDK version 2.4, 64-bit, is used as the OpenCL platform for CPUs and AMD GPUs. NVIDIA CUDA 3.2.1 is used as the OpenCL platform for NVIDIA GPUs. Because extremely little C++ code is timed in these measurements, the C++ compiler options have been observed to not affect the device numbers.

TABLE I
OpenCL Devices Tested

	Vendor	Device	Type
A	Intel	Core i7 930	CPU
B	Intel	Xeon E5520 (x2)	CPU
C	AMD	Mobility Radeon HD 5650	GPU
D	AMD	Radeon HD 5850	GPU
E	NVIDIA	GeForce GTX 460	GPU
F	NVIDIA	Tesla C2050	GPU

Every device is left at its factory settings for clock speed. Intel CPU microcode is the latest available for both CPUs. GPU drivers and OpenCL implementations are the latest available as of April 2011, upgrading over those supplied with Ubuntu Linux 10.10. Hyperthreading, SpeedStep and TurboBoost are disabled on the CPUs, as they all add considerable noise to performance measurements.

TABLE II
RAW THROUGHPUT (vpe/μs)

	A	B	C	D	E	F
	930	E5520	5650	5850	460	C2050
Sub-slice	7191	10770	18030	77499	84471	139788
Full-slice	213	390	624	2925	3003	4524

C. Reconstruction Throughput

For all benchmarks, the reconstructed volume is a solid cube of 128^3 (corresponding to 32.0 mm^3), although the Derenzo phantom is planar. It was estimated that the width of any line on any slice is at most 10 voxels. Therefore, the number of LoRs performed per pinhole per event is at most 128×10^2, reducing the total number by a factor of 163 compared to the full ("brute force") 128^3 possible. The chain of LoR formation, forward-projection and back-projection were performed for each device.

The event vector was chunked into groups of 256 per OpenCL kernel invocation. Recall that the computation is performed in parallel over voxels, serially over events and so the small chunk size does not effect the parallelism of the computation.

Throughput is presented in *voxel-pinhole-events per microsecond*, shortened to vpe/μs. This represents the effective number of back-projection intensity calculations completed per microsecond. This metric was designed to normalize against the volume size, the number of pinholes, and the event vector size, characterizing how well a *device* performs in a specific combination of features with limited influence from the data dimensionality.

Table II contains the raw total throughput. To gauge the effectiveness of the "fast" sub-slice computation technique, the "slow" "brute force" alternative is also presented.

From the data, it can be seen that the actual runtime saving is not as high as the factor of 163. This is attributed to the pre-calculation having a non-zero cost and a degree of under-utilization of the available threads on the GPUs, from this generalized algorithm. As the purpose of this work is to establish a reasonable implementation of a new software design with a new probability distribution function, we defer any high-end device optimization to further works.

It can also be seen that, at the purely algorithmic level, no operation scales worse than linearly with respect to the adopted metric. In practice, resource limits such as latency and memory architecture may cause nonlinear effects. For this reason these benchmarks are conducted with a reasonable fixed volume size and over large vectors of simulated list-mode event data. All outputs were numerically verified against each other, as well as visually with a simple volume viewer.

The more powerful devices require a large volume size to utilize fully. Even though they have a higher maximum throughput, for a small volume they may be no faster than a less powerful device.

Table III shows the ratios of device reconstruction throughput as compared to the baseline CPU device. The GPU devices clearly dominate CPU devices in performance, as intended and expected.

D. Runtime and Event Rate in Practice

A specific test setting has been selected to estimate the *rate* of events that can be reconstructed into volumes of different sizes.

TABLE III
RELATIVE THROUGHPUT (DIMENSIONLESS)

A	B	C	D	E	F
930	E5520	5650	5850	460	C2050
1.00	1.46	2.45	10.47	13.11	21.65

TABLE IV
THROUGHPUT FOR DIFFERENT IMAGE SIZES ON A TESLA C2050. RECONSTRUCTING 100,000 EVENTS THROUGH THREE PINHOLES.
s = SECONDS FOR A SINGLE ITERATION.
e/s= EVENTS PER SECOND IN THE ITERATION.

Unit	16^3	32^3	64^3	128^3	256^3
s	1.209	1.55	1.886	5.144	26.544
e/s	82713	64516	53022	19440	3767

The test device is the same NVIDIA Tesla C2050, alias F in the benchmarks above. Exactly 100,000 events are reconstructed in each test, with an event chunk size of 256, suggesting that the OpenCL loop is repeated 390 times with a full chunk and once with a partial chunk. The measurement is intended to be holistic, so the software is not deliberately optimized for this scenario. All of the supporting code, such as for operating multiple devices in parallel, contributes to the overhead. As stated in Section V-A , the data set is simulated from a geometry with three pinholes, requiring three sets of volume back-projections per event.

Table IV presents the total runtime, including memory and computational overhead but *not* file IO overhead (see the computational complexity summary in Section II-B for the justification). There is a nonlinearity at the low end of the image sizes, where not only is the set-up overhead of the computation a large factor in the total runtime, but the device itself is being only minimally utilized. At the high end, overall throughput is promising. These figures are most informative when taken in the context of an expected experimental setting where the geometry is known and the range of event throughputs can be estimated, informing reasonable image volume sizes that would support reconstruction, perhaps even real-time in some cases.

We test with cubic volumes as is conventional in image reconstruction benchmarking. However, the software does not require a cubic volume, as would be the case for many real imaging scenarios.

Where the data rate is too high for a system, the image volume can be rebalanced down (to any product of powers of two, not just cubes), and where the data rate is low enough, the image volume can be rebalanced up to make better use of the data available. Devices have memory limits that prevent an especially large volume from being completed, but the total data vector size is effectively boundless, and affects the total runtime by a linear factor.

VI. CONCLUSION

A new software platform has been developed that incorporates a novel analytic LoR distribution function for image reconstruction in SPEI devices. The techniques used are implemented in highly portable open standard programming languages and APIs. The platform enables arbitrary probability distributions and experimental SPEI geometries to be deployed.

The OpenCL implementation of the LM-ML-EM algorithm with the new analytic LoR distribution function was tested—with throughput assessed in vpe/μs. The tests show modest gains in computational efficiency, with the Tesla C2050 recording an improvement of a factor of more than 21, as compared to a benchmark device.

In assessing the performance of the C2050 in executing the kernels as a function of reconstruction volume size, the time taken to process 100,000 events was measured. A nonlinearity at the low end of the image sizes was observed. For small volumes, the set-up overhead of the computation becomes a large factor in the total runtime and the device itself is only minimally utilized.

Although the work presented in this paper is conceptually similar to that in Cui *et al.*, the complexity of the distribution function and the generality of the code for accommodating various imaging geometries means that our improvement of 21 times is significantly lower than that of the 188 times reported in [28].

Appendix
Analytic Model

In forming an LoR, the back-projection angle θ is determined by the photon trajectory relative to the center-line of the (cylindrically symmetric) pinhole. The intensity in an image voxel assigned from the formation of an LoR can be described as the convolution,

$$I_{vox}(\mathbf{r} \mid \theta) = \int\int\int \left(T_{pin}(\theta) \otimes S_{det}(\theta)\right) dV'_{vox} \tag{2}$$

where $\mathbf{r}$ is the location of the voxel center, T_{pin} is the pinhole transmission function, S_{det} is the detector sensitivity and the integral term accounts for the intersection of the LoR with the volume of the image voxel V_{vox}.

A. Pinhole Transmission

Fig. 7 shows schematically how the 2D convolution of the pinhole and pixel shapes yields a trapezoid, where d_{pix} is the diameter of the pixel and d_{eff} is the effective diameter of the pinhole ((10) in [13]). In the extension to 3D, the pixel is considered a disc of diameter $d_{pix} = width$ (e.g., 0.4 mm for the Si-DSSDs), to enable the convolution. Therefore, the trapezoid becomes a Frustrum with inner and outer radii R_{in} and R_{out}, given by

$$R_{in} = \frac{|D_{eff} - D_{pix}|}{2} \tag{3}$$

and

$$R_{out} = \frac{D_{eff} + D_{pix}}{2} \tag{4}$$

where

$$D_{eff} = d_{eff}\frac{l_1 + l_2}{l_2} \tag{5}$$

and

$$D_{pix} = d_{pix}\frac{l_1}{l_2}. \tag{6}$$

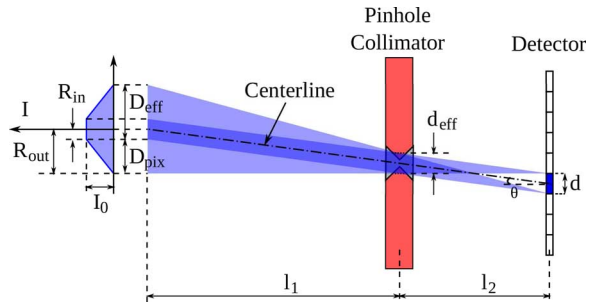

Fig. 7. Schematic representation of the 2D trapezoidal pinhole transmission function. The shape is determined by d_{eff}, d_{pix}, l_1 and l_2.

The intensity distribution of the LoR varies with distance from the pinhole (l_1) such that its integral over the cross-section is unchanged, i.e., 1. The integral over the cross-section of the LoR is equivalent to the volume of the Frustrum with height I_0 that varies with radial distance R as

$$I(R) = \begin{cases} I_0 & \text{if } |R| \le R_{in} \\ 0 & \text{if } |R| \ge R_{out} \\ I_0\left(1 - \frac{R - R_{in}}{R_{out} - R_{in}}\right) & \text{if } R_{in} < |R| < R_{out}. \end{cases} \tag{7}$$

The angular dependence of the pinhole transmission modifies this distribution by a factor

$$T_{pin}(\theta) = T_{hor}(\theta) \cdot T_{ver}(\theta) \cdot T_{pin-det}(\theta) \tag{8}$$

where T_{hor} and T_{ver} are the transmission components due to the changes in effective horizontal and vertical diameters of the pinhole cross-section, and $T_{pin-det}$ is the angular dependence of the pixel cross-section due to the distance between the pinhole and the pixel. Furthermore, the magnitude of the angle θ can fall into one of two ranges $0 < \theta \le \alpha/2$ and $\theta > \alpha/2$, which must also be accounted for.

1) Vertical Transmission for $0 < \theta \le \alpha/2$: T_{ver}, in the range $0 < \theta \le \alpha/2$, depends on both the mechanical and effective diameters (d_{eff}) of the pinhole. In this discussion, the angle α is defined by the inner mechanical pinhole diameters (see Fig. 8(a)). The maximum intersection length of a photon that traverses the Tungsten collimator is determined by d_{eff} (or r_{eff}). The term r_{eff} can be calculated as a function of θ using the assumption that the effective intersection length, δ_{int}, is constant with respect to θ. When $\theta = 0$, the edge of the ray lies along AB, $r_{eff} = d_{eff}/2 = OI \equiv r_{eff,0}$ and hence,

$$\delta_{int} = AB = \frac{2(r_{eff,0} - r_{in})}{\tan(\frac{\alpha}{2})}. \tag{9}$$

If δ_{int} is assumed constant through the range $0 < \theta \le \alpha/2$, then $\delta_{int} = AB = CD$ and the distance from O to CD gives the new effective radius at the angle θ i.e., $r_{eff} = OJ$. From this simple geometric representation, five equations can be assembled. The first two describe the intersection locations of the line CD with the *r-axis* as

$$r_C = r_{in} - x_C \tan\left(\frac{\alpha}{2}\right) \tag{10}$$

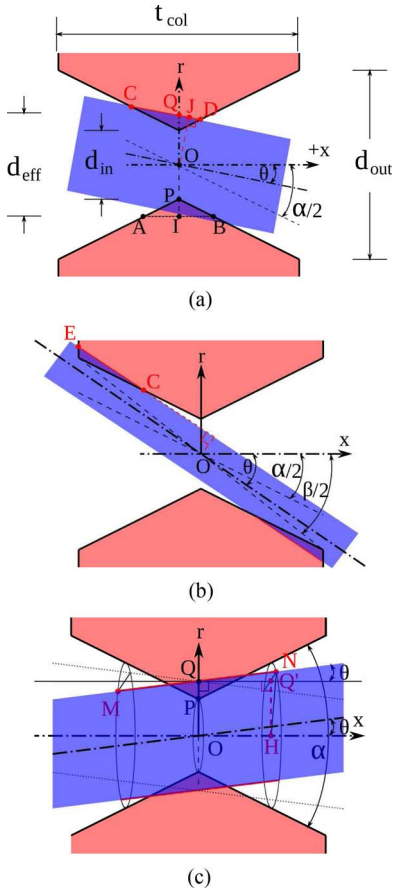

Fig. 8. (a) The variation of effective radius OJ (where O is the origin at the centre of the symmetric pinhole) with respect to the beam angle θ. (b) The beam passing through pinhole is partially blocked by the out edges when $\alpha/2 < \theta < \beta/2$. The beam is fully blocked when $\theta \geq \beta/2$. (c) Effective radius for the horizontal component.

and

$$r_D = r_{in} + x_D \tan\left(\frac{\alpha}{2}\right) \tag{11}$$

where the difference in sign for (10) and (11) is the result of choosing the origins of the x and *r-axes* at the centre of the pinhole. The gradient of the line CD also yields the following relations:

$$x_D - x_C = \delta_{int} \cos\theta \tag{12}$$

and

$$r_D - r_C = -\delta_{int} \sin\theta. \tag{13}$$

The last of the five equations gives the tangent of α as

$$\tan\left(\frac{\alpha}{2}\right) = \frac{2(r_{out} - r_{in})}{t_{col}} \tag{14}$$

where t_{col} is the thickness of the collimator. Subtracting (10) from (11) and substituting the result into (13) yields

$$x_D = -\frac{\delta_{int} \sin\theta}{\tan\left(\frac{\alpha}{2}\right)} - x_C. \tag{15}$$

Rearranging (12) for x_D and then equating this to (15) then gives

$$x_C = -\frac{\delta_{int}}{2}\left(\cos\theta + \frac{\sin\theta}{\tan(\frac{\alpha}{2})}\right). \tag{16}$$

The vertical component of the effective radius $r_{eff,ver}$ for a the plane at angle θ is the distance OJ. This can be calculated from x_C and x_D using the standard equation for the perpendicular distance between a point and line:

$$r_{eff,ver} = \frac{|\,(x_D - x_C)r_C - x_C(r_D - r_C)\,|}{\sqrt{(x_D - x_C)^2 + (r_D - r_C)^2}}. \tag{17}$$

By definition,

$$\sqrt{(x_D - x_C)^2 + (r_D - r_C)^2} = \delta_{int}. \tag{18}$$

Therefore, substitution of (11), (12) and (18) into (17) yields

$$r_{eff,ver} = \quad |\, r_C \cos\theta + x_C \sin\theta \,|\,. \tag{19}$$

Normalization of (19) gives the vertical contribution to the transmission:

$$T_{ver}(\theta) = \frac{|\, r_C \cos\theta + x_C \sin\theta \,|}{r_{eff,0}} \quad \text{if } \theta < \frac{\alpha}{2}. \tag{20}$$

2) Vertical Transmission for $\theta > \alpha/2$: For photon trajectories with angle of incidence $\theta > \alpha/2$, the attenuation due to the intersection of the photon path with the outer extents of the pinhole (see Fig. 8 b) is described by a different set of equations. This intersection, represented by the line $\delta_{int} = EC$ gives the set of equations

$$r_C = r_{in} - x_C \tan(\frac{\alpha}{2}), \tag{21}$$

$$x_E = -\frac{t_{col}}{2}, \tag{22}$$

$$x_C - x_E = \delta_{int} \cos\theta, \tag{23}$$

$$r_C - r_E = -\delta_{int} \sin\theta. \tag{24}$$

Through steps analogous to those used to obtain (19), it can be shown that

$$r_{eff,ver} = \quad |\, r_E \cos\theta + x_E \sin\theta \,|\,. \tag{25}$$

Normalization yields

$$T_{ver}(\theta) = \frac{|\, r_E \cos\theta + x_E \sin\theta \,|}{r_{eff,0}} \text{ if } \frac{\alpha}{2} \leq \theta < \frac{\beta}{2}, \tag{26}$$

$$T_{ver}(\theta) = 0 \text{ if } \theta \geq \frac{\beta}{2}, \tag{27}$$

where β is another pinhole opening angle defined as $\beta/2 = \arctan(r_{out}/t_{col}/2)$.

B. Horizontal Transmission

In deriving the contribution to the pinhole transmission in the vertical direction, the pinhole was modelled as a 2D object. This oversimplification can be addressed by accounting for the contribution in the horizontal direction (see Fig. 8(c). The horizontal effective radius $r_{eff,hor}$ for a photon of vertical

angle of incidence θ can be calculated with the same assumption that the intersection depth $\delta_{int} = MN$ is constant. Here, $r_{eff,hor} = OQ$ and the coordinates of N are

$$x_N = \frac{\delta_{int}}{2}\cos\theta \tag{28}$$

and

$$r_N = r_{in} + x_N \tan\left(\frac{\alpha}{2}\right). \tag{29}$$

By definition $OQ = Q'H = \sqrt{(NH)^2 - (Q'N)^2}$, giving

$$r_{eff,hor} = \sqrt{r_N^2 - x_N^2 \tan^2\theta}. \tag{30}$$

Normalization yields

$$T_{hor}(\theta) = \frac{r_{eff,hor}}{r_{eff,0}} \text{ if } \theta < \frac{\beta}{2}, \tag{31}$$

$$T_{hor}(\theta) = 0 \text{ if } \theta \geq \frac{\beta}{2}. \tag{32}$$

1) Detector Pixel Solid Angle: The number of photons incident on the area of a pixel $T_{pin-det}$ decreases with distance squared as

$$T_{pin-det}(\theta) = \frac{d_{eff}^2 \cos^2\theta}{l_2^2}. \tag{33}$$

C. Detector Sensitivity

The effective thickness, and hence the sensitivity of the detector ($S_{det}(\theta)$) is a function of the incident angle of the gamma-ray. $S_{det}(\theta)$ is defined as

$$S_{det}(\theta) = 1 - \exp\left(-\mu_{det}\frac{t_{det}}{\cos\theta}\right) \tag{34}$$

where μ_{det} and t_{det} are the linear attenuation coefficient and the thickness of the detector material.

For cases where the detector has multiple layers (a stack geometry), the sensitivity ($S_i(\theta)$) of detector i, which accounts for the attenuation of detectors from 1 to $i-1$, can be expressed as

$$S_i(\theta) = \exp\left(-\sum_{j=1}^{i-1}\mu_j\frac{t_j}{\cos\theta}\right)\left[1 - \exp\left(-\mu_i\frac{t_i}{\cos\theta}\right)\right]. \tag{35}$$

D. Calculation of the LoR-voxel Intersections

The intensity assigned to each voxel in the imaging volume is the intersection of the LoR (i.e., the Frustrum that represents the convolution in (2)) with the voxel. Ideally this should be calculated from a 3D integration. To simplify this numerical procedure into an analytic function, the calculation is reduced to 1D integration over the intensity of the Frustrum:

$$\int\int\int I(\mathbf{r}')dV'_{vox} \approx 2r_{vox}\int_{r-r_{vox}}^{r+r_{vox}} I(r')dr' \tag{36}$$

where r_{vox} is half the voxel width and the 1D radial integration is the summation of the trapezoidal areas between $r - r_{vox}$ and $r + r_{vox}$.

A further approximation that has been applied is the assumption that the intensity is constant through the depth of a single voxel, provided the voxel size remains small. This approximated integration was found to reduce artifacts in the reconstructed image, as compared to intensity calculated at the center of the voxel alone.

Acknowledgment

The authors would like to thank Enzo Reyes of VPAC (Victorian Partnership for Advanced Computing), for expert advice on GPU optimization and high performance computing in general, and for reviewing versions of this paper. The authors would also like to thank NVIDIA for the award of a professor partnership grant.

References

[1] S. Wilderman, N. Clinthorne, J. Fessler, C.-H Hua, and W. Rogers, "List mode EM reconstruction of Compton scatter camera images in 3-D," in *2000 IEEE Nuclear Science Symp. Conf. Record*, 2000, vol. 2, pp. 15/292–15/295.

[2] M. Dimmock, J. Gillam, T. Beveridge, J. Brown, R. Lewis, and C. Hall, "A pixelated emission detector for RadiOisotopes (PEDRO)," *Nucl. Instrum. Meth. Phys. Res. A*, vol. 612, pp. 133–137, 2009.

[3] C. Nguyen, J. Gillam, J. Brown, D. Martin, D. Nikulin, and M. Dimmock, "Towards optimal collimator design for the PEDRO hybrid imaging system," *IEEE Trans. Nucl. Sci.*, vol. 58, no. 3, pp. 639–650, Jun. 2011.

[4] M. Rosenthal, J. Cullom, W. Hawkins, S. Moore, B. Tsui, and M. Yester, "Quantitative SPECT imaging: A review and recommendations by the focus committee of the society of nuclear medicine computer and instrumentation council," *J. Nucl. Med.*, vol. 36, no. 8, p. 1489, 1995.

[5] J. Dobbins 3rd and D. Godfrey, "Digital X-ray tomosynthesis: current state of the art and clinical potential," *Phys. Med. Biol.*, vol. 48, no. 19, p. R65, 2003.

[6] D. Wilson, H. Barrett, and E. Clarkson, "Reconstruction of two- and three-dimensional images from synthetic-collimator data," *IEEE Trans. Med. Imag.*, vol. 19, no. 5, pp. 412–422, 2000.

[7] F. Beekman and B. Vastenhouw, "Design and simulation of a high-resolution stationary SPECT system for small animals," *Phys. Med. Biol.*, vol. 49, pp. 4579–4579, 2004.

[8] H. O. Anger, "Use of a gamma-ray pinhole camera for in vivo studies," *Nature*, vol. 170, pp. 200–201, Aug. 1952.

[9] H. Barber *et al.*, "Semiconductor pixel detectors for gamma-ray imaging in nuclear medicine," *Nucl. Instrum. Meth. Phys. Res. Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 395, no. 3, pp. 421–428, 1997.

[10] G. Pratx and C. Levin, "Bayesian reconstruction of photon interaction sequences for high-resolution PET detectors," *Phys. Med. Biol.*, vol. 54, p. 5073, 2009.

[11] S. Metzler, J. Bowsher, K. Greer, and R. Jaszczak, "Analytic determination of the pinhole collimator's point-spread function and RMS resolution with penetration," *IEEE Trans. Med. Imag.*, vol. 21, no. 8, pp. 878–887, 2002.

[12] M. Williams, A. Stolin, and B. Kundu, "Investigation of efficiency and spatial resolution using pinholes with small pinhole angle," *IEEE Trans. Nucl. Sci.*, vol. 50, no. 5, pp. 1562–1568, 2003.

[13] S. Metzler, J. Bowsher, M. Smith, and R. Jaszczak, "Analytic determination of pinhole collimator sensitivity with penetration," *IEEE Trans. Med. Imag.*, vol. 20, no. 8, pp. 730–741, 2001.

[14] G. Jung, A. Berry, S. Midgley, and G. Panjkovic, "Testing of pixellated CzT and CdTe detectors at the 200μm level," in *Proc. 2009 12th Int. Symp. IEEE Integrated Circuits, ISIC'09*, pp. 187–190.

[15] M. Chmeissani *et al.*, "First experimental tests with a CdTe photon counting pixel detector hybridized with a Medipix2 readout chip," *IEEE Trans. Nucl. Sci.*, vol. 51, no. 5, pp. 2379–2385, 2004.

[16] K. Lange and R. Carson, "EM reconstruction algorithms for emission and transmission tomography," *J. Comput. Assist. Tomogr.*, vol. 8, pp. 306–316, 1984.

[17] L. Parra and H. Barrett, "List-mode likelihood: EM algorithm and image quality estimation demonstrated on 2-D PET," *IEEE Trans. Med. Imag.*, vol. 17, no. 2, pp. 228–235, 1998.

[18] A. Rahmim, M. Lenox, A. Reader, C. Michel, Z. Burbar, T. Ruth, and V. Sossi, "Statistical list-mode image reconstruction for the high resolution research tomograph," *Phys. Med. Biol.*, vol. 49, pp. 4239–4239, 2004.

[19] J. Stone, D. Gohara, and G. Shi, "OpenCL: A parallel programming standard for heterogeneous computing systems," *Comput. Sci. Eng.*, vol. 12, p. 66, 2010.

[20] The OpenCL Specification, version 1.0.29. Khronos OpenCL Working Group, Dec. 8, 2008 [Online]. Available: http://khronos.org/registry/cl/specs/opencl-1.0.29.pdf

[21] K. Karimi, N. G. Dickson, and F. Hamze, "A performance comparison of CUDA and OpenCL," *CoRR*, 2010, Abs/1005.2581.

[22] K. Komatsu, K. Sato, Y. Arai, K. Koyama, H. Takizawa, and H. Kobayashi, "Evaluating performance and portability of OpenCL programs," in *Proc. Fifth Int. Workshop on Automatic Performance Tuning (iWAPT2010)*, 2010.

[23] A. Barak, T. Ben-Nun, E. Levy, and A. Shiloh, "A package for OpenCL based heterogeneous computing on clusters with many GPU devices," in *Proc. 2010 IEEE Int. Conf. Cluster Computing Workshops and Posters (CLUSTER WORKSHOPS)*, Sep. 2010, pp. 1–7.

[24] R. Aoki, S. Oikawa, T. Nakamura, and S. Miki, "Hybrid OpenCL: Enhancing OpenCL for distributed processing," in *Proc. 2011 IEEE 9th Int. Symp. Parallel and Distributed Processing with Applications (ISPA)*, May 2011, pp. 149–154.

[25] V. Nguyen, S. Lee, and M. Lee, "GPU accelerated statistical image reconstruction for Compton cameras," in *2009 IEEE Nuclear Science Symp. Conf. Record (NSS/MIC)*, 2010, pp. 3550–3555.

[26] G. Pratx, G. Chinn, P. Olcott, and C. Levin, "Fast, accurate and shift-varying line projections for iterative reconstruction using the GPU," *IEEE Trans. Med. Imag.*, vol. 28, no. 3, pp. 435–445, 2009.

[27] T. Budinger, S. Derenzo, G. Gullberg, W. Greenberg, and R. Huesman, "Emission computer assisted tomography with single-photon and positron annihilation photon emitters," *J. Comput. Assist. Tomogr.*, vol. 1, no. 1, p. 131, 1977.

[28] J. Cui, G. Pratx, S. Prevrhal, L. Shao, and C. Levin, "Fully 3-D list-mode positron emission tomography image reconstruction on gpu using cuda," in *2010 IEEE Nuclear Science Symp. Conf. Record (NSS/MIC)*, 2010, pp. 2635–2637.

[29] S. Agostinelli *et al.*, "Geant4-A simulation toolkit," *Nucl. Instrum. Meth. Phys. Res. A*, vol. 506, no. 3, pp. 250–303, 2003.

[30] A. Reader, S. Ally, F. Bakatselos, R. Manavaki, R. Walledge, A. Jeavons, P. Julyan, S. Zhao, D. Hastings, and J. Zweit, "One-pass list-mode em algorithm for high-resolution 3-D PET image reconstruction into large arrays," *IEEE Trans. Nucl. Sci.*, vol. 49, no. 3, pp. 693–699, Jun. 2002.

[31] A. Reader, P. Julyan, H. Williams, D. Hastings, and J. Zweit, "EM algorithm system modeling by image-space techniques for PET reconstruction," *IEEE Trans. Nucl. Sci.*, vol. 50, no. 5, pp. 1392–1397, 2003.