Rapid processing of PET list-mode data for efficient uncertainty estimation and data analysis

2016 Phys. Med. Biol. 61 N322

(http://iopscience.iop.org/0031-9155/61/13/N322)

OPEN ACCESS
IOP Publishing | Institute of Physics and Engineering in Medicine Physics in Medicine & Biology
Phys. Med. Biol. **61** (2016) N322–N336 doi:10.1088/0031-9155/61/13/N322

Note

Rapid processing of PET list-mode data for efficient uncertainty estimation and data analysis

P J Markiewicz[1,2], K Thielemans[2], J M Schott[3], D Atkinson[4], S R Arridge[5], B F Hutton[2] and S Ourselin[1,3]

[1] Translational Imaging Group, CMIC, University College London, London, UK
[2] Institute of Nuclear Medicine, University College London, London, UK
[3] Dementia Research Centre, University College London, London, UK
[4] Centre for Medical Imaging, University College London, London, UK
[5] Centre for Medical Image Computing, University College London, London, UK

E-mail: p.markiewicz@ucl.ac.uk

Received 15 August 2015, revised 29 January 2016
Accepted for publication 12 May 2016
Published 9 June 2016

Abstract

In this technical note we propose a rapid and scalable software solution for the processing of PET list-mode data, which allows the efficient integration of list mode data processing into the workflow of image reconstruction and analysis. All processing is performed on the graphics processing unit (GPU), making use of streamed and concurrent kernel execution together with data transfers between disk and CPU memory as well as CPU and GPU memory. This approach leads to fast generation of multiple bootstrap realisations, and when combined with fast image reconstruction and analysis, it enables assessment of uncertainties of any image statistic and of any component of the image generation process (e.g. random correction, image processing) within reasonable time frames (e.g. within five minutes per realisation). This is of particular value when handling complex chains of image generation and processing.

The software outputs the following: (1) estimate of expected random event data for noise reduction; (2) dynamic prompt and random sinograms of span-1 and span-11 and (3) variance estimates based on multiple bootstrap realisations of (1) and (2) assuming reasonable count levels for acceptable accuracy. In addition, the software produces statistics and visualisations for immediate quality control and crude motion detection, such as: (1) count rate curves; (2) centre of mass plots of the radiodistribution for motion detection;

(3) video of dynamic projection views for fast visual list-mode skimming and inspection; (4) full normalisation factor sinograms. To demonstrate the software, we present an example of the above processing for fast uncertainty estimation of regional SUVR (standard uptake value ratio) calculation for a single PET scan of ^{18}F-florbetapir using the Siemens Biograph mMR scanner.

Keywords: posistron emission tomography, graphics processing unit, list mode, motion detection, projection views, histogramming, fast data processing

(Some figures may appear in colour only in the online journal)

1. Introduction

In positron emission tomography (PET) each detected coincidence event (a photon pair) can be stored in a list-mode data format using data packets which contain the necessary information for subsequent image formation. Each data packet contains information about the nature of the event, (i.e. whether it is a prompt or delayed event) and the detector pair address of the line of response (LOR) along which a photon pair was detected. Also, detector single rates and time markers are stored in dedicated data packets. In this work, we refer to the list-mode event packets as containing only the bin address corresponding to a given crystal pair with some information about timing and energy of the detected photon pair being lost after event positioning.

Storing and processing list-mode data is usually preferred in comparison to sinograms as the raw nature of the list-mode preserves all the spatio-temporal information which is partially lost using the sinogram format (e.g. it is difficult to correct for motion that occurred within the time frame of the sinogram). Also, in some instances where the time of flight (TOF) information is available and where the total number of events detected is smaller then the number of all possible LORs, list-mode format is more compact (Matej *et al* 2009).

The purpose of this work is to provide an open source software solution enabling very fast list-mode data processing which allows practical and efficient generation of multiple bootstrap realisations of image datasets being processed within arbitrarily complex reconstruction and analysis chains (Markiewicz *et al* 2015). Based on these datasets, distributions of any statistic can be formed indicating the uncertainty of given parameter of interest which can be used, for example, in the quantitative image analysis (Verhaeghe *et al* 2010) or in the study of discriminative power of amyloid imaging (Herholz *et al* 2014). Certain properties of such distributions can also be found using analytical and fully Bayesian methods on sinogram and list-mode data (Barrett *et al* 1994, Fessler and Rogers 1996, Sitek 2012), however with some approximations which may not always hold, especially in cases of further image processing (e.g. spatial image registration between MR and PET images, feature extraction and classification). In addition, this software enables real-time and concurrent processing used for estimating mean random events in each bin, motion detection and generating other informative statistics of the PET acquisition.

The presented methodology of data processing can be implemented on most multi-threading architectures like multi-core CPU and GPU devices. Although, many aspects of the methodology is applicable to multi-threaded CPUs, greater emphasis is put on graphics processor units (GPU) as they are optimised for operating on datasets by performing very fast the same kernel functions accessing the data elements in a continuous manner (e.g. 3D image rendering). PC microprocessors, on the other hand, are optimised for general computing with

random/irregular access to different parts of the data. A multi-threaded execution model was used while addressing efficiently the common bottleneck of data transfers between a hard disk, the CPU (host) memory and GPU (device) memory. Since image reconstruction and analysis can benefit from GPU computing, it seems reasonable to also exploit the computational powerhouse for processing of list mode datasets. Processing of the list-mode data is performed without perceivable delay relative to data transfer times with the following outcomes:

- Reduced-noise random event data estimated for each LOR using crystal fan-sums obtained from the delayed coincidences (Hogg *et al* 2002, Panin *et al* 2007).
- Static and dynamic sinograms of span-1 and span-11 for random and prompt data (Defrise and Kinahan 1998).
- Multiple bootstrap replications of prompt and random data which are used for generating a voxel level distribution of any image statistic for the assessment of noise and uncertainty.
- Plots of the variation of the centre of the radiodistribution mass due to kinetics and motion which involves crude motion detection.
- Visual inspection of projection views (sagittal and coronal) produced in video format, useful for motion detection and for general inspection of data quality.
- The total count-rate data (also known as the head curve) which reports total prompts, randoms and single rates over time.
- Full normalisation factor sinograms, including detector dead-time correction which is based on the measurement of single rates for each detector bucket.

2. Materials and methods

2.1. The GPU device

All the list mode data processing and the subsequent image reconstruction is performed on the GPU device. The device used in this work was the NVIDIA Tesla K20 (NVIDIA's compute capability of 3.5) consisting of 5 GB of on-board memory, 13 streaming multiprocessors, each with 192 single-precision CUDA cores (2496 in total). All the GPU resources are available within the compute unified device architecture (CUDA) platform version 7.0. CUDA kernels (CUDA extended C functions) are executed by multiple threads organised into thread blocks in which each thread can be identified by three indices (x, y, z) with the maximum total number of threads per block limited to 1024. Any set of 32 threads (called a warp) is executed in parallel. All threads in a single block can share data through the very fast shared memory, whereas threads outside given block can share data through the slower but much larger global memory. These blocks are further arranged into a 3D grid which can be indexed by three indices (x, y, z) with the total number of blocks equal to 2147483647.

2.2. Output data

Span-1 sinogram bins are accessed using 30-bit address in each event data packet (Jones 2013). For axially compressed sinograms of span-11, a look-up table is used to perform axial grouping yielding 837 sinograms out of 4084 span-1 sinograms (see table 1). The file size of span-1 sinogram is around 1.4 GB, compared to 290 MB for span-11 (with crystal gaps included in the sinograms).

Table 1. Siemens Biograph mMR sinogram dimensions for maximum ring difference MRD = 60.

Native mMR sinogram dimensions			
Span	Projection bins	Views	Sinograms
Span-1	344[a]	252	4084
Span-11	344[a]	252	837

[a] The bins are interleaved between two different angular views.

The output of all the GPU processing remains in the device memory for subsequent image reconstruction using a fast forward and back-projector (Markiewicz *et al* 2014). The sinograms can also be written to disk in the ECAT8 file format compatible with the vendor's software for further image reconstruction. For inspection purposes, sinograms in span-1 and span-11 can be exported to files in NIfTI format and viewed in most medical image visualisation software. Currently, the proposed software is dedicated to the PETLINK^{TM} list-mode data format and is developed as part of a software platform for Siemens Biograph mMR (Delso *et al* 2011) 3D and 4D PET image reconstruction. It is freely accessible to the community as open source, available at http://cmictig.cs.ucl.ac.uk/people/research-staff/pawel-markiewicz

2.3. Concurrent processing

The key to rapid, and essentially real time, processing of the list-mode data on the GPU is the division of the whole data into data chunks which are executed independently and asynchronously, starting with the first data chunks transferred to the GPU. The list-mode data is divided into chunks on the host with a predefined number of list mode data packets processed by each thread. Furthermore, for 60 min florbetapir brain scan, for which the list-mode file size averages around 10 GB, such a division is even more justified in cases where the device memory is not large enough to deal with all the data at once.

The division into smaller data chunks enables overlapping GPU kernel execution, with data transfers from disk to the CPU memory, and from the CPU memory to the GPU memory. For this to happen, firstly, the GPU device has to be capable of concurrent data copy and execution (NVIDIA 2015). Secondly, the overlapping can be managed only through multiple CUDA streams (sequences of operations issued by the host for execution on the device). Thirdly, the CPU memory from which the transfer to the GPU device memory occurs, has to be pinned memory which is page-locked memory (as opposed to host data which is pageable by default) acting as a buffer from which only the device can access the data (Harris 2012a, 2012b).

To each CUDA stream, one list-mode data chunk $L[n]$, of approximate size of 50 MB, is reserved for transferring and processing. In the used GPU device there are $S = 32$ possible streams meaning that the device can concurrently execute a maximum S kernel launches with overlapped memory transfers of S data chunks. The overlapping is shown in figure 1 where after transferring the first data chunk ($n = 1$) by the host-to-device engine and stream $s = 0$, asynchronous kernel execution of data chunks processing begins. Once the processing of a data chunk within any given stream is finished the stream can be reused again for reading from disk and copying to the memory for further processing until all data chunks N are processed. In addition to concurrent execution, an advantage of using streams is that they can be interleaved.

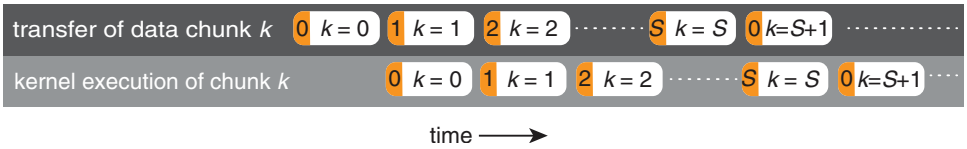

Figure 1. Time-line of overlapped transfer and kernel execution of list-mode data using CUDA S streams, each asynchronously processing list-mode data chunk k. The stream number is depicted in orange boxes with the data chunk number in the white boxes. For large list-mode data files $K > S$.

2.4. CUDA workflow of list-mode data processing

The whole workflow for reading, transferring and processing of the list-mode data is depicted in figure 2. The data chunks $L[n]$ are read to a buffer (pinned memory) of modifiable size chosen for optimal performance for a given CPU/GPU setup (in this work it was 1.5 GB). The buffer is further divided into S chunks corresponding to the maximum number of concurrent streams. The first S chunks are read from disk to the buffer before the concurrent data transfer to the device memory and overlapping kernel execution is launched. Each data chunk is processed by a grid of 10 blocks, each containing 256 threads (this can be modified for fine tuning), with 22 local registers per thread, giving rise to 100% occupancy of all the streaming multiprocessors. The next data chunks are read and processed within the *for* loop once any of the previous stream becomes available again. To achieve seamless processing of the whole dataset, stream synchronisation is used for constant influx of new list mode data chunks ready for execution. The host is notified about finished processing in any stream by the use of a callback function which reads next data chunk from the disk. Once data chunk $L[n]$ is read to the buffer, a flag is set notifying the launcher that the next data chunk is ready in $b[s]$ for processing by a free stream s.

2.5. Estimation of random event data

The random coincidences in each sinogram bin are measured through the delayed time window method. However, this measurement is usually very noisy potentially having an adverse effect on the reconstructed image (Hogg *et al* 2002). The noise can be reduced using variance reduction methods exploiting the model for random data in the LOR formed between crystals i and j:

$$R_{ij} = 2\tau S_i S_j, \tag{1}$$

where τ is the coincidence time window, S_i and S_j are the singles rates at the two crystals. Since the measured delayed window data can be described using Poisson statistics, the expected values of random data in any LOR can be found using the maximum likelihood (ML) approach with the model for the expected data in (1). Therefore, the log-likelihood function becomes:

$$L(\mathbf{S}) = \frac{1}{2}\sum_{i,j\in\mathbb{J}_i}\{r_{ij}\log(2\tau S_i S_j) - 2\tau S_i S_j\}, \tag{2}$$

where r_{ij} is the measured delayed data between crystals i and j, and $\mathbb{J}_i$ is the set of opposing crystals j which are in coincidence with crystal i. In order to enable simultaneous update of all crystal single rates S_i, the method of Panin *et al* (2007) was adopted, where the single rates are separated by the use of a surrogate function, which is formed using the inequality $S_i S_j \leqslant \frac{1}{2}(S_i^2 + S_j^2)$. Maximising the surrogate leads to the update equation for iteration $(k+1)$:

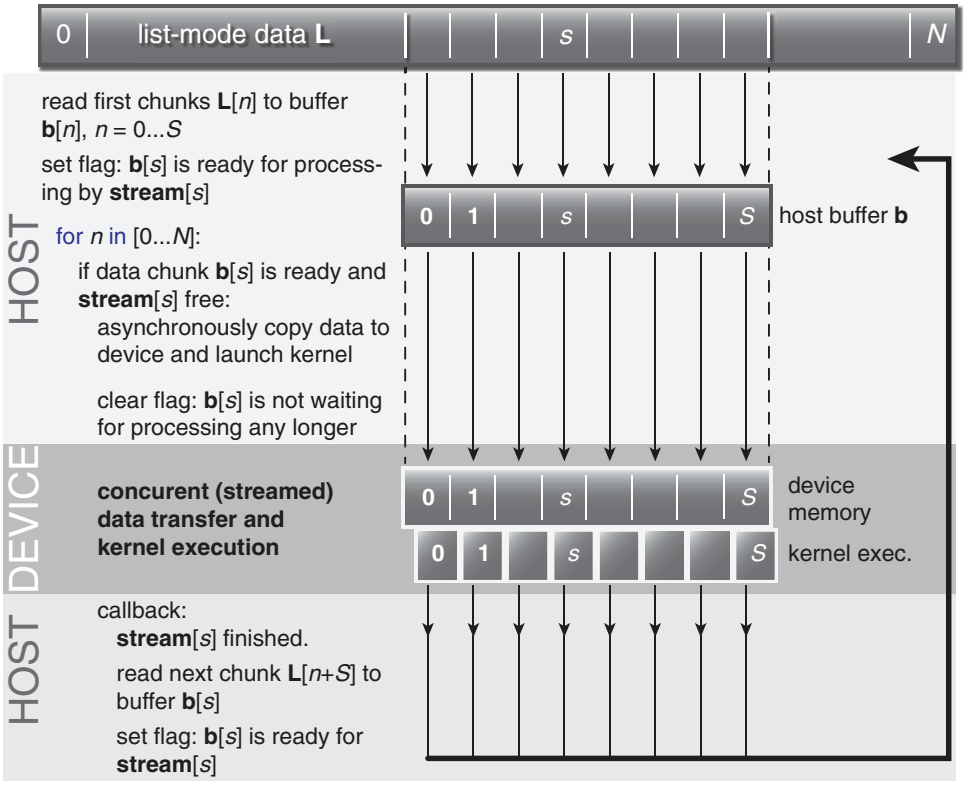

Figure 2. Workflow of the list-mode data processing. The list-mode data is read in data chunks $L[n]$ to host buffer (pinned memory) $b[s]$. Each stream s is continually checked in the *for* loop for its availability together with the availability of new data chunk in $b[s]$. When a stream and data chunk are available, the chunk is concurrently copied to the device while other chunks are being processed. By the end of kernel execution of data chunk $L[n]$, the next data chunk is read from disk and made available for transfer and execution on the device.

$$S_i^{(k+1)} = \frac{1}{2} S_i^{(k)} + \frac{1}{2} \frac{\sum_{j \in \mathbb{J}_i} r_{ij}}{\sum_{j \in \mathbb{J}_i} S_j^{(k)}} \tag{3}$$

The terms in the numerator of equation (3), $\sum_{j \in \mathbb{J}_i} r_{ij}$, are the fan sums of all random events recorded along those LORs which are formed by crystal i and opposing crystal $j \in \mathbb{J}_i$. The fan sums are found on the fly when processing the delayed events via decomposition of the sinogram address into ring and crystal indices for each delayed event. The events are then accumulated for static or dynamic time frames in crystal maps of fan sums (figure 3, right) using the CUDA atomic add operator to avoid rare but possible race conditions. The way the fan sums are found is shown in figure 3(left) for the transaxial plane and crystal i with a corresponding mark on the crystal map of unrolled ring of detectors (right figure).

The sum over single rates in the denominator of equation (3) has to be found for each crystal and any iteration k. To ensure fast sum calculations (reductions) inter-thread communication is exploited by the use of *shuffle instructions* introduced in the Kepler architecture (NVIDIA 2012). The shuffle instructions permit exchanging variables within a warp (a group of 32 threads executed in parallel). Such variable sharing is significantly faster when

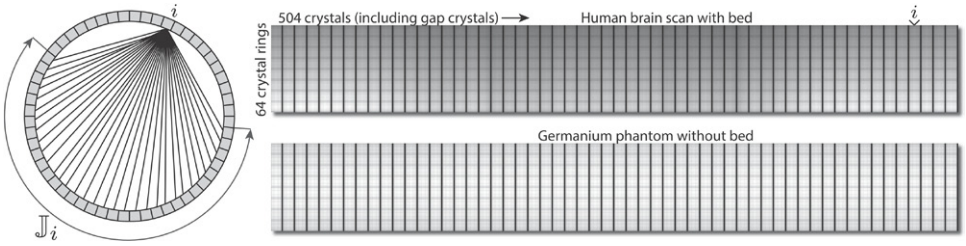

Figure 3. Fan sums of the measured delayed events. Left: transaxial view of the fan sums for crystal i and the opposing crystals $\mathbb{J}_i$ creating the fan on the ring of 56 detector blocks. Right: two crystal maps of fan sums for a brain scan with the bed in the field of view (top right) and for a germanium phantom without the bed (bottom right). The crystal map can be viewed as the unrolled ring of detectors.

performing reductions as it reduces the use of shared memory and additional synchronisation (see NVIDIA (2015)).

The reductions are first performed axially for 64 rings by two CUDA warps for a given transaxial position, and then the next 64 axial crystals for another fixed transaxial position. This process of reduction is repeated until all transaxial positions are considered. Shared memory is used for further reducing the above partial warp sums. The use of shared memory is however limited to a block of 1024 threads and a *for* loop is used to cover all crystal rings (there are less than 20 000 crystals in coincidence with any other opposing crystal). Once an estimate of the single rates has been found for each crystal, a sinogram of expected random events may be formed based on equation (1).

2.6. Bootstrap replications

Within each CUDA thread, the kernel can also generate a number of statistical realisations of the list-mode data through non-parametric bootstrap resampling with replacement from a chosen pool of prompt and delayed events (Markiewicz *et al* 2015). The random access to the memory due to the bootstrap resampling causes some drop of kernel performance (the list-mode data processing rate of 0.47 GB s^{-1} drops to 0.42 GB s^{-1}). The random access to the memory can be avoided, leaving the performance unaffected, if parametric bootstrap is used instead with parameter $\lambda = 1$ for each recorded event (Haynor and Woods 1989). The size of the resampling pool depends on the chosen grid of blocks and threads which deals with each data chunk $L[k]$, and in this work it was around 4500 events resampled by each thread using the NVIDIA CUDA Random Number Generation library (cuRAND). The resampled events are histogrammed on the fly in the device kernel. Note that for successive bootstrap replications the list mode data chunks are not read from the disk but from the cache memory making the processing considerably faster by removing the bottleneck of disk access for reading the list-mode data.

It is worth noting, that the division into data chunks is also necessary for generating bootstrap replicates of the list-mode datasets (Markiewicz *et al* 2015). Through such division all the dynamics recorded in the LM dataset are preserved including the correspondence between all the time tags, prompt, delayed events and time-varying detector single rates tags.

2.7. Dynamic projection view analysis

In addition to the above histogramming, for visual inspection purposes, the CUDA kernel rebins the data on the fly to direct and cross sinograms through the single slice rebinning

(SSRB) method (Daube-Witherspoon and Muehllehner 1987), which enables using time frames of 12 s or similar with reasonable statistics for axial views (sagittal and coronal). This allows the short dynamic frames of the sagittal and coronal views to be exported to a video which can be used for a quick quality control of the acquired data. The duration of the frames may vary and can be independent from the dynamic time frames used for kinetic analysis.

Based on the above SSRB projection data, it is also possible to get crude motion detection and estimates of the displacement through the use of the centre of projection distribution mass $C_t^{(z)}$ according to:

$$C_t^{(z)} = \frac{1}{P_t}\sum_{i}^{I} p_{it} z_i, \tag{4}$$

where P_t is the total number of prompts detected in a time frame t, I is the total number of direct and cross sinograms ($I = 127$) after SSRB, p_{it} is the number of events recorded in sinogram i in time frame t and z_i is the axial coordinate of the ith sinogram. Note that due to the low statistics in short time frames the centre of mass is limited to the axial direction (z).

2.8. Histogramming

The prompt and delayed events are histogrammed to corresponding sinogram bins with varying axial angular grouping. In span-11, five or six (depending on the position within the FOV) positive and negative segment numbers are combined, thus significantly reducing the number of sinograms (see table 1).

Fast atomic add operations are used for histogramming to avoid race condition in case when two CUDA threads are accessing the same sinogram bin for updating the bin value (nevertheless, it is very unlikely to get such conflicts and hence the performance is not significantly affected). The static sinogram bin values are well represented by 16-bit integer values, however, the atomic operations use 32-bit or 64-bit integers only. Hence the 32-bit word is divided into two parts, with the lower part used by prompt events and the higher part being used by the delayed events. For dynamic sinograms it is enough to represent each sinogram bin by just 8 bits, thus, two prompt and two delayed bins are accumulated and stored in one 32-bit word. Nevertheless, such compressed sinogram with 30 dynamic frames will not fit into 5 GB of memory and therefore histogramming is performed separately in two batches of 15 sinogram frames each.

Total count-rate curves are calculated and stored for each second of acquisition including total prompts and delayed events. Singles rates are extracted from dead-time tracking packets which are reported for each detector bucket consisting of two detector blocks transaxially.

2.9. Detector dead time and normalisation.

In any realistic detection process, there is always a minimum amount of time called the dead time that is required between two separate events so that they will be recorded as two distinct events. Therefore, the efficiency of detector i due to dead time will be dependent on the detector count rate (Casey *et al* 1996), i.e.:

$$\epsilon_i^{(\mathrm{dt})} = \exp\left(-\frac{S_i t_{\mathrm{p}}}{1 + S_i t_{\mathrm{np}}}\right)/(1 + S_i t_{\mathrm{np}}), \tag{5}$$

where S_i is the block single rate, t_{p} and t_{np} are the paralysing and non-paralysing components (they are constant for all detector rings) obtained from the component-based normalisation file written out by the scanner for each acquisition . The single rates S_i are measured and reported in the list-mode data for each detector bucket which consists of two detector blocks transaxially. In total there are 224 buckets (28 transaxially $\times$ 8 axially).

Sinograms of normalisation factors are calculated if the file with normalisation components is provided. The efficiency ϵ_b for a given LOR is decomposed into components including (1) geometric factors, $\epsilon_b^{(\mathrm{g})}$, (2) crystal interference, $\epsilon_b^{(\mathrm{i})}$, (3) crystal efficiencies $\epsilon_i^{(\mathrm{e})}$, (4) axial effects (which include axial block and geometric factors for span-11), $\epsilon_b^{(\mathrm{a})}$, (5) paralysing ($t_{\mathrm{p}}$) and non-paralysing ($t_{\mathrm{np}}$) dead time parameters (see equation (5)), leading to

$$\epsilon_b = \epsilon_b^{(\mathrm{g})}\epsilon_b^{(\mathrm{i})}\epsilon_b^{(\mathrm{a})}\epsilon_i^{(\mathrm{e})}\epsilon_j^{(\mathrm{e})}\epsilon_i^{(\mathrm{dt})}\epsilon_j^{(\mathrm{dt})}. \tag{6}$$

2.10. Brain scan example

The processing of the list-mode data is shown on an example of real brain scan of [^{18}F]-florbetapir, acquired dynamically for 50 min on the Siemens Biograph mMR PET-MR scanner resulting in 8 GB list-mode data file. Simultaneously to the PET acquisition, T1 weighted MR images were acquired which were used for regional segmentation of the PET image after spatially registering the MR image to the PET image. To obtain SUVR values in different grey matter regions, the regions have to be normalised to the average uptake in cerebellar grey matter. Note that despite the data is acquired simultaneously, there is need for spatial registration due to small patient movements. It is very likely that motion will occur between the T1 weighted acquisition at the beginning of the 50 min scan and the last 10 min, which are used in the list-mode processing and bootstrapping.

3. Results and discussion

3.1. Estimation of random events

The use of simultaneous update algorithm (Panin *et al* 2007) with 10 iterations of the algorithm expressed by (3) makes it possible to find the estimated random sinogram for any span in 0.5–0.6 s using CUDA implementation. This timing is particularly useful for generating multiple bootstrap replications where all list-mode events are resampled and the whole process of random estimation is repeated for each realisation independently.

Figure 4 shows two plots of sinogram profiles of measured and estimated random events of the 10 min of the [^{18}F]-florbetapir scan. The left plot shows a profile for one direct sinogram row while the right plot shows the same profile but for all direct and oblique sinograms summed up for better statistics of the measured data, indicating considerable noise reductions with no significant bias.

3.2. Estimating SUVR uncertainties using the bootstrap

The ten minute list mode data was resampled with $B = 100$ bootstrap realisations. Due to the fast bootstrap data generation it was possible to investigate the noise properties of individual components (e.g. random data estimation, image co-registration, etc) of the imaging pipeline for SUVR estimation. Grey matter cerebellum was used as a reference region for the SUVR

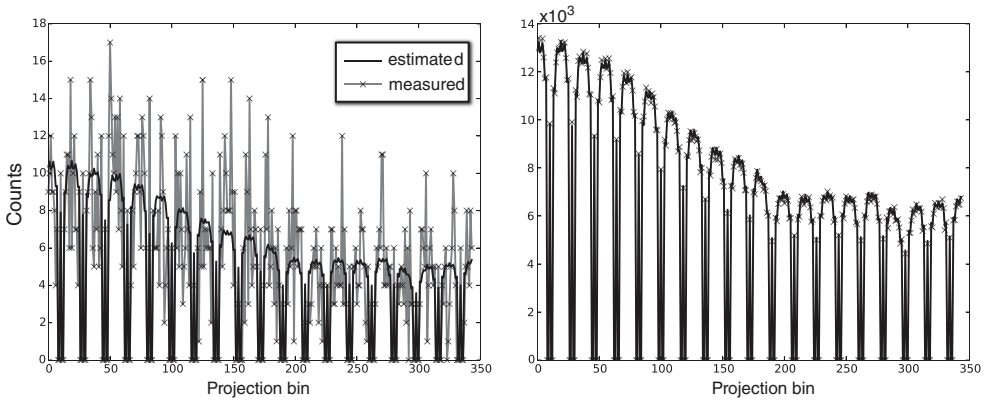

Figure 4. Sinogram profiles for measured and estimated random events (left) and a sum over all sinograms for the same sinogram profile location (right).

calculations after co-registration (Modat *et al* 2014) of T1 weighted image to each bootstrap PET image and propagation of regions of interest from the parcellated T1 weighted image (Cardoso *et al* 2012) to the PET native space. Image reconstruction was performed using the ordinary Poisson ordered subsets expectation maximisation algorithm (OP-OSEM (Comtat *et al* 2004)) within the off-line version of the Siemens Healthcare reconstruction software which was made available for this project. Each bootstrap histogramming took less than 2 seconds with additional 0.5 s used for estimating the mean random sinograms. The total time for each bootstrap SUVR realisation took four minutes, which included image reconstruction and co-registration.

Figure 5 presents standard error images in red with resampling performed for the delayed events separately (figure 5, left) in addition to the full resampling of delayed and prompt events (figure 5(A), middle). The distributions (uncertainties) of the SUVR for the region of cingulate gyrus (an important region in the study of Alzheimer's disease) were generated for the two cases of delayed and prompt resampling and shown in figure 5(B) below. As can be seen, the noise effects of delayed events are significantly reduced due to the noise reduced ML estimation of the randoms events. Also noted is the greater impact of prompt count statistics compared to the delayeds due to the fact that random events have smoother spatial representation and their variance is reduced through the ML estimate method). An example direct random data sinogram is shown in figure 6 with the corresponding standard error sinogram generated through the bootstrap resampling. Such resampling of all events in list mode data has been shown useful in estimating the effect of ^{18}F florbetapir dose reduction on classification between Alzheimer's diseased subjects and elderly controls, although the image co-registration was performed only for the original PET scans (Herholz *et al* 2014).

It has to be noted that the derived uncertainties through bootstrapping are estimated sampling errors caused by limited number of recorded events. Hence, the uncertainties are likely to be underestimated due to other effects, which bootstrap resampling cannot model such as errors in event positioning from detector readouts, normalisation, attenuation, scatter and any other systematic errors. These systematic errors, however, can be estimated through real phantom scans and Monte Carlo simulations. Furthermore, for reasonably accurate variance estimates it has to be ensured that the original dataset has high enough number of recorded

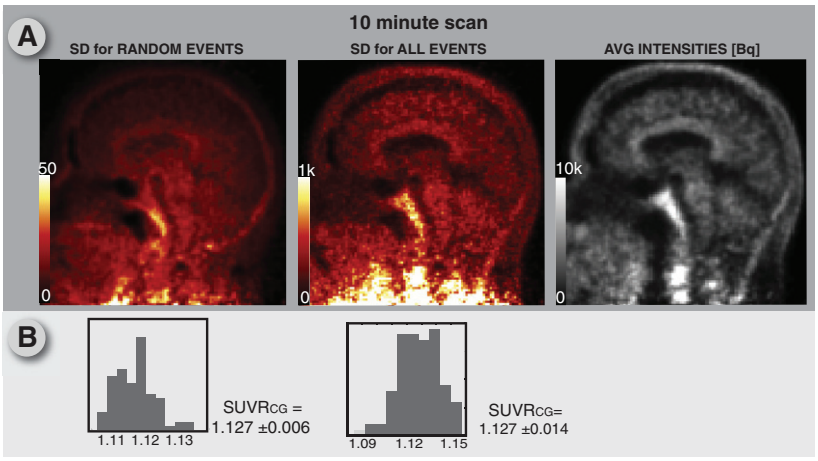

Figure 5. Estimated uncertainties at voxel levels (row A) of 10 min PET scan of [^{18}F] Florbetapir. The standard deviation images are shown in red while in grey the mean values which are almost identical to the original image (not resampled) are shown. In row (B) are plotted histograms of the SUVR for the whole cingulate gyrus when bootstrap sampling is performed on the random (left) and prompts and random (right) events. The SUVR values with the standard deviations are also given.

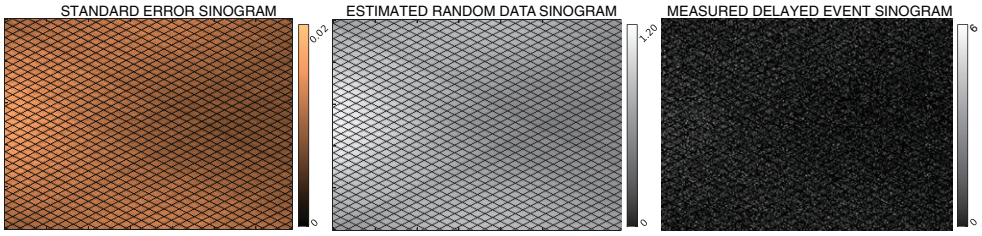

Figure 6. Measured delayed event sinogram (right), reduced variance estimate of sinogram random events (middle) and corresponding uncertainties (standard errors) estimated by the bootstrap (left).

counts for each resampling time frame while keeping the time frames short enough to account for radiotracer dynamics (Lartizien *et al* 2010, Markiewicz *et al* 2015).

3.3. Visual inspection of dynamic projection views

The inspection is performed for the whole duration of the scan (50 min) with time resolution of 12 s per frame (the duration can be changed) for which two projection views, sagittal and coronal, are extracted based on the SSRB method. The frames are then converted to video format together with the count-rate curves and a time marker making fast skimming through the list-mode data for quality control very straightforward. When the list-mode data is processed for dynamic reconstruction with variable duration time frames, the projection views for each frame are also exported to a video. The videos are available online for two cases (see figure 7): with considerable motion (https://vimeo.com/129831136 and https://

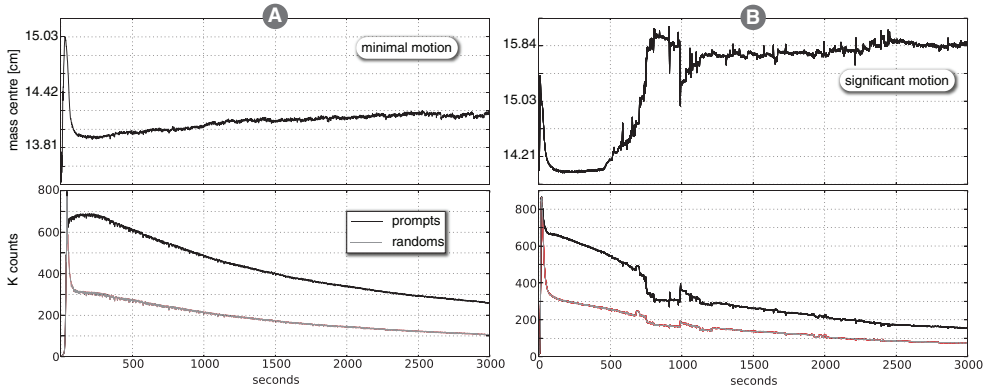

Figure 7. Example of count-rate (bottom) and centre of projection mass (top) curves for two cases: (A) a brain scan with little motion and (B) brain scan with considerable motion. Note that the motion effect is detected earlier in the centre of mass than in the count-rate curves. Also note that the centre of mass is given for the axial component only (z) due to poor statistics in the short time frames considered for motion detection.

vimeo.com/129831482) as well as with no or little motion (https://vimeo.com/129830997 and https://vimeo.com/129830998).

3.4. Total count-rate and centre of projection mass

Figure 7 shows the count-rate curves (head curves) of prompts and delayeds per second as well as the centre of projection mass curves for the whole scan duration (50 mins) and for two brain scans: with minimal and considerable motion. Note that effect of motion is detected earlier in the centre of mass curve (case 'B') than in the count-rate curve on which the motion has some effect too. Nevertheless, motion detection and some quantification are rather crude and serve as an indicator only. This method of mass centre has its limitations due to poor statistics in short time frames used for motion detection and currently it is limited to axial direction only. Also, interpretation of the centre of mass curve may be more difficult for non-rigid motion. However, this can be further refined, for example, by adopting the method of PCA on short time frame sinograms (Thielemans *et al* 2013).

3.5. Histogramming and detector normalisation

The prompt and delayed event sinograms for span-1, span-11 and full span are obtained and stored in the device memory in a compressed integer. No difference in performance was observed for different spans. They can be copied to the CPU memory, decompressed and stored to disk in the ECAT or NIfTI file format. Since the number of sinograms for dynamic studies is too large for 5 GB of device memory, the dynamic sinograms are found in two GPU kernel batches.

It has been observed that the biggest bottleneck of the histogramming process is the disk access to read the list-mode data. It is anticipated that any CPU or GPU implementations would equally suffer from this bottleneck. Therefore, it is advisable to use the transfer time for many other useful calculations like motion detection, random event estimation. If the whole

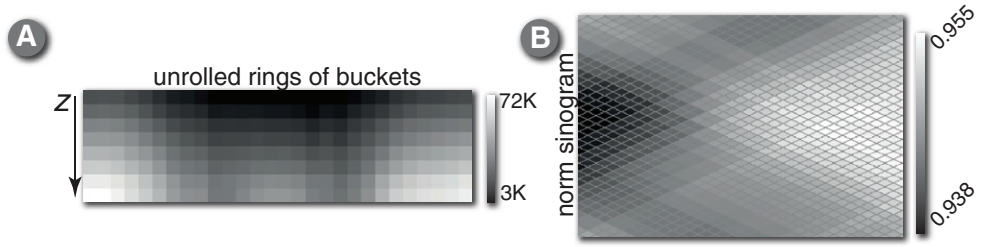

Figure 8. (A) Unrolled ring of detectors buckets [8×28] with single rates averaged over the whole duration (50 min) of a brain scan. (B) One of the sinograms with dead-time normalisation factors. Note that the lighter colour sinogram regions correspond to the LORs which are shielded by the bed and therefore receive lower rates of single events.

list-mode data is transferred to CPU memory, the processing is done at 0.47 GB s^{-1}, otherwise the rate is limited by the disk access.

With the scanner's normalisation components provided in a binary file, it is possible to output span-11 normalisation sinogram (axial components for span-1 are not provided). The necessary detector single rates are extracted from the list-mode data for each detector bucket and are used for determining bin dead-time efficiencies (figure 8 shows 28×8 buckets of unrolled detector ring with a direct sinogram of dead-time normalisation factors). The whole normalisation sinogram is calculated in less than 0.5 s.

4. Conclusion

The presented workflow of list-mode data processing enables very efficient generation of prompt sinogram, estimation of random event sinograms, crude motion detection and other useful list-mode statistics such as the head curve. All the processing is done concurrently while transferring the data from disk and the CPU memory to the device memory. The presented methodology is also applicable to multi-core CPU implementations, however the optimisation of some parameters would have to be different. Also, the presented software can be used with PET scanners using PETLINK^{TM} data format, however currently without the support for time of flight acquisitions, which will be added later. Such rapid processing is very helpful in fast creation of bootstrap realisations and multiple image reconstructions providing valuable insight into distribution of any statistic (e.g. the SUVR) or can be used in the development of new correction and reconstruction algorithms by investigating their average performance based on many random realisations. Such fast list-mode processing can also be advantageous for list-mode image reconstruction and data compression. The software using the presented workflow will be made available as open source.

Acknowledgments

The Tesla K20 used for this research was donated by the NVIDIA Corporation. The Florbetapir PET tracer was provided by AVID Radiopharmaceuticals (a wholly owned subsidiary of Eli Lilly & Co). Support for this work was received from the EPSRC (EP/H046410/1, EP/J020990/1, EP/K005278, EP/M022587/1), the MRC (MR/J01107X/1, CSUB19166), the EU-FP7 project VPH-DARE@IT (FP7-ICT-2011-9-601055), the NIHR Biomedical Research Unit (Dementia) at UCL and the National Institute for Health Research

University College London Hospitals Biomedical Research Centre (NIHR BRC UCLH/ UCL High Impact Initiative- BW.mn.BRC10269), the NIHR Queen Square Dementia BRU, Wolfson Foundation, ARUK (ARUK-Network 2012-6-ICE; ARUK-PG2014-1946), European Commission (H2020-PHC-2014-2015-666992), the Dementia Research Centre as an ARUK coordinating centre.

References

Barrett H H, Wilson D W and Tsui B M W 1994 Noise properties of the em algorithm. I. Theory *Phys. Med. Biol.* **39** 833

Cardoso M, Wolz R, Modat M, Fox N C, Rueckert D and Ourselin S 2012 Geodesic information flows *MICCAI* **7511** 262–70

Casey M E, Gadagkar H and Newport D 1996 A component based method for normalization in volume PET *Three-Dimensional Image Reconstruction in Radiology and Nuclear Medicine* ed P Grangeat and J L Amans (Dordrecht: Kluwer) pp 66–71

Comtat C, Bataille F, Michel C, Jones J, Sibomana M, Janeiro L and Trebossen R 2004 OSEM-3D reconstruction strategies for the ECAT HRRT *IEEE Nuclear Science Symp. Conf. Record* vol 6 pp 3492–6

Daube-Witherspoon M E and Muehllehner G 1987 Treatment of axial data in three-dimensional PET *J. Nucl. Med.* 28 1717–24 (PMID: 3499493)

Defrise M and Kinahan P 1998 The theory and practice of 3D PET *Developments in Nuclear Medicine* vol 32, ed B Bendriem and D Townsend (New York: Springer) pp 11–53

Delso G, Frst S, Jakoby B, Ladebeck R, Ganter C, Nekolla S G, Schwaiger M and Ziegler S I 2011 Performance measurements of the siemens mmr integrated whole-body pet/mr scanner *J. Nucl. Med.* **52** 1914–22

Fessler J and Rogers W 1996 Spatial resolution properties of penalized-likelihood image reconstruction: space-invariant tomographs *IEEE Trans. Image Process.* **5** 1346–58

Harris M 2012a How to optimize data transfers in CUDA C/C++ http://devblogs.nvidia.com/parallelforall/how-optimize-data-transfers-cuda-cc/

Harris M 2012b How to overlap data transfers in CUDA C/C++ http://devblogs.nvidia.com/parallelforall/how-overlap-data-transfers-cuda-cc/

Haynor D and Woods S 1989 Resampling estimates of precision in emission tomography *IEEE Trans. Med. Imaging* **8** 337–43

Herholz K, Evans R, Anton-Rodriguez J, Hinz R and Matthews J 2014 The effect of 18F-florbetapir dose reduction on region-based classification of cortical amyloid deposition *Eur. J. Nucl. Med. Mol. Imaging* **41** 2144–49

Hogg D, Thielemans K, Mustafovic S and Spinks T 2002 A study of bias for various iterative reconstruction methods in PET *IEEE Nuclear Science Symp. Conf. Record* vol 3 pp 1519–23

Jones W F 2013 *PETLINK—a Propsed Digital Interconnect Standard for Data Acquisition in Nuclear Medicine* (Konxville TN: Siemens Medical Solutions) (revision J1 edn)

Lartizien C, Aubin J B and Buvat I 2010 Comparison of bootstrap resampling methods for 3D PET imaging *IEEE Trans. Med. Imaging* **29** 1442–54

Markiewicz P J, Reader A J and Matthews J C 2015 Assessment of bootstrap resampling performance for PET data *Phys. Med. Biol.* **60** 279

Markiewicz P, Thielemans K, Ehrhardt M, Jiao J, Burgos N, Atkinson D, Arridge S R, Hutton B F and Ourselin S 2014 High throughput CUDA implementation of accurate geometric modelling for iterative reconstruction of PET data *IEEE Nuclear Science Symp. Conf. Record* vol 4 pp 0–3

Matej S, Surti S, Jayanthi S, Daube-Witherspoon M, Lewitt R and Karp J 2009 Efficient 3D TOF PET reconstruction using view-grouped histo-images: DIRECT-direct image reconstruction for TOF *IEEE Trans. Med. Imaging* **28** 739–51

Modat M, Cash D M, Daga P, Winston G P, Duncan J S and Ourselin S 2014 Global image registration using a symmetric block-matching approach *J. Med. Imaging* **1** 024003

NVIDIA 2012 NVIDA's Next Generation CUDA Compute Architecture: Kepler GK110 *White Paper*

NVIDIA 2015 CUDA C Programming Guide http://docs.nvidia.com/cuda/cuda-c-programming-guide/#asynchronous-concurrent-execution

Panin V, Chen M and Michel C 2007 Simultaneous update iterative algorithm for variance reduction on random coincidences in PET *IEEE Nuclear Science Symp. Conf. Record* vol 4 pp 2807–11

Sitek A 2012 Data analysis in emission tomography using emission-count posteriors *Phys. Med. Biol.* **57** 6779

Thielemans K, Schleyer P, Dunn J, Marsden P and Manjeshwar R 2013 Using PCA to detect head motion from PET list mode data *IEEE Nuclear Science Symp. and Medical Imaging Conf.* pp 1–5

Verhaeghe J, Gravel P and Reader A J 2010 Task-oriented quantitative image reconstruction in emission tomography for single- and multi-subject studies *Phys. Med. Biol.* **55** 7263