POLISH ACADEMY OF SCIENCES
INSTITUTE OF FUNDAMENTAL TECHNOLOGICAL RESEARCH

Ph.D. Dissertation

Image reconstruction and simulation of strip Positron Emission Tomography scanner using computational accelerators

Adam Strzelecki

Thesis Advisor: Prof. Piotr Białas

Kraków 2016

Abstract

Positron emission tomography (PET) is one of the medical imaging methods enabling determination of images representing the glucose metabolism in vivo. It is used to observe and diagnose body diseases, especially cancerous lesions in the human tissue. Together with other tomography techniques delivering morphologic and anatomic imagery, it greatly improves the medical diagnosis correctness and overall time to diagnosis.

With all their benefits PET devices are still very expensive and therefore their availability is limited. J-PET is a joint effort of many sciences and scientists to create more affordable PET device based on plastic scintillators, unique geometry, dedicated electronics and modern computing techniques.

This work describes derivation of a statistical model for J-PET device, together with accompanying simulation and image reconstruction algorithms. Additionally, it introduces the reader to PET operational and image reconstruction principles, and also serves as a brief guide to computer science specific subjects of parallelization, vectorization and computational accelerator architectures.

The description starts with a formulation of image reconstruction solutions for two specific 2D subproblems – 2D barrel of detectors with simulated system matrix, and 2D two-strip scanner with analytic approximation kernel. These two methods are later merged into a complete solution for 3D J-PET scanner. Each method is provided with a generic and general-purpose computing on graphics processing units (GPGPU) accelerated implementation. Finally, each implementation is benchmarked and its image reconstruction performance is

tested against subset of National Electrical Manufacturers Association (NEMA) standard measures.

The presented material comes from three years of research, studies, discussions and programming conducted by me at Jagiellonian University in Krakow in J-PET project. J-PET simulation and image reconstruction tools are a tangible result of my effort. These tools helped us to establish understanding on J-PET unique geometry and make important decisions throughout the project. Thanks to the original algorithms employing parallel computing on GPU devices, our device is able to deliver imaging results almost instantly.

Nevertheless, my work represents just a small but vital fraction of the whole process required to make J-PET device functional – an image reconstruction taking use of the whole information provided by the other device components. Making this unique PET scanner would not be possible without the amazing collaboration of the whole team and leveraging and combining modern technologies from many sciences.

Abstrakt

Pozytonowa tomografia emisyjna (PET) jest jedną z technik obrazowania medycznego, która pozwala generować obraz reprezentujący metabolizm glukozy w organizmie. Jest ona używana do obserwacji i diagnozowania chorób, a w szczególności zmian nowotworowych. Wraz z innymi technikami tomografii dostarczającymi obrazy morfologiczne i anatomiczne, PET znacznie polepsza poprawność i czas stawiania diagnozy medycznej.

Ze wszystkimi swoimi zaletami urządzenia PET są niestety nadal bardzo drogie, a w związku z tym ich dostępność jest ograniczona. Projekt J-PET to połączony wysiłek wielu nauk i naukowców w celu stworzenia bardziej przystępnego tomografu PET bazującego na plastikowych scyntylatorach, unikalnej geometrii, specjalizowanej elektronice oraz nowoczesnych technikach obliczeniowych.

Rozprawa opisuje wyprowadzenie modelu statystycznego dla tomografu J-PET, wraz z towarzyszącymi mu algorytmami symulacji i rekonstrukcji obrazu. Dodatkowo wprowadza ona czytelnika w zasady działania PET i rekonstrukcji obrazu, oraz służy jako krótki przewodnik po specyficznej dla informatyki problematyce zrównoleglania, wektoryzacji i architektur akceleratorów obliczeniowych.

Opis ten zaczyna się od stworzenia rozwiązań dla dwóch specyficznych dwuwymiarowych podproblemów – dwuwymiarowej beczki z symulowaną macierzą systemową oraz dwuwymiarowego dwupaskowego tomografu z przybliżonym analitycznie jądrem. Te dwie metody później łączą się w kompletne rozwiązanie dla trójwymiarowego tomografu J-PET. Obie metody posiadają generyczną i zoptymalizowaną dla akceleratorów GPGPU (z ang. general-purpose computing on graphics processing units) implementację. Każda z tych

implementacji została przetestowana pod względem szybkości oraz wydajności i jakości z użyciem podzbioru miar standardu NEMA (z ang. National Electrical Manufacturers Association).

Prezentowany materiał jest wynikiem trzech lat badań, dyskusji i programowania prowadzonego przeze mnie na Uniwersytecie Jagiellońskim w projekcie J-PET. Narzędzia do symulacji i rekonstrukcji J-PET są namacalnym rezultatem mojej pracy. Te narzędzia pozwoliły nam zrozumieć właściwości unikalnej geometrii tomografu J-PET oraz podjąć istotne decyzje w czasie trwania projektu. Dzięki zastosowaniu obliczeń równoległych na urządzeniach GPU, nasz tomograf jest w stanie dostarczyć wyniki prawie natychmiastowo.

Tym niemniej moja praca reprezentuje tylko małą, aczkolwiek istotną część całego procesu potrzebnego do stworzenia działającego urządzenia J-PET – rekonstrukcję obrazu korzystającą z informacji dostarczanych przez inne moduły tomografu. Stworzenie tego unikalnego urządzenia nie byłoby możliwe bez niezwykłej współpracy całego zespołu J-PET oraz bez zastosowania nowoczesnych technologii z wielu dziedzin nauki.

"To my father"

Acknowledgements

First of all I want to express my gratitude towards my supervisor prof. Piotr Białas, who shared with me his knowledge and passion to the subject of parallel computing. Without his guidance and patience for my constant source code refactoring this work would never reach its final shape.

I am greatly indebted to prof. Paweł Moskal for offering me this exceptional opportunity of joining J-PET project team and to the rest of my teammates for their enthusiasm and friendliness.

I wish also to express my thanks to prof. Ewa Grabska for inviting me to the PhD studies and Jagiellonian University for providing great work environment and top computing workstations.

I want to recognize the contribution of my colleague Jakub Kowal, who unfortunately quit our group due to personal reasons. Some ideas enclosed in this work would not exists without the fruitful discussions we had together.

Finishing this work would not be possible without support and patience of my family, especially my wife Ola, who encouraged me not to give up and keep working. And foremost, I would like to thank and dedicate this work to my late father for recognizing my interests in mathematics and computer science in my early life that made me who I am today.

Symbols and abbreviations

$\mathbf{e}$	emission (event)
$\mathbf{E}$	unobserved directly data (set of events)
$\widetilde{\mathbf{e}}$	response to event (measurement)
$\widetilde{\mathbf{E}}$	scan (set of responses)
$\widetilde{T}$	measured time
$\widetilde{z}_u$	measured hit position along upper detector scintillator
$\widetilde{z}_d$	measured hit position along lower detector scintillator
$\Delta\widetilde{l}$	distance difference between emission point and hit points
R	distance between scintillators
t	tube of response (TOR)
u	detector index, e.g. upper detector in TOR
d	other detector index, e.g. lower detector in TOR
$\mathcal{D}$	all detectors set
$\mathcal{T}$	all tubes of response set
i	2D pixel
j	other 2D pixel, e.g. inner loop pixel
$\mathcal{I}$	2D image pixel space
v	3D voxel
w	other 3D voxel, e.g. inner loop voxel
$\mathcal{V}$	3D image voxel space
ρ	emission density map image
p	point, e.g. origin of gamma quanta emission

θ	angle of emission direction
P	probability or probability density
s	sensitivity
E	mean (expected value)
$\mathcal{L}$	likelihood
ℓ	log-likelihood
PET	positron emission tomography
J-PET	Jagiellonian PET
TOR	tube of response
TOF	time of flight
ML	maximum likelihood
EM	expectation maximization
LM	list mode
NRMSE	normalized root mean square error
CPU	central processing unit
GPU	graphics processing unit
GPGPU	general-purpose computing on graphics processing units
SIMD	single instruction multiple data
MIMD	multiple instructions multiple data
SIMT	single instruction multiple threads
FLOP	floating point operation
FLOPS	floating point operations per second

Contents

Chapter 1.

Introduction to PET tomography and J-PET scanner

Advancement in the medical sciences is an important factor in improving life expectancy and quality. Being able to understand our bodies, prevent and treat common diseases is an ultimate goal of many sciences and scientists. In order to reach this goal we need to constantly develop better instruments that provide means of observation of biological processes in living subjects – necessary to gain further understanding of the phenomena driving these processes. Starting on invention of a microscope, finding of X-rays and using them for imaging, molecular imaging and position emission tomography (PET) is a next step in this advancement.

PET devices are nowadays available in many medical facilities and used for detecting cancer, brain or heart tumors. PET imaging belongs to a wider group of molecular imaging techniques and it is one if its *modalities*. Molecular imaging itself was derived from radio-pharmacology in order to better understand living body processes at cellular level. It can be perceived as fusion of molecular biology with *in vivo* imaging. Because living body molecules do not emit signals intrinsically, different techniques and methods have to be applied in order to make them emit signals that can be registered by medical instruments.

We can distinguish four major modalities of molecular imaging, depending on how signal is made to be emitted by the molecules:

- *Magnetic resonance imaging* (MRI)

  Strong magnetic field (B0) is used to align the spin of the body's atoms. Then short radio frequency pulse (B1) is emitted towards the body, as a consequence

some of the atoms loose their alignment emitting back the radio signals registered by MRI device and processed into slice images.

In general MRI measures water or fluid characteristics of the tissue and is used most often for imaging anatomical structures, providing better results than computed tomography (CT) especially in terms of image sensitivity. This is visible in comparison to CT brain imaging, where skull bone absorbs most of the X-rays, making difficult to produce detailed image of the enclosed brain.

MRI-specific probes have been developed, so MRI can also be used for imaging molecular processes, but this is still limited comparing to other modalities, especially PET.

- *Optical imaging*

  Uses the light as a source of a signal. The light is a product of either an absorption, reflectance, fluorescence or bioluminescence. The infra-red light absorption can be used for example to "observe" brain neural activity, since the absorption depends on the chemicals present in the active areas. The main problem of this method is that the light does not penetrate the tissue well enough, so it is both hard for the light to escape in order to reach the instrument detecting signal and even reach the observed tissue in order to reflect or to provoke fluorescence.

- *Positron emission tomography* (PET)

  Radionuclide tracer injected into the body emits positrons, that annihilate with the body's matter electrons. Pair of gamma quanta is emitted in the place of each annihilation. These gamma rays, traveling in almost parallel opposite direction, interact with scintillators that emit visible light transformed next by photomultipliers into electric signals registered by electronics.

  The major drawback of PET comparing to other modalities, especially MRI, is its low spatial resolution. This is usually a consequence of a small number of events being registered by PET scanners, mostly due to the limits of radioisotope tracers activity considered to be safe and limited time of the examination.

  Image reconstruction methods and algorithms can be also considered as a limiting factor. While it is possible to employ some extra information gathered during the measurement process, such as time of flight (TOF) [1] or even do precise signal

shape matching [2] in order to reduce time errors, this implies substantially more computations to be done.

- *Single photon emission computed tomography* (SPECT)

  SPECT devices detect gamma emission similarly to PET, however SPECT radiotracers injected to the body are direct gamma radiation emitters. Moreover, instead of detecting coincidences, SPECT devices use a rotating gamma camera with a collimator to detect single gamma quanta coming from one direction and build a set of 2D projection images. Like in computed tomography (CT), these projections are used to produce the final 3D image. This makes SPECT operation principle different from PET and closer to CT.

At the time being *computed tomography* (CT) is not considered as a molecular imaging, since the CT method *as-is* does not produce molecular activity image and so far no molecular activity agents were developed for CT.

1.1. PET principles and conventional PET scanners

The positron emission tomography (PET) works by estimating the radioactive fluid density (tracer) from the measurements of the γ quanta emitted from an annihilation of the positron produced by the beta plus (β^+) decay. The two quanta are emitted simultaneously and almost back-to-back. This emission is called an *event*. γ quanta travel through the detectors surrounding the measured subject, usually a patient. Some of these quanta interact with scintillators belonging to the individual detectors – this is called a *hit*.

Upon interaction, scintillators emit visible light that later reaches photo-multipliers (PMT) attached to the scintillators. Each PMT transforms arriving photons into an electric charge that is transferred as an electric signal to the attached electronics. Appearance of some signal on PMTs belonging to a single detector pair within given time period is called a *coincidence* and this time period is called a *time slot*. The detector pair yields a *tube of response* (TOR) – a subspace spanned by these two detectors passing through the emission point.

The coincidence electric signal is processed by electronics and their readout for this coincidence is called a *response*. During single examination, responses are collected

into a complete *scan* data. Conventional PET scan consists of a number of coincidences detected on each TOR. Modern PET scanners may use *time of flight* (TOF) information to improve resolution. This information carries relative time information about arrival of the signal to each PMT, that can be used to determine position of the emission within the detector. Currently all PET scanners perform the measurements using the non-organic scintillating crystals and the spatial resolution is determined by the crystal size which can be as small as few millimeters across.

PET technique has its roots in early 1950s, when Gordon Brownell and Charles Burnham of Massachusetts General Hospital conducted first demonstration of annihilation radiation for medical imaging use. This was an inspiration for the concept of emission tomography used to visualize functional processes in the body – presented by David E. Kuhl, Luke Chapman and Roy Edwards in the late 1950s. This concept was realized as the first single-plane PET scanner called *head-shrinker*.

The first 3D PET scanner called *PC-I* was completed in 1969 as a two 2-dimensional arrays of scintillators [3]. The cylindrical array of detectors soon replaced 2D arrays [4] and it is used until today.

1.2. Radiopharmaceutical tracers

Together with developing first devices, there was a need to create radiopharmaceutical tracers for PET that could be administered safely to human subjects. The first one was labeled 2-fluorodeoxy-D-glucose (2FDG) and it was administered in 1979 at the University of Pennsylvania [5]. At the time non-PET device was used to show concentration of the tracer. However, later this substance was used in development of conventional PET scanners.

2FDG is a glucose analog, that uses modified glucose molecule having $-OH$ hydroxyl group replaced by radioactive fluorine ^{18}F. Fluorine is produced in a cyclotron and has 109.8 minute half-life, that makes it effective for administration within a single day. As a glucose analog 2FDG is absorbed by high glucose consuming cells such as brain, kidney and what is most important cancer cells presenting abnormally high glucose consumption, they appear as brightest spots on PET image. Therefore 2FDG distribution in the body represents very well glucose consumption by the cells and an overall cellular activity.

1.3. Physics of PET

When radio-emitter probe decays in β^+ process, it emits a positron (anti-electron) which annihilates with surrounding matter electron after traveling short distance – up to few millimeters long. This distance depends on the type of the tissue and can vary producing different measurement errors as described in [6]. Two gamma quanta (γ) with the energy of $511keV$ are emitted almost back-to-back as a result of this annihilation. This high energy lets the gamma photons easily escape the surrounding matter of the measured subject. However the same reason makes them quite hard to interact with plastic scintillators used in J-PET, that are less dense than crystal ones, and e.g. for 2 cm thick plastic scintillator the detection efficiency amounts to about only 18%.

It is also possible that gamma quanta scatter with either subject matter or scintillator matter, causing the change of direction and some energy loss. Such *scatter fraction* may represent substantial part of overall number of detected quanta and may negatively impact on a reconstruction image quality. Since the scattered quanta have lower energy than $511keV$ there are some techniques developed [7, 8] to measure and filter detector responses originating from such scattering.

Gamma quanta hitting detectors' scintillators surrounding the subject cause them to scintillate visible light photons. Conventional PET scanners use scintillation crystals, usually inorganic high-Z GSO, LSO or BGO crystals. They are cut into the form of small cubes and glued together to form arrays or matrices.

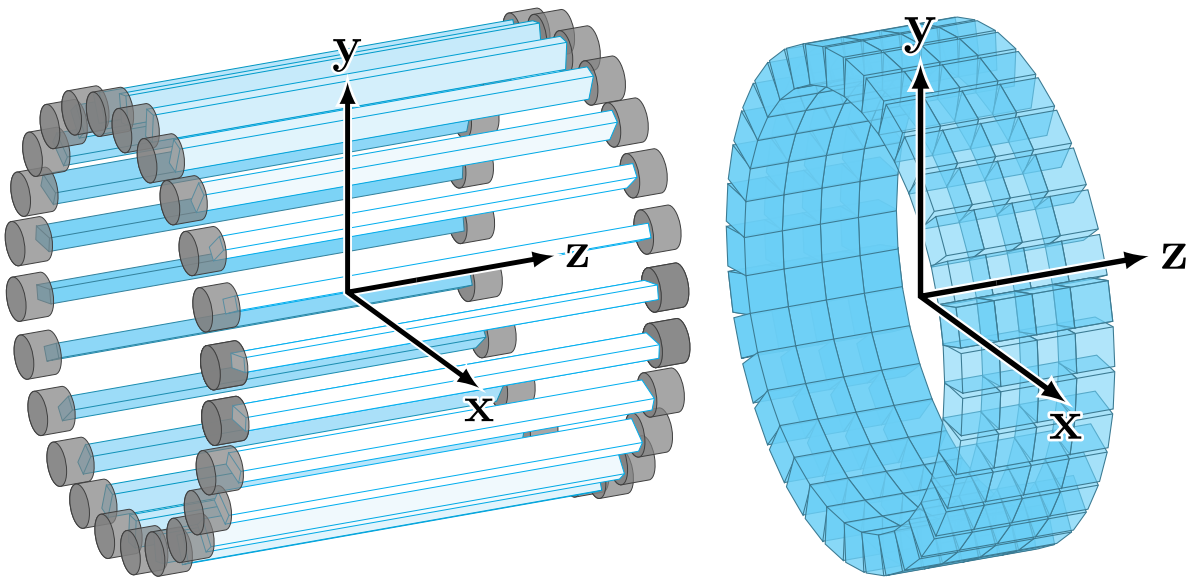
Usually few thousand photons are emitted isotropically for $511keV$ gamma quantum as a result of the scintillation. Depending on emission angle some of the light escapes and the rest undergoes internal reflection. This process is depicted in Figure 1.2 for case of J-PET scanner.

Some fraction of these photons reach photodetector, usually photo-multiplier tube (PMT), attached to one or more faces of the scintillator. The light photons are converted then to electrons in photo-electric process, which are next amplified and converted into more electrons in an electric field of PMT. Finally, the energy of each photon arriving to PMT is observable as a charge on the PMT connectors.

From this point the analog boards connected to PMTs analyze these electric signals using discriminators to detect time of arrival of the photons. The rise time of signals is

equal to about one nanosecond (10^{-9} s), and in case of J-PET scanner multi-threshold discriminators are set to detect the arrival time with the precision of tens of picoseconds (10^{-12} s). Such pre-processed signal is then supplied to the electronics and finally arrives in a form of a binary data to the image reconstruction.

1.4. Novel plastic scintillator J-PET scanner



(a) J-PET strip scanner **(b)** Classic crystal PET scanner

Figure 1.1.: PET scanners comparison

J-PET scanner is a novel tomography device using plastic polymer in place of conventional crystal scintillators. The concept of TOF-PET detector based on organic scintillators appeared in 2011 [9–12], showing that organic scintillators are more suitable for TOF than crystal ones, while at the same time being much cheaper and easier to manufacture. Plastic scintillators can also receive much larger form than scintillation crystals, that could be used to enlarge field of view of the device without affecting its complexity [13].

Idea behind J-PET operation differs from conventional tomography devices. J-PET is meant not only to replace the expensive scintillator crystals, but also to reduce overall device construction complexity. Multiple conventional scintillators along z axis are replaced with a single scintillator strip in J-PET. This significantly reduces the number

of PMTs, cables and electronic boards in the device that has an inevitable impact on production costs. Using the single long scintillator strip requires employment of advanced time of flight (TOF) techniques from the very beginning, as a position of an emission along z axis (Figure 1.1) can only be reconstructed from the precise TOF information.

1.4.1. Advantages of plastic scintillators over crystal counterparts

While lower production costs is the most important advantage of plastic scintillators, there are couple of others worth to mention:

1. Polymer plastic has better light transfer properties than crystals

   Polymers absorb internally much less light emitted by scintillation from gamma radiation. As a result enough light reaches PMTs in order to produce precise readouts when using long polymer scintillator strips. Long crystal scintillators would simply absorb all of the light before it had a chance to reach PMTs.

2. Reduced complexity of the whole device

   Because series of crystal scintillators along z axis is replaced by a single plastic scintillator strip, overall device complexity is reduced. Especially in the case of the large field of view, the device needs far less photo-multipliers that turn photons into electric signals and far less cables that carry these signals to electronic circuit boards. Finally, the device response space is reduced too, as number of detector pairs where signal coincidence can appear in is substantially smaller.

3. Complexity invariance to the length of the scintillator and the length of the scanner along z axis

   Changing the length of the scintillator and effectively the length of the whole device along z axis has no impact on the complexity of the J-PET scanner. Unlike J-PET, conventional scanners need extra scintillators, photomultipliers and cables in order to enlarge their field of view (FOV) along z axis.

   However increasing the length of scintillators in the J-PET scanner has an impact on spatial resolution along z axis [14]. This is because longer scintillator produces less distinct readouts on attached photomultipliers, affecting time measurement precision. Therefore one needs to find a balance between the resolution and the

field of view, when defining specific device geometry configuration, e.g. when resolution along z is important, shorter scintillator strips can be used, when resolution along z is less important, very long strips can be used to create the whole body PET scanner.

4. Ability to operate together with computed tomography (CT) solutions

   Plastic scintillators absorb CT radiation much less than crystals, therefore J-PET scanner can be coupled with CT device, operating in the area not covered by PMTs and cables, in order to provide perfectly aligned PET and CT imagery from single medical examination [15].

1.4.2. J-PET scanner principles

J-PET scanner operational principles are similar to classic scanners, except that the exact time information plays crucial role for J-PET. Classic crystal matrix scanners can optionally use TOF to improve the quality and the resolution of a reconstructed image, but they can operate as well in less computationally demanding bin-mode, that relies only on counting signals on TORs. J-PET however requires TOF information to perform an image reconstruction along z axis and to produce full 3D image.

In general J-PET device scan $\widetilde{\mathbf{E}}$ records information about signal coincidences represented by responses $\widetilde{\mathbf{e}}_k$ belonging to the response space $\mathcal{R}$

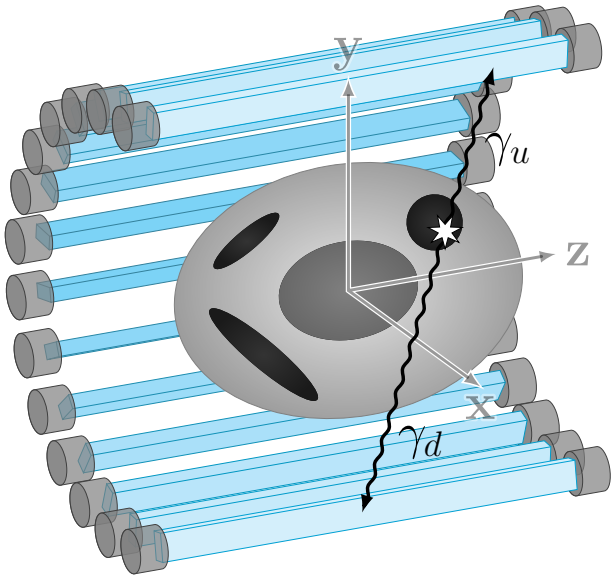
$$\widetilde{\mathbf{E}} = \{\widetilde{\mathbf{e}}_1, \widetilde{\mathbf{e}}_2, \ldots, \widetilde{\mathbf{e}}_N\} \subset \mathcal{R} \tag{1.1}$$

Responses are denoted with tilde to emphasize that they represent measured information that is a subject to measurement errors explained in detail in Chapter 3.

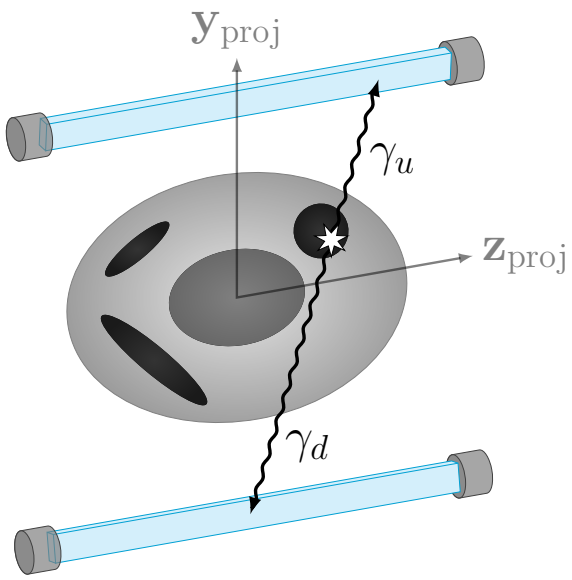
Each response contains an information about scintillator pair indices (TOR) denoted later in this work with $t = (u, d)$, where u stands for "*up*" and d stands for "*down*", and information about photons' time of arrival on each of the photomultipliers attached to the scintillators denoted with $\widetilde{T}_{ul}, \widetilde{T}_{ur}, \widetilde{T}_{dl}, \widetilde{T}_{dr}$

$$\widetilde{\mathbf{e}} = (u, d, \widetilde{T}_{ul}, \widetilde{T}_{ur}, \widetilde{T}_{dl}, \widetilde{T}_{dr}) \tag{1.2}$$

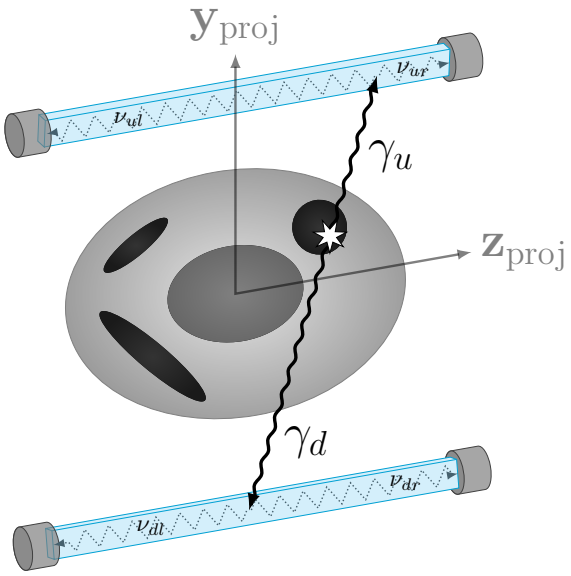
where indices l and r indicate left and right side of the scintillator strip, respectively as shown in Figure 1.2. Time to position conversion is explained in detail in Chapter 6.



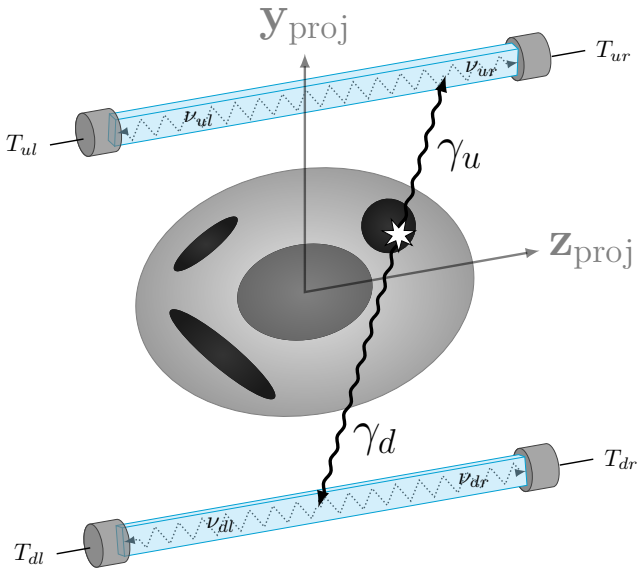
(a) Emission of gamma quanta pair



(b) Hit coincidence on a pair of scintillators



(c) Emission of photons in the scintillators



(d) Detection of photons and registration of time of their arrival

Figure 1.2.: J-PET scanner principles

1.4.3. Signal and time of flight (TOF) sampling

Aside of its unique geometry and scintillators J-PET detector consist of specialized electronics being able to measure time with great precision. Exact description of these electronics are beyond the scope of this work. The interested reader is referred to [16]

and the PhD thesis of Dr. Korcyl [17] for the details. J-PET project also tries to employ additional methods to improve the TOF and position resolution based on the shape of registered signals [2].

Current prototype electronics are able to measure signals with a precision of about 30 ps (picoseconds, $30 * 10^{-12}$ s) [18] combining analog discriminator circuits with fast FPGA electronics. Yet the time spread due to the light transport along scintillators and small photon statistics results, in case of the present J-PET prototype, in the TOF resolution of about 125 ps (σ) [19] corresponding to 1.6 cm along the scintillator and 2.7 cm between the scintillators.

In addition, it was shown recently in [14] that the physical limitation for the TOF resolution with the J-PET method amounts to 50 ps for the 50 cm long axial field of view. Still 50 ps represents about 1 cm of distance between the scintillators, so it is impossible to produce high resolution image without employing statistical methods, e.g. maximum likelihood expectation maximization (MLEM) described in Chapter 3.

Later in this work we will assume TOF resolution of 1 cm along and 2 cm between the scintillators, which is a small but expected improvement to the current prototype electronics' resolution, as such accuracy was already obtained for single strip measurements, as shown in [14].

1.5. Overview of this work

This work describes a statistical model of J-PET scanner and a set of algorithms and software tools used to create J-PET simulation and reconstruction tools being a functional part of the complete J-PET scanner device. Many of the presented algorithms employ existing well known methods, such as MLEM. Such methods however provide only generic templates for a specific application. In order to use them, J-PET scanner statistical model was developed and validated using Monte-Carlo simulations. Finally, the whole implementation was optimized in terms of the performance for modern massively parallel architectures.

This dissertation is organized into eight chapters. The first three chapters serve as an introduction to the subject of Positron Emission Tomography, J-PET device and parallel computing. Chapter 2 introduces the reader to the parallelization and vectorization which is key aspect of this work and single-instruction multiple-threads

(SIMT) programming paradigm specific to NVIDIA CUDA GPGPU environment. Chapter 3 describes PET image reconstruction theory and principles necessary to understand mathematical methods used in J-PET image reconstruction.

Description of my research begins in Chapter 4, treating about a simulation of 2D barrel detector geometry. Conducting simulations of the barrel geometry was the first and important research phase that was necessary to constitute understanding and the geometrical and mathematical model, and effectively make important decisions about geometry of the first prototypes. Chapter 5 is a continuation that describes methods of 2D image reconstruction for the 2D barrel detector.

Chapter 6 represents an original approach for image reconstruction based solely on time of flight (TOF) information in 2D strip detector. The 2D strip detector being subject of this chapter has specific geometry unlike existing detectors or 2D barrel. In order to be able to perform image reconstruction for 2D strip detector we had to create a specific statistical model that describes probability of detecting emission within the two scintillator strip detector's field of view, including time measurement errors that have serious impact on the detector's response.

Chapter 7 is a completion of my research combining models and developed algorithms for 2D subproblems into a full solution for 3D J-PET scanner image reconstruction. Included performance and image quality statistics expressed according to the industry standards can be used to compare J-PET with other PET tomography devices.

Final Chapter 8 contains the conclusions and opens the discussion about the future directions for development of the project and its image reconstruction methods. Currently J-PET project is in a prototype phase and there is still a long road ahead to make the J-PET device ready for commercial production.

Since the source code for the programs forming J-PET simulation and image reconstruction tools is an inherent element of my research, this dissertation is supplied with appendices containing description how to obtain, build and run the programs and also how to visualize and analyze the output data produced by the programs.

1.6. Current state-of-the-art of PET image reconstruction

PET tomography and image reconstruction methods were constantly developed during over 40 years. Starting from the invention of PET radiopharmaceuticals in 1970s [5], followed by setting the foundations for PET image reconstruction statistical methods in 1980s [20, 21] and list-mode reconstruction proposed in 1990s [1].

The more advanced methods and the more precise models needed more advanced computing techniques. The last 10 years have brought many applications of new computing platforms for PET tomography, beginning with an employment of computational clusters [22] and SIMD instructions [23]. Today, the usage of GPU and computing accelerators is de-facto standard in modern PET scanner devices [24–32].

Nowadays, many of PET tomographs take advantage of time of flight information to improve the resulting image quality and resolution [32–39]. However, since most of the described methods apply to conventional crystal scintillator PET devices, they can be only partially applicable to J-PET specific scintillators and geometry. Therefore, it was necessary to develop dedicated statistical model and algorithms for J-PET tomograph, which is the subject of this work, presented in the next chapters.

Chapter 2.

Parallelization, vectorization and computational accelerators

Producing computer hardware capable of performing enormous amount of computations is no longer an issue today. Today's mobile phones hold the power of supercomputers from the decade ago, modern desktop computers contain components capable of performing breathtaking 10^{12} operations per second. This power has to be yet tamed and mastered, and it is not easily available. The *peak performance* which is used to describe and advertise theoretical computing power of a specific device is not necessarily reachable for real use. What is worse, there are couple of new problems that did not exist while ago – the costs of electricity running the hardware and the problems of heat dissipation.

While this work do not address these two problems, it tries to help to solve the other – taming the power of the computing devices built using new parallel and vector architectures.

2.1. Modern computational accelerators

Throughout this work we will consider *graphics processing units* (GPU) and *general-purpose computing on graphics processing units* (GPGPU) devices to be equivalent of modern computational accelerators. This is simply because modern computational accelerators are either GPU devices such as NVIDIA *GeForce* series, AMD *Radeon*, or were derived from GPU architecture, such as Intel *Xeon Phi* that was originally set to be GPGPU device codenamed *Larrabee*. This is visible on TOP 500 supercomputer

list [40], where strongest two supercomputers at the time of this writing are hybrids using *Xeon Phi* and *NVIDIA* GPU solutions.

The GPU origin of the modern computational accelerators is very convenient in the context of PET image reconstruction, which is a subject of this work. Especially when it comes to the spatially coalescent memory access that can be provided by GPU hardware texture units, or geometry manipulation using trigonometric functions that are hardware optimized and inherent to GPU.

2.2. *Moore*'s law myths and facts

Almost everyone who had a contact with computer science had heard about *Moore*'s law. However, just a few remember its message in the original form, that the number of transistors in a dense integrated circuit doubles approximately every two years. Most of us rather think of it, as the performance of computers doubling in every two years. Moreover, many think that existing software will run twice as fast when buying new machine every two years. This has been true for almost twenty years, but recently this is not working anymore. Here is why.

We need to get back to the original *Moore*'s message. It is all about the number of transistors in silicon. The law still works, but doubling number of transistors does not have immediate impact on existing software solutions.

Until the end of twentieth century, the computer chips were improving their performance mostly by increasing their clock frequencies and reducing fabrication process down to tens of nanometers. However, due to subtle nanoscale effects we can no longer raise frequencies without emitting large waste of heat.

Therefore today the raising number of transistors is reflected in new processor cores, wider vectors and new computing units, while the frequency is capped to around 1–4 Ghz.

2.3. $\mathcal{O}(n)$ in context of computational accelerators

Understanding algorithms and their complexity is a foundation of computer science. However these algorithms do not run in the vacuum. They run on real hardware,

having its limitation coming straight from number of transistors and the architecture. Choosing best algorithm in terms of computational and memory complexity is still important, but has to be done in the context of the real hardware.

Lower complexity algorithm is not necessarily better than higher complexity algorithm in all of the cases. That depends, among others, on the data size. We can imagine some higher complexity algorithm that performs much better for a data that fits into available computer memory. The point where lower complexity algorithm's curve takes over the higher complexity's one may be far beyond available device memory as shown in Figure 2.1.

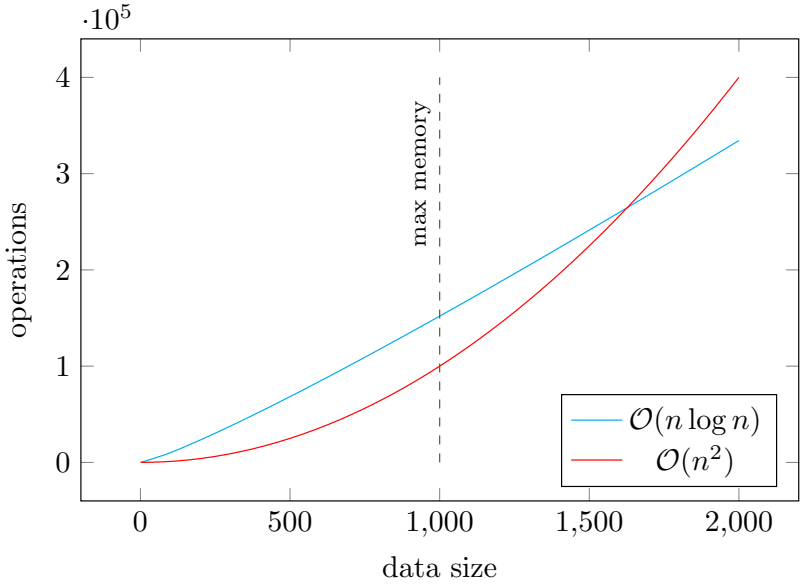


Figure 2.1.: Algorithm complexity in context of available memory

Lets take an example of well known and universally used *quicksort* classic algorithm in context of parallel computers. It turns out that it does not perform best on parallel hardware, but some less known algorithms, such as merge sort, perform better. In case of *quicksort* its "divide and conquer" strategy requiring recursive execution stack handling, affecting memory complexity, is a weakness on parallel machines. Memory access is a limiting factor on parallel architectures, providing much higher compute throughput than memory bandwidth. Therefore parts of algorithm that require memory access are often a performance bottleneck.

Aside the memory complexity, some algorithms are just inherently susceptible to parallelization, whether others are inherently resistant. Usually such resistance comes from the sequential nature of some algorithms. This leads us to Amdahl's law,

that states that inherent sequential part *seq* of the every parallel algorithm limits the speedup S of that algorithm relative to N number of processors

$$S(N, seq) = \frac{N}{seq * N + (1 - seq)} \tag{2.1}$$

When we look on the number of processors of modern GPUs, that often reaches thousands, Amdahl's law is even more visible. Therefore when choosing or creating an algorithm we must ensure that its sequential part is minimal as possible, otherwise it will not scale well on massively parallel modern devices.

2.4. Compute-bound and memory-bound problems

Modern processors and computing accelerators are usually described by two values – a peak performance expressed as *elementary* floating point operations per second (FLOPS) and a memory bandwidth, expressed as a number of bytes that can be transferred from or to the memory. We need to understand that these are inviolable hardware limits.

The elementary arithmetic operations such as an addition, subtraction or multiplication can be usually executed within one cycle of the processor, therefore the peak FLOPS can be expressed as a product of the processor clock frequency and the number of available arithmetic-logic units (ALU). There are many other operations, such a transcendental functions, that have their own hardware instructions, but are not elementary and cannot be equated with FLOP. On the other hand, many modern processors offer fused multiply-add (FMA) that can multiply two operands then add third one within one cycle. In most of the cases peak FLOPS is expressed with FMA giving factor of 2 boost. Unfortunately not all algorithms can fully or even partially utilize FMA, therefore half of the peak performance is often a limit.

Roofline model [41] describes and explains the relation between the peak performance and the memory bandwidth limits. This model relies on definition of parameter Q that represents a ratio of a number of elementary arithmetic operations to a number of elements needed to be transferred from or to the memory. This parameter is expressed either in $FLOP/element$ or $FLOP/byte$ and is specific to the algorithm. The algorithm performance P_{max} is limited by both peak performance P_{peak} and a product

of Q parameter and memory bandwidth B_{mem} of the device or computing architecture

$$P_{\text{max}}(Q) = \min\{P_{\text{peak}}, Q * B_{\text{mem}}\} \tag{2.2}$$

We can find Q_{dev} constant for which $P_{\text{peak}} = Q_{\text{dev}} * B_{\text{mem}}$, specific to the device and representing ratio where memory bandwidth is balanced by the peak computing performance. The higher Q_{dev} means that every algorithm must perform more operations for a single element loaded from the memory in order to take advantage of the processor's computing power.

The model can be also visualized as an intersection of two half planes representing computing performance and memory bandwidth limits as shown in Figure 2.2. Using this model we can distinguish *memory bound* algorithms with $Q \leq Q_{\text{dev}}$ – lying in the left blue region, loading too much data and *compute bound* algorithms with $Q \geq Q_{\text{dev}}$ – lying in the right red region, performing more operations that is necessary to balance memory bandwidth.

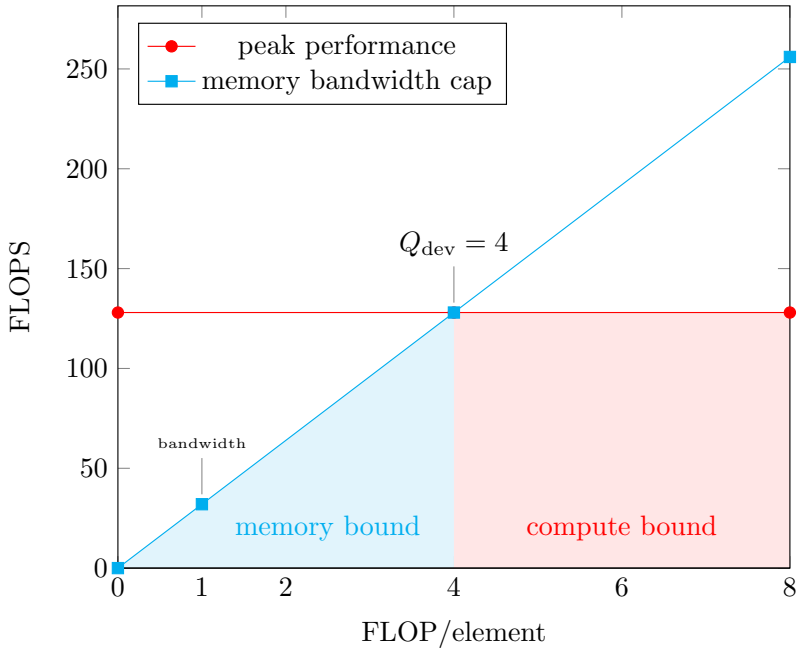


Figure 2.2.: Roofline model with $Q_{\text{dev}} = 4$

As shown in Table 2.1, processors with a high peak performance have usually large Q_{dev} factor. This means that the strongest processors and accelerators need substantially more operations (FLOP) to be performed for a single loaded memory element than the weaker models. This implies another important observation that

algorithms that are compute bound on weaker units may be memory bound on the stronger ones. This means that the whole optimization process has to be done always in the context of a particular device or device architecture.

Device	P_{peak} (SP)	B_{mem}	Q_{dev}
Intel i5-4258U[1]	76 GFLOPS	26 GB/s	12 FLOP/element
Intel Core i7-4770K[2]	224 GFLOPS	26 GB/s	34 FLOP/element
Intel Xeon E5-1650v3[2]	307 GFLOPS	51 GB/s	24 FLOP/element
Intel Xeon E5-2699v3[3]	662 GFLOPS	68 GB/s	39 FLOP/element
Intel Xeon Phi 3120A[2]	2006 GFLOPS	240 GB/s	33 FLOP/element
GeForce GTX 980 Ti[2]	5632 GFLOPS	336 GB/s	67 FLOP/element

[1]Mobile processor of my MacBook Pro
[2]Processors and accelerators available in our workstations
[3]Strongest CPU available currently from Intel

Table 2.1.: Theoretical peak performance (single precision), memory bandwidth and Q_{dev} parameter for few example modern processors

There exists another memory related limiting factor, not taken into account in the Roofline-model – *memory access latency*, describing the delay, expressed usually as a number of cycles, between issuing the memory access instruction and executing it. In other words, saying how long we need to wait for the data transfer to start, rather than how quick data is transferred – described by memory bandwidth.

Memory access instructions are usually pipelined, therefore the latency can be hidden, e.g. using several hardware threads utilizing a single memory unit. However in some specific cases device schedulers are unable to hide latency, that appears as a degraded performance, worse than the implied by the available memory bandwidth.

2.4.1. Estimation of efficiency using *Roofline-model*

Roofline model can be used to estimate efficiency of the existing algorithm and make the survey of possible room for improvement. In order to do that we need to

1. Estimate number of operations performed by the algorithm, e.g. in the single cycle of the main algorithm's loop – this determines y position on the roofline model Figure 2.2.

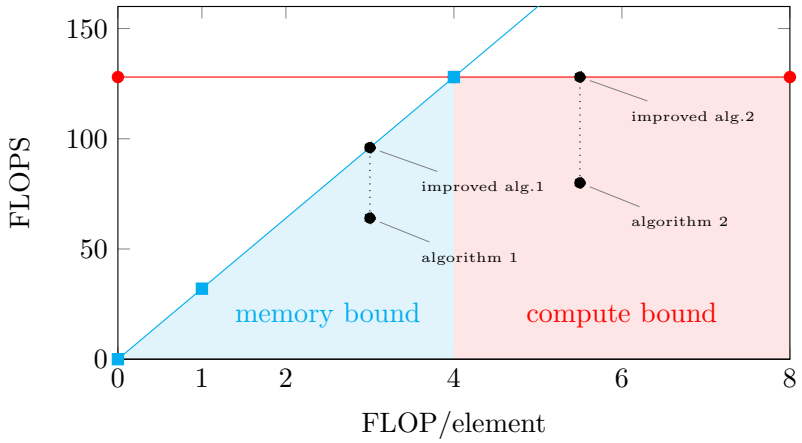


Figure 2.3.: Example estimation using roofline model

2. Estimate number of elements (e.g. floating point numbers) loaded and stored from and to memory within single cycle of the main algorithm's loop.

3. Calculate Q_{alg} ratio of the number of operations (point 1) to the number of elements (point 2). This gives us x position and the whole (x, y) algorithm position on the roofline model Figure 2.2.

4. Find the point (x, y') lying on the memory bandwidth or the peak performance limit boundary above (x, y).

The $y' - y$ difference represents possible improvement of the algorithm. The usual reason that the original algorithm lies under the boundary is incomplete paralleliza-tion e.g. due to branch divergence [42], incomplete use of the available pipelining, unhidden memory latency or in-coalescent memory access.

2.5. Parallel computing models

Parallel computing is generally described as a type of computing involving multiple operations at once. There exist many different parallel computing models realizing this definition and many classifications of them. One significant of these is Flynn's taxonomy [43] – a classification of algorithms and computer architectures depending on how the computer hardware instructions are executed and what are their operands.

SISD *single instruction, single data* – equivalent of sequential processing, where single processor is executing single instruction operating on single value or memory address.

MISD *multiple instruction, single data* – a processing where many processors or units operate on single value or memory address. Such model is not very popular and may be used to realize fault-tolerance.

MIMD *multiple instruction, multiple data* – a fully parallel processing where many independent processors or units operate on own values or memory addresses.

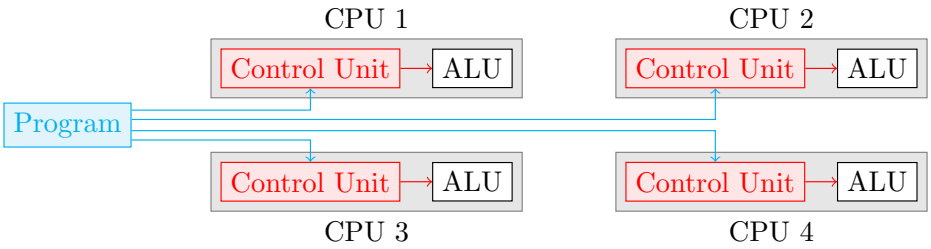


Figure 2.4.: MIMD processing

SIMD *multiple instruction, multiple data* – also called *vector processing*, where single instruction operate on multiple values or memory addresses, forming a vector of data. This can be perceived as a constrained MIMD model, where all units must execute same instruction at a given time.

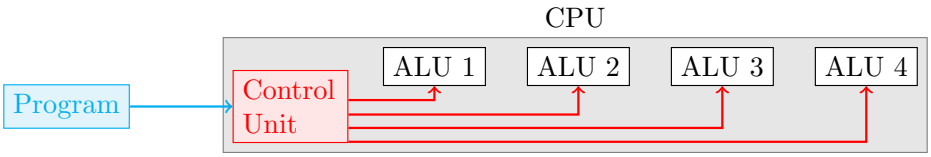


Figure 2.5.: SIMD processing

Most of the modern processors are hybrid of MIMD and SIMD, depicted in Figure 2.6. Such a hybrid architecture is a result of economic calculation – making all available arithmetic-logic units (ALU) operate in full MIMD manner would raise number of transistors and energy consumption. On the other hand, SIMD-only architecture using long vector operands, not providing any means of independent processing, is problematic for the programmers and compilers, as most of the algorithms cannot be completely vectorized for SIMD processing.

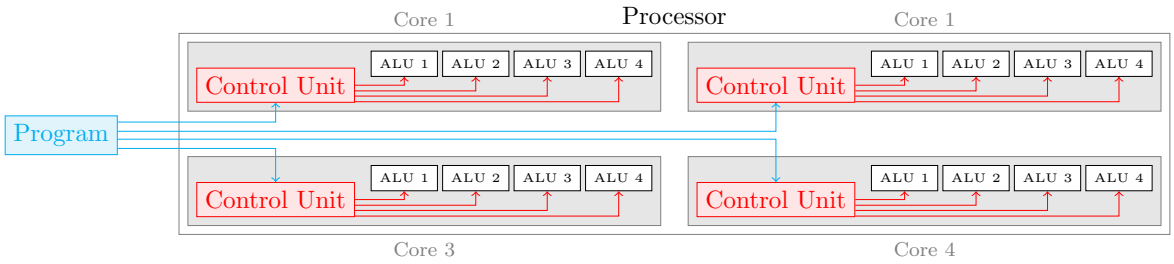


Figure 2.6.: Modern processor architecture (MIMD and SIMD hybrid)

Many modern processors often offer, in some extent, *simultaneous multithreading* (SMT) [44, 45] – a hardware thread support, e.g. Intel's *hyper-threading* (HT), as an addition to their MIMD architecture to better utilize available computing units. For example, two hardware threads running on the same core may utilize its resources better, because when one thread stalls on memory access unit (MU), the other one may use ALU.

2.5.1. SIMT CUDA processing model

Compute Unified Device Architecture (CUDA) parallel computing platform introduced in 2007 by NVIDIA offers a new *single instruction, multiple threads* (SIMT) architecture and simple scalar-like parallel programming model that unifies SIMD, SMT and MIMD. Rather than operating directly on SIMD vector registers as in Algorithm 1, the code is expressed in scalar manner and then scheduled across large number of hardware threads, each having unique thread identifier threadIdx as shown in Algorithm 2.

Algorithm 1 SIMD memory add algorithm

```
function ADDSIMD(r[], a[], b[], n)
    for i ← 1 to n step 4 do
        a⃗ ← vec4load(a, i)
        b⃗ ← vec4load(b, i)
        r⃗ ← vec4add(a⃗, b⃗)
        vec4store(r, i, r⃗)
    end for
end function
```

Each group of 32 threads with consequent threadIdx identifiers forms one *warp*, depicted in Figure 2.5, which is executed in SIMD fashion – all threads within a warp execute the same instruction at given time. In fact a warp is an equivalent of a vector

Algorithm 2 CUDA SIMT scalar-like memory add algorithm

function ADDCUDA($r[], a[], b[]$)
  $i \leftarrow \text{blockDim}.x * \text{blockIdx}.x + \text{threadIdx}.x$
  $r[i] \leftarrow a[i] + b[i]$
end function

from traditional SIMD architectures, yet it is not expressed explicitly as in Algorithm 1. Several warps form a *block* of threads, which runs on single GPU core with its warps scheduled in SMT fashion – one warp is active, other warps are waiting. When one warp stalls, e.g. on memory access, the other may be made active to use idle ALU and fill the pipeline. Finally, several blocks are run on multiple GPU cores (SMX) in MIMD fashion.

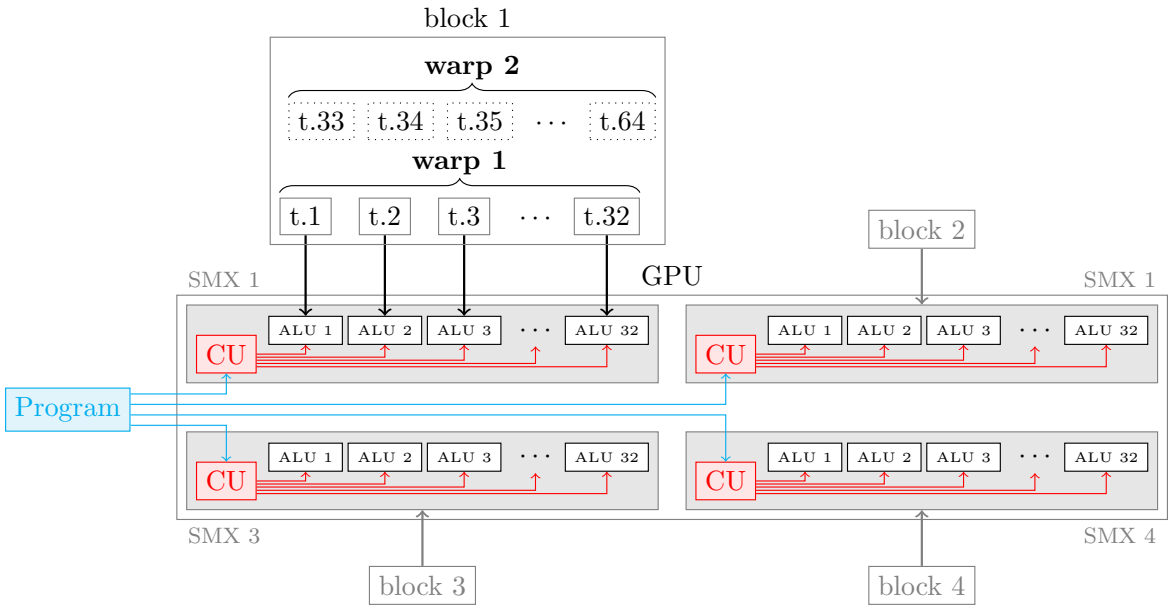


Figure 2.7.: CUDA SIMT architecture

CUDA SIMT offers hardware branch divergence support within a warp, explained by us in detail in [46], so given a predicate, some warp threads for which the predicate is true, execute instruction in the conditional block, where others not, unlike in other SIMD architectures where single SIMD instruction always operates on all vector register elements. Some recent SIMD architectures, such as Intel AVX-512, introduce explicit mask registers and an additional vector instructions mask operand which can hold predicate mask, but the branch divergence and convergence still has to be done explicitly, either automatically by the compiler or manually by the programmer.

Finally, CUDA SIMT offers a scattered memory location access support, e.g. $r[f(i)] \leftarrow a[g(i)] + b[h(i)]$, where f, g and h denote any function of thread index. The hardware optimizes such accesses, so they are done with a minimum number of hardware memory read or store operations. This is again unlike most of the conventional SIMD architectures, requiring all read or stored vector elements lie in consequent memory locations. This is partially addressed in AVX-512, offering hardware gather and scatter instructions – still however representing an explicit approach.

Altogether SIMT model is easier to program for than explicit SIMD models. Optimizing algorithms for SIMT usually means reducing a number of incoherent (scattered) memory accesses, effectively reducing a number of hardware atomic memory operations, thus improving the algorithm's memory throughput, and reducing a number of divergent conditional blocks, causing less warp hardware vector elements to be idle.

2.6. Implicit and explicit parallel programming models

It is important to leverage both parallelism and vector computing properties of the hardware, otherwise the computing solution is less cost effective as some computing units are either idle or draining power regardless of not performing any operations. Only algorithms utilizing both SIMD and MIMD computer architecture properties are able to reach the peak performance of the computer illustrated in Figure 2.8.

Along with many SIMD and MIMD architectures, there exist many specific programming models, either provided by the vendors or defined by some standards. Below the most popular ones were named, together with a distinction whether they are *explicit* – programmer must explicitly use vector instructions and special pragmas to achieve full parallelism, or *implicit* – programmer writes scalar-like sequential code that is implicitly parallelized and vectorized by the compiler or a hardware device scheduler.

We also make a distinction between parallelization and vectorization. Both are meant to leverage parallel computing, however parallelization is generally speaking – making the algorithm run on multiple cores using multiple threads, while vectorization is making the algorithm use the computer SIMD vector instructions and units. Therefore both represent same purpose, but different realization.

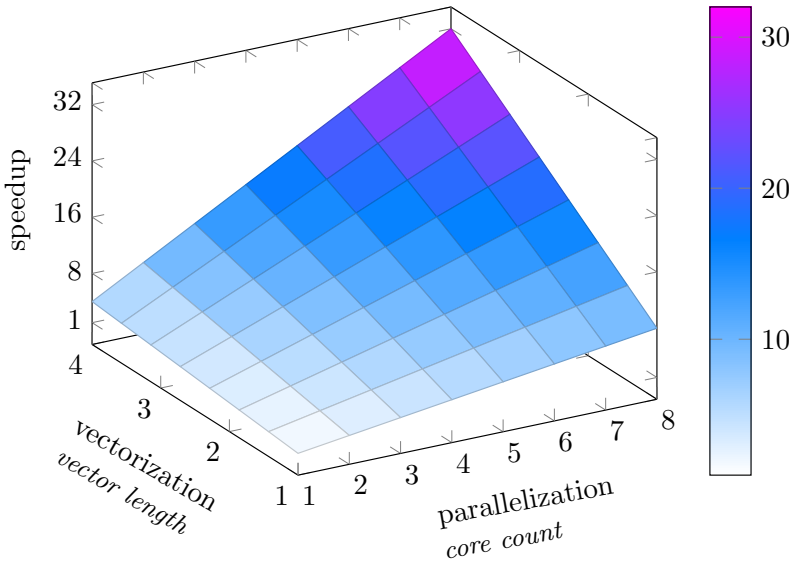


Figure 2.8.: Combined parallel and vector computing performance

2.6.1. Explicit parallelization with *OpenMP*

OpenMP standard defines a compiler extension and special compiler pragmas used to annotate source code in order to instruct the compiler to parallelize given programming language constructs such as loops or blocks of code, as shown in Algorithm 3.

Algorithm 3 OpenMP multi-threaded add algorithm

```
function AddOpenMP(r[], a[], b[], n)
    #pragma omp parallel for        ▷ schedule loop iterations across available cores
    for i ← 1 to n do
        r[i] ← a[i] + b[i]
    end for
end function
```

2.6.2. Explicit vectorization with *OpenMP*

Version 4 of the OpenMP standard introduces new SIMD pragmas to mark chunks of code that can be subject to the vectorization as shown in Algorithm 4. This looks like a great extension, but in practice these constructs are just suggestions or hints to the compiler that may be often unable to perform vectorization regardless of use of these annotations. Moreover, where possible compilers often perform automatic vectorization regardless of OpenMP SIMD pragmas. Therefore these pragmas serve their

purpose only when they carry an extra information letting the compiler relax some assumptions about data dependence, making the automatic vectorization possible.

Algorithm 4 Hybrid OpenMP multi-threaded SIMD add algorithm

```
function AddOpenMP(r[], a[], b[], n)
    #pragma omp parallel for        ▷ schedule loop iterations across available cores
    for i ← 1 to n step 4 do
        #pragma omp simd            ▷ turn inner loop into single vector instruction
        for j ← 1 to 4 do
            r[i + j] ← a[i + j] + b[i + j]
        end for
    end for
end function
```

2.6.3. Compiler automatic vectorization

While modern compilers do not offer automatic parallelization they try to leverage vectorization where possible. Tight loops containing few arithmetical operations with no conditional clauses are perfect candidates for automatic vectorization.

Mathematics behind these automatic optimizations evolved recently. Polyhedral model is best example of this evolution [47]. It describes a topology of a loop iteration space including all pre and post conditions and offers means to divide this space into subspaces than can be a subject of an independent parallel execution. Polyhedral frameworks such as *Cloog* [48] offer new optimizations to the compilers especially in combination of APIs such as OpenMP. Unfortunately these frameworks are usually not included by default in compiler distributions, mostly because they noticeably degrade compile time, while not really guarantying performance improvements.

2.6.4. Explicit vectorization with *SIMD* intrinsics

Using intrinsics is a last resort to vectorize code that cannot be vectorized otherwise, especially when your code contains conditional clauses and larger loop bodies that cannot be comprehended by the compiler with its limited look-ahead or look-behind capabilities.

Intrinsics are however architecture and device dependent and are against generic programming practices. Using intrinsics require deep understanding of the platform and hardware architectures. They also make the code obscure and often unreadable.

2.6.5. Explicit vectorization using meta-programming

Luckily there is a good and feasible alternative to intrinsics. Many vendors offer some platform independent abstractions over intrinsics. For purpose of ϕ^4 project [49] we have created our own *Vector* wrapper type that hide intrinsics into intuitive and generic C++ syntax.

This approach gave us the best results for CPU vectorization in ϕ^4 project, far better than compiler automatic vectorization or OpenMP SIMD constructs. Using *Vector* class utilizing C++ operator customization and templates let us vectorize significant parts of the code without degrading readability of the code.

Moreover, our *Vector* class offered some means of branch divergence handling based on masks – something that was previously exclusive to CUDA SIMT model.

2.6.6. CUDA's *SIMT* implicit parallelization and vectorization

As described previously, CUDA SIMT model offers unique approach to parallelization and vectorization, which uses a concise scalar-like syntax with an implicit thread handling. Parallelization and vectorization of CUDA programs is executed at hardware level, where the scheduler plays a vital role grouping threads into warps and blocks, unlike other programming models where multi-threading and vectorization must be done either at source code level and/or hardware instruction level.

2.6.7. OpenCL and *SPIR* implicit parallelization and vectorization

OpenCL is a direct response to NVIDIA's CUDA. It is an effort for a SIMT-like programming paradigm open standard. It is actively developed by AMD and Intel. Intel has recently introduced many additions to its processors' SIMD instruction sets, e.g. AVX-512, to make the implicit model realizable easier on x86 platforms. NVIDIA is also providing support OpenCL, however not for the recent versions of the standard.

2.6.8. Lessons learned from ϕ^4 Monte-Carlo simulation

Prior conducting my research in J-PET project I had an opportunity to design algorithms for Monte-Carlo simulations of the ϕ^4 lattice field theory [49]. The problem space was represented by a 2D lattice of elements that were randomly altered by some kernel. This project was a great playground for trying several explicit and implicit parallelization and vectorization techniques.

It turned out that the implicit SIMT mode, present in CUDA and OpenCL, delivered best results for massively parallel architectures, such as GPUs, while the explicit parallelization and vectorization was more laborious, yet not rendering equally good results in terms of performance. This gave us an intuition for J-PET project to use SIMT paradigm for compute intensive tasks.

For the purposes of ϕ^4 simulation we have developed an efficient random number generator (RNG) [49] derived from Tausworthe algorithm. We use this RNG also for the purposes of J-PET detector simulation presented in Chapter 4.

Chapter 3.

PET image reconstruction theory and principles

The objective of PET image reconstruction is to determine a distribution of the emitter in a given subject. In particular case the subject may be a living body and the emitter may be a radio-tracer injected prior to the examination. In such case the output image will represent a distribution of the radio-tracer in the patient's body.

So the problem can be defined as follows – given a scan

$$\widetilde{\mathbf{E}} = \{\widetilde{\mathbf{e}}_1, \widetilde{\mathbf{e}}_2, \ldots, \widetilde{\mathbf{e}}_N\} \subset \mathcal{R} \tag{3.1}$$

a set of measurement responses $\widetilde{\mathbf{e}}_k$ belonging to PET scanner response space $\mathcal{R}$, we want to find the emission density

$$\rho(p) : \mathbb{R}^3 \to \mathbb{R} = \text{emissions/second} \tag{3.2}$$

that yielded some unobserved directly emission events

$$\begin{aligned} \mathbf{E} &= \{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_M\} \subset \mathbb{R}^3 \times \mathbb{R}^3, \\ \mathbf{e}_k &= (p_k, \vec{r}_k) \quad p_k \in \mathbb{R}^3,\ \vec{r}_k \in \mathbb{R}^3 \end{aligned} \tag{3.3}$$

that were observed indirectly through their manifestation $\widetilde{\mathbf{E}}$, where p_k denotes emission point, $\vec{r}_k$ denotes emission direction vector. Some of these events may not be detected by the PET scanner at all, therefore $N \leq M$ and $\#\widetilde{\mathbf{E}} \leq \#\mathbf{E}$.

Since we want to make this problem computable, we will limit the problem to finite discrete voxel space $\mathcal{V}$ and a discrete emission density map representation

$$\begin{aligned} &\rho(v) : \mathcal{V} \to \mathbb{R}, \\ &\mathcal{V} = \{0, \ldots, \mathcal{V}_x\} \times \{0, \ldots, \mathcal{V}_y\} \times \{0, \ldots, \mathcal{V}_z\}, \\ &v = (v_x, v_y, v_z) \in \mathcal{V} \end{aligned} \tag{3.4}$$

where v is a voxel belonging to discrete 3-dimensional image space $\mathcal{V}$ of size $\mathcal{V}_x, \mathcal{V}_y, \mathcal{V}_z$.

In order to be able to determine $\rho(v)$ first we need to find a model that lets us transform emission $\mathbf{e}_k \in \mathbb{R}^3 \times \mathbb{R}^3$ into response of detector – $\tau(p_k, \vec{r}_k) = \widetilde{\mathbf{e}}_k \in \mathcal{R}$. The sought τ transform depends on a geometry, physics of the scanner and other components properties such as precision of the electronics.

Ideally, if τ was just a function $\tau : \mathbb{R}^3 \times \mathbb{R}^3 \to \mathcal{R}$, then image reconstruction would be a problem of finding τ^{-1}. Unfortunately due to several types of uncertainties and errors, listed below, defining such function unambiguously is not possible, hence τ can be perceived as a *random function*.

scintillator hit position uncertainty – each detector made of a scintillator and two photomultipliers is unable to distinguish exact place where the gamma quanta hit the scintillator along the secant made of the gamma ray. The scanner measures just the time, so the distance to the hit point, but this information is not sufficient to ascertain hit position along the secant – red line segment in Figure 3.1.

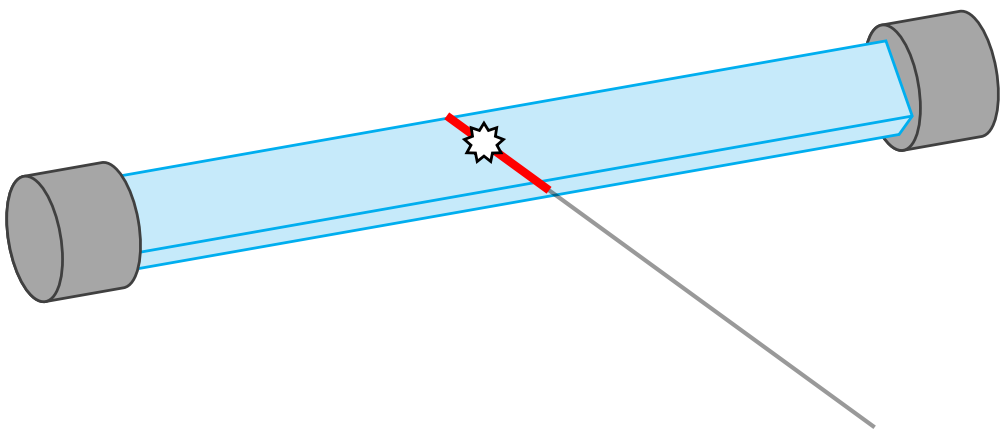


Figure 3.1.: Scintillator hit position uncertainty

scintillator interaction uncertainty – there is no guarantee that the gamma quantum traversing the scintillator will interact with it at all. This interaction is described by an attenuation law probabilistic model explained later in this work.

tube of response emission position uncertainty – even when using precise TOF information the scanner is unable to distinguish an exact place crosswise the TOR made of two scintillators. This error is specific to $x - y$ plane and it is relative to width of TOR – red line segment in Figure 3.2, which is in turn relative to dimensions of the scintillators and their arrangement in the scanner.

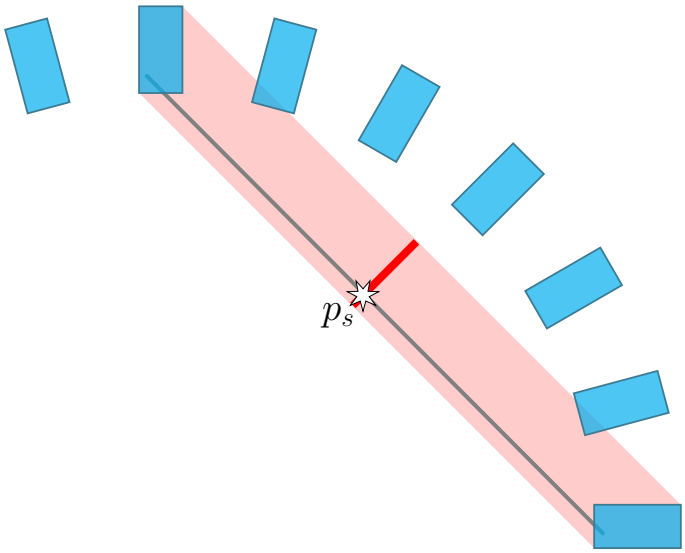


Figure 3.2.: Emission point uncertainty across TOR

time measurement errors or *electronics errors* – as explained previously J-PET scanner requires TOF information in order to reconstruct the position of the emission. This time information is measured separately for each event by specialized electronics. Even if we were able to reach an amazing tens of picoseconds precision, such precision still implies errors that have negative impact on reconstructed image resolution.

Considering what is above, we can instead express τ in terms of $P(\widetilde{\mathbf{e}} \mid v)$ – a probability that detected emission from voxel v was measured as $\widetilde{\mathbf{e}}$

$$P(\widetilde{\mathbf{e}} \mid v) = \frac{P(\widetilde{\mathbf{e}} \cap v)}{P(v)} \stackrel{\text{def}}{=} \frac{P(\widetilde{\mathbf{e}} \cap v)}{s(v)} \tag{3.5}$$

where $s(v) \overset{\text{def}}{=} P(v)$ denotes *sensitivity* – a probability of registering any emission originating from voxel v and $P(\widetilde{\mathbf{e}} \cap v)$ denotes an unconditional probability of an emission from voxel v measured as $\widetilde{\mathbf{e}}$. This measure is inherent to given detector and its geometry. Image space region where $s(v)$ is non-zero is called *field of view* (FOV).

Such representation forms a statistical model that can include the geometry and the errors of the scanner. Such definition can be considered independent from ρ, thus expressing only "transfer" probability between $\mathbb{R}^3 \times \mathbb{R}^3$ event space and $\mathcal{R}$ response space.

In the next sections we will introduce image reconstruction methods that rely on P. Later in this work we will present how to construct P statistical model for PET and particularly for J-PET scanner.

3.1. Naive point-to-point reconstruction

Point-to-point reconstruction is a most straightforward, performant, but also a naive method of image reconstruction simply ignoring existence of uncertainties and errors. Therefore these uncertainties and errors are transferred into the resulting image and manifest as noise and blur.

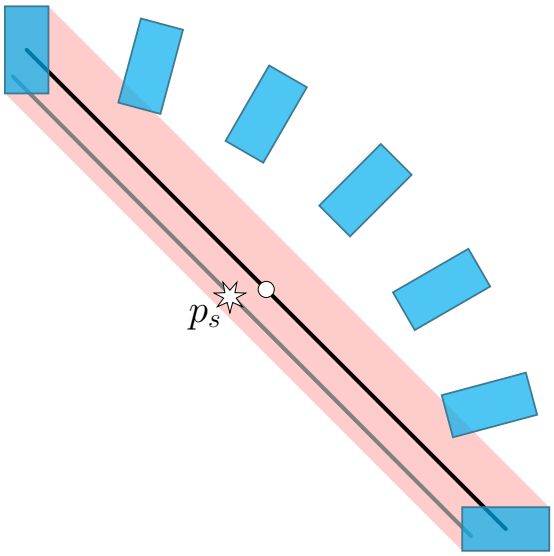


Figure 3.3.: Naive reconstruction mapping
(p_s – true emission point, white dot – point assumed by naive reconstruction)

This reconstruction method postulates well defined $\tau^{-1} : \widetilde{\mathbf{E}} \to \mathbb{R}^3$ that simply transforms all responses into some emission points. Since such τ^{-1} must ignore uncertainties, thus as shown in Figure 3.3 all responses having their real emissions points lying in a specific TOR, denoted with a star, are mapped into points, denoted with a circle, lying on the single line segment spanned by centers of TOR scintillators. Even if there were no time measurement errors, the naive reconstruction would still suffer from the geometric errors.

3.2. Back-projection

Back-projection is the simplest image reconstruction method using a statistical model to compute an output image. For each image voxel v its emission density $\rho(v)$ is computed as a sum of probabilities of each event

$$\rho(v) = \sum_{\widetilde{\mathbf{e}} \in \widetilde{\mathbf{E}}} P(\widetilde{\mathbf{e}} \,|\, v) \tag{3.6}$$

Effectively each response is back-projected onto the voxels according to the probability of registering a given response under the condition that the registered event was emitted from a given voxel.

This method however does not correctly solve the problem, because the output image ρ is blurred and does not correspond to the true source emission density that yielded $\widetilde{\mathbf{E}}$ response, neither maximizes $P\left(\widetilde{\mathbf{E}} \,|\, \rho\right)$. It is just a very rough approximation of the correct density and may be a subject for further processing, such as image sharpening or filtering, constituting another method – *filtered back-projection* (FBP) [50].

3.3. Maximum Likelihood Expectation Maximization (MLEM)

Expectation maximization [51–53] is an iterative method of finding *maximum likelihood* (ML) estimate [54] from observed data $\widetilde{\mathbf{E}}$, e.g. scan. The goal of EM is to find model parameter ρ for which the observed data is most likely i.e. $P\left(\widetilde{\mathbf{E}} \,|\, \rho\right)$ is a maximum.

The *likelihood* is a measure used to evaluate agreement between candidate set of statistical model parameters ρ, e.g. an emission density image, and an observation $\widetilde{\mathbf{E}}$, e.g. a set of photo-multiplier signal arrival time readouts.

Assuming $\widetilde{\mathbf{E}}$ is constant, the likelihood is a function of ρ

$$\mathcal{L}(\rho) = P\left(\widetilde{\mathbf{E}} \mid \rho\right) \tag{3.7}$$

When working with exponential family P, which is the case of PET, where the emission is modeled as Poisson process, we introduce *log likelihood* defined as

$$\ell(\rho) = \ln\ \mathcal{L}(\rho) \tag{3.8}$$

Upon each iteration EM algorithm is set to find new ML estimate $\rho^{(t+1)}$ satisfying

$$\ell(\rho^{(t+1)}) \geq \ell(\rho^{(t)}) \tag{3.9}$$

Since $\ln(x)$ is strictly increasing convex function, ρ value maximizing $\mathcal{L}$ maximizes ℓ as well.

In order to make finding ML estimate tractable, EM introduces *missing complete data* $\mathbf{E}$ related to measurement $\widetilde{\mathbf{E}}$ with

$$P\left(\widetilde{\mathbf{E}} \mid \rho\right) = \sum_{\{\mathbf{E}:\tau(\mathbf{E})=\widetilde{\mathbf{E}}\}} P\left(\widetilde{\mathbf{E}} \mid \mathbf{E}, \rho\right) P(\mathbf{E} \mid \rho) \tag{3.10}$$

Without going into details, explained in [51–53], finding EM iteration $\rho^{(t+1)}$ ML estimate can be reduced, under the assumption that $\rho > 0$, to

$$\rho^{(t+1)} = \arg\max_{\rho} \left\{ \sum_{\mathbf{E}} P\left(\mathbf{E} \mid \widetilde{\mathbf{E}}, \rho^{(t)}\right) \ln\ P\left(\widetilde{\mathbf{E}} \mid \mathbf{E}, \rho\right) P(\mathbf{E}, \rho) \right\} \tag{3.11}$$

Taking into account missing unobserved data $\mathbf{E}$ makes the maximization more convenient and simplified and can be applied to PET model postulating well defined $P(\widetilde{\mathbf{e}} \mid v)$ "transfer" probability for each response $\widetilde{\mathbf{e}} \in \widetilde{\mathbf{E}}$ and voxel v pair that can be used to compute $P\left(\widetilde{\mathbf{E}} \mid \mathbf{E}, \rho\right)$ and finally seek ρ parameter specific to $P(\mathbf{E} \mid \rho)$ image space distribution.

3.3.1. MLEM for PET image reconstruction

First application of MLEM to PET [20, 21] assumed *bin-mode*, where the scan was a histogram of coincidence counts in TORs, thus

$$\begin{aligned} \mathbf{E} &= \left\{ \mathbf{e} \overset{\text{def}}{=} n(v) : v \in \mathcal{V} \right\}, \\ \widetilde{\mathbf{E}} &= \left\{ \widetilde{\mathbf{e}} \overset{\text{def}}{=} n^*(t) : t \in \mathcal{T} \right\} \end{aligned} \tag{3.12}$$

where $n(v)$ denotes true unobserved emission count from a given voxel and $n^*(t)$ denotes measured coincidence count in given TOR. Given that PET emission model is Poisson process of independent $n(v)$ variables with mean $\rho(v)$

$$P(n \mid \rho) = \prod_v e^{-\rho(v)} \frac{\rho(v)^{n(v)}}{n(v)!} \tag{3.13}$$

$n^*(t)$ are independent Poisson variables with mean $\rho^*(t) = \sum_v \rho(v) P(t|v)$, hence the likelihood can given by

$$\mathcal{L}(\rho) = P(n^* \mid \rho) = \prod_t e^{-\rho^*(t)} \frac{\rho^*(t)^{n^*(t)}}{n^*(t)!} \tag{3.14}$$

This together with (3.11) leads to EM iteration step for bin-mode as shown in [21]

$$\rho(v)^{(t+1)} = \sum_{t \in \mathcal{T}} \frac{n^*(t)\, P(t \mid v)\, \rho(v)^{(t)}}{\sum\limits_{w \in \mathcal{V}} P(t \mid w)\, s(w)\, \rho(w)^{(t)}} \quad \text{for } v \in \mathcal{V} \tag{3.15}$$

where v, w denote voxels and $s(v)$ denotes scanner sensitivity for voxel v.

Starting $\rho^{(0)}$ must be finite and greater than zero. While $\rho^{(0)}$ can be chosen at will, as long as it satisfies these two constraints, all-ones $\rho^{(0)} \equiv 1$ is recommended in [20, 21]. Nevertheless, choice of $\rho^{(0)}$ can have an impact on the result.

As shown in [53], under some circumstances MLEM can converge to different estimates, including stationary points that are not a $\mathcal{L}$ maxima, e.g. saddle points that are not necessarily ML. This usually happens when the input to MLEM method contains not enough statistics. Particularly, in [20, 21] it was shown that $\#\mathcal{T} \gg \#\mathcal{V}$ – a number of TORs must be significantly greater than a number of voxels, in order to EM produce correct ML estimate.

3.4. Reconstruction modes

While original MLEM method applied to PET [20] was *bin-mode* reconstruction, MLEM can be applied to other reconstruction "modes" without loss of generality.

3.4.1. Bin-mode reconstruction

Bin-mode, histogram mode, or counting mode reconstruction originally presented in [20, 21] is not computationally demanding as the response size depends only on a number of TORs

$$\#\widetilde{\mathbf{E}}_{\text{binmode}} \leq \#\mathcal{T} = \frac{\#\text{detectors}\,(\#\text{detectors} - 1)}{2} \tag{3.16}$$

That is why bin-mode was used exclusively for PET image reconstruction until late 1990s, when the computer processor technology advancement provided enough compute power to explore other possibilities.

This representation is unfortunately barely suitable for using an extra time information. This is because $n^*(t)$ histogram requires a discrete number of t bins. So in order to use bin-mode for TOF reconstruction, time would need to be quantized. Depending on the number of quantization steps, the overall number of bins can grow drastically, increasing the number of operations in the algorithm. On the other hand, using a small number of quantization steps (below 10) to keep the number of operations low does not bring significant improvement to the reconstruction image quality.

3.4.2. List-mode reconstruction

Raising the number of bins in order to inject TOF information soon leads to the situation where most of these bins are empty and the rest hold small value of 1 or 2, making the overall number of bins greater than the number of events in a single scan. This has led to the idea of using an event information directly in the reconstruction rather than binning their number per TOR

$$\begin{aligned} \widetilde{\mathbf{E}} &= \{\widetilde{\mathbf{e}}_1, \widetilde{\mathbf{e}}_2, \ldots, \widetilde{\mathbf{e}}_N\}\,, \\ \widetilde{\mathbf{e}} &= (t, \widetilde{T}_{ul}, \widetilde{T}_{ur}, \widetilde{T}_{dl}, \widetilde{T}_{dr}) \end{aligned} \tag{3.17}$$

The working concept for 2D was demonstrated in late 1990s in [1] [55], showing that such representation can be also modeled as Poisson process and yields EM iteration step for list-mode

$$\rho(v)^{(t+1)} = \sum_{\widetilde{\mathbf{e}}\in\widetilde{\mathbf{E}}} \frac{P(\widetilde{\mathbf{e}}\,|\,v)\,\rho(v)^{(t)}}{\sum\limits_{w\in\mathcal{V}} P(\widetilde{\mathbf{e}}\,|\,w)\,s(w)\,\rho(w)^{(t)}} \quad \text{for } v \in \mathcal{V} \tag{3.18}$$

which is more generic representation of equation (3.15).

Full 3D implementations for list-mode had to wait next few years for more computing power available. Before 2011 there were several attempts to leverage modern computing techniques for PET tomography, starting with use of SIMD [23], followed by use of GPU [24, 25]. Finally, in 2011 the complete 3D image list-mode reconstruction implementations for classical crystal PET scanner were presented in [26, 27], employing GPU via CUDA programming interface.

3.4.3. Detected emission density reconstruction

Substituting $\rho(v)$ with $\frac{\rho'(v)}{s(v)}$ in above equation (3.18) where $\rho'(v) \equiv s(v)\,\rho(v)$ denotes detected emission density yields an alternative MLEM equation

$$\begin{aligned}
\rho(v)^{(t+1)} &= \sum_{\widetilde{\mathbf{e}}\in\widetilde{\mathbf{E}}} \frac{P(\widetilde{\mathbf{e}}\,|\,v)\,\rho(v)^{(t)}}{\sum\limits_{w\in\mathcal{V}} P(\widetilde{\mathbf{e}}\,|\,w)\,s(w)\,\rho(w)^{(t)}} \\
&= \rho(v)^{(t)} \sum_{\widetilde{\mathbf{e}}\in\widetilde{\mathbf{E}}} \frac{P(\widetilde{\mathbf{e}}\,|\,v)}{\sum\limits_{w\in\mathcal{V}} P(\widetilde{\mathbf{e}}\,|\,w)\,s(w)\,\rho(w)^{(t)}}
\end{aligned} \tag{3.19}$$

$$\begin{aligned}
\frac{\rho'(v)^{(t+1)}}{s(v)} &= \frac{\rho'(v)^{(t)}}{s(v)} \sum_{\widetilde{\mathbf{e}}\in\widetilde{\mathbf{E}}} \frac{P(\widetilde{\mathbf{e}}\,|\,v)}{\sum\limits_{w\in\mathcal{V}} P(\widetilde{\mathbf{e}}\,|\,w)\,\rho'(w)^{(t)}} \\
\rho'(v)^{(t+1)} &= \rho'(v)^{(t)} \sum_{\widetilde{\mathbf{e}}\in\widetilde{\mathbf{E}}} \frac{P(\widetilde{\mathbf{e}}\,|\,v)}{\sum\limits_{w\in\mathcal{V}} P(\widetilde{\mathbf{e}}\,|\,w)\,\rho'(w)^{(t)}}
\end{aligned} \tag{3.20}$$

This gives some basic optimization for MLEM method where ρ' detected emission density can be used throughout the reconstruction and can be converted to ρ once the reconstruction is finished. For the same results as with recommended starting all-ones $\rho^{(0)}$ for equation (3.18), $\rho^{(0)} \equiv s$ has to be used in alternative MLEM equation (3.19).

3.5. PET interaction model

The methods above postulate existence of $P(\widetilde{\mathbf{e}} \cap v)$ and bound $P(\widetilde{\mathbf{e}} \mid v)$ (3.5) "transfer" probability which represents a conditional probability that given response $\widetilde{\mathbf{e}}$ originates from image voxel v. They do not define how P is calculated or represented.

In order to understand how P is calculated, *intrinsic detector response function* (IDRF) has to be introduced. For an infinitely thin parallel coherent beam of gamma quanta, emitted from a source of intensity I_s – expressed as gamma quanta per second, originating from point p with direction θ, the number of photons detected by detector d per second depends on detector's IDRF g_d function

$$I_s g_d(\theta, p) \tag{3.21}$$

Given that g_d can be perceived as a probability of registering a single gamma ray emitted from p in direction θ towards detector d. The value of g_d depends on the length of the secant produced by the gamma ray intersecting scintillator d.

An interaction of the gamma quantum with the scintillator – depositing energy and causing scintillation – photons emission, can be modeled with *probability density function* (PDF) of the exponential attenuation law expressing probability of an interaction after distance l travelled within the scintillator and dependent l_s attenuation factor constant distinctive for scintillator type s

$$f_s(l) = \frac{l}{l_s} e^{-\frac{l}{l_s}} \tag{3.22}$$

and corresponding *cumulative distribution function* (CDF) representing a probability of an interaction with the scintillator along l long secant

$$F_s(l) = 1 - e^{-\frac{l}{l_s}} \tag{3.23}$$

Having that we can express g_d as a composition of F_s and $\text{secant}_d(\theta, p)$ function returning a length of the secant produced by a gamma beam emitted from point p towards detector d in direction θ

$$g_d(\theta, p) = F_s\left(\text{secant}_d(\theta, p)\right) = 1 - e^{-\frac{\text{secant}_d(\theta,p)}{l_s}} \tag{3.24}$$

Assuming for the moment that detectors do not occlude each other, so single beam cannot intersect more than one detector, we can define a probability for registering an emission originating from point p in direction θ in TOR $t = (u, d)$ as

$$P(t, p, \theta) = g_u(\theta, p) g_d(\theta + \pi, p) \tag{3.25}$$

and a probability for detecting an emission from p in TOR t

$$\begin{aligned} P(t \cap p) &= \int_{-\pi}^{\pi} P(t, p, \theta) \, \mathrm{d}\theta \\ &= \int_{-\pi}^{\pi} g_u(\theta, p) g_d(\theta + \pi, p) \, \mathrm{d}\theta \\ &= \int_{-\pi}^{\pi} (1 - e^{-l_s * \mathrm{secant}_u(\theta, p)})(1 - e^{-l_s * \mathrm{secant}_d(\theta + \pi, p)}) \, \mathrm{d}\theta \end{aligned} \tag{3.26}$$

Next we can determine $P(t \cap v)$ for voxel v integrating its points

$$P(t \cap v) = \int_{p \in v} \int_{-\pi}^{\pi} g_u(\theta, p) g_d(\theta + \pi, p) \, \mathrm{d}\theta \, \mathrm{d}p \tag{3.27}$$

Finally we calculate the sensitivity from $P(t \cap v)$ as

$$s(v) = \sum_{t \in \mathcal{T}} P(t \cap v) \tag{3.28}$$

and the conditional probability from equation (3.5) as

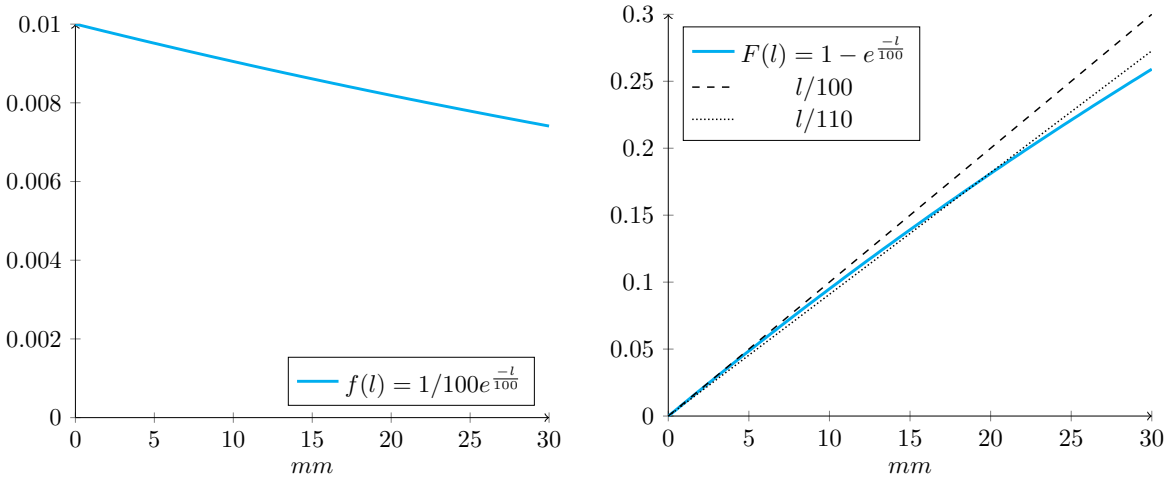
$$P(t \mid v) = \frac{P(t \cap v)}{s(v)} \tag{3.29}$$

This gives some intuition how P can be calculated and that secant function is a component of P that strictly depends on the detectors geometry and their spatial distribution in the scanner.

Unfortunately in many realistic cases we have detectors occluding each other for certain gamma quanta trajectories originating from FOV. This requires more complicated description than equation (3.26) to express the conditional probability that a

single gamma ray not detected by one detector, may leave it and hit another one behind the first. Such case is described in next Chapter 4 using an algorithmic approach.

3.5.1. J-PET scintillator attenuation



(a) PDF **(b)** CDF

Figure 3.4.: Attenuation PDF and CDF for $l_s = 100\,\text{mm}$

J-PET scintillator plastic has $l_s = 100\,\text{mm}$ attenuation factor [14]. Currently used J-PET scintillators are 19 mm thick and maximum secant length is $\approx 30\,\text{mm}$. This makes PDF (3.22) and CDF (3.23) for J-PET scintillators nearly linear as shown in Figure 3.4.

3.5.2. Analytic representation of J-PET statistical model

Analytic representation expressed as a programming language function or an expression dependent only on few input variables and constants is the most desired representation for P. It does not require any pre-computation or memory to store the P model. However providing an exact analytic function is simply not possible, because secant does not have well defined integral in most cases. For the simplest case of 2D barrel with circular scintillators integral (3.26) consists of elliptical functions composition not very suitable for computing. There were attempts to get analytic approximation of secant for rectangular 2D scintillator shapes in [56] using piecewise linear functions.

In case of J-PET, analytic P approximation for 2D detector frame, representing a plane along z axis in 3D detector, is presented in Chapter 6.

3.5.3. System matrix

System matrix is an alternative method of representing P model as a matrix of $P(t \mid v)$ values. These values may be computed with certain precision using a simulation, e.g. Monte-Carlo method. From the reconstruction point of view, system matrix is an external P model that can be loaded into the memory prior starting the reconstruction.

Even for a relatively small number of TORs and image voxels, system matrix may be significantly large and storing it directly in a generic form may be problematic. As shown in Figure 3.1 very small case of 100^3 image space and 100 detectors barely fits in the memory, while a realistic case of 256^3 image space and 192 detectors, which is default J-PET image reconstruction setup, does not fit in memory at all.

Image size	No. of detectors	Matrix size
100^3	100	$5 * 10^9$ elements
256^3	192	$\approx 309 * 10^9$ elements

Table 3.1.: System matrix size depending on image size and number of detectors

Fortunately PET system matrix has less than 1% non-zero elements and it is a perfect candidate for storing in a sparse matrix format. Method of constructing system matrix for J-PET and its sparse format is described in next Chapter 4.

3.6. Performance measures

Starting from 1994 *National Electrical Manufacturers Association* (NEMA) publishes NEMA NU-2 standard for performance measurements of PET devices. Latest version for the time being is 2-2012 [57]. NEMA NU-2 identifies the following performance measures

Spatial resolution measured with *point spread function* (PSF) *full width at half-maximum* (FWHM) and full width at 10% (FW10%) for ^{18}F point source at six locations:

$(1\,\text{cm}, 0, 0)$, $(10\,\text{cm}, 0, 0)$, $(0, 10\,\text{cm}, 0)$, $(1\,\text{cm}, 0, L/4)$, $(10\,\text{cm}, 0, L/4)$, $(0, 10\,\text{cm}, L/4)$ where $(0, 0, 0)$ denotes central point and L denotes detector depth along z-axis.

Sensitivity measures ability of PET device to detect annihilation radiation, equivalent to s defined in Section 4.4.2

Scatter fraction measures sensitivity to scattered quanta, explained in Section 1.3

$$\text{SF} = \frac{C_{\text{scatter}}}{C_{\text{scatter}} + C_{\text{true}}} \tag{3.30}$$

where C_{scatter} denotes number of detected emissions coming from scattering in the imaged object and C_{true} denotes true coincidences.

Noise equivalent counts

$$\text{NEC} = \frac{C_{\text{true}}}{1 + \frac{C_{\text{scatter}}}{C_{\text{true}}} + \frac{C_{\text{random}}}{C_{\text{true}}}} \tag{3.31}$$

where C_{random} denotes number of detected random coincidences not coming from single emission.

Image quality measures errors in image reconstruction in comparison to the reference image of the preset phantom simulating whole body composed of several spheres of a different activity.

In this work we will be concerned only by the spatial resolution, sensitivity and the image quality that directly depends on image reconstruction algorithms and the statistical model. Particularly we will present PSF FWHM and *normalized root mean square error* (NRMSE), described later, for each image reconstruction method introduced in next chapters. Other measures are subject of different works [7, 8] of our J-PET team members.

Most of the measures in this dissertation were taken on the simulated data, some were however done on real scans, e.g. two strip detector PSF FWHM presented in Chapter 6. In most cases, unless explicitly noted, we will be presenting performance measures for J-PET "big" barrel – a full scale prototype described in the next chapter.

3.6.1. Point spread function

Point spread function (PSF) is a response of a measuring instrument or an imaging system to a single point source. It is usually related to the spatial resolution of the imaging system and measurement errors, where more spreading occurs for low-resolution, high-error systems.

In case of PET, PSF is an image reconstruction of the PET scanner response to a "*point-like*" emitting object. While it is easy to simulate a virtual phantom with point's zero-dimensions and a non-zero emission intensity, in real life a physical phantom must have non-zero dimensions, as a radio-emitter probe e.g. ^{18}F must have some minimum volume to reach desired emission intensity. NEMA requires the source's diameter not to be greater than 1 mm.

3.6.2. PSF full width at half-maximum

Full width at half-maximum (FWHM) of function $f : D \to \mathbb{R}$ with single maximum $f_{\max}$ is a distance between two points $x_1, x_2 \in D$

$$FWHM(f) = |x_1 - x_2| \text{ where } f(x_1) = f(x_2) = \frac{f_{\max}}{2} \tag{3.32}$$

FWHM is used to effectively measure the spread of PSF. Its visual interpretation is depicted in Figure 3.5. In case of probability distribution, FWHM is strictly related to its standard deviation, e.g. normal (Gaussian) distribution $FWHM = 2.3548\,\sigma$.

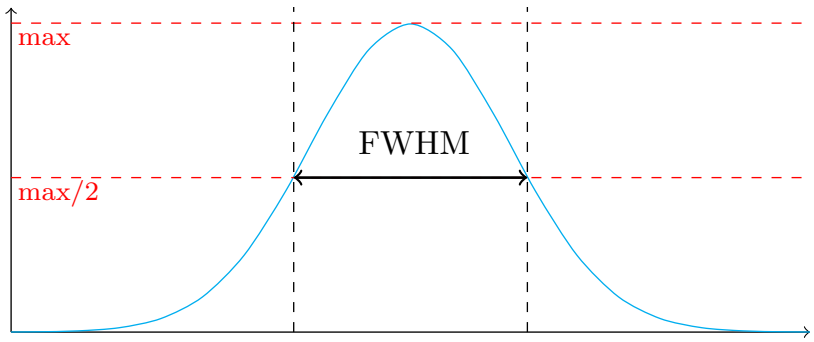


Figure 3.5.: Full width half-maximum visual representation

3.6.3. Normalized root mean square error

Normalized Mean Squared Error (NRMSE) may be used to quantify goodness of the image reconstruction *Img* relative to reference image *Ref*

$$\mathrm{NRMSE}(Ref, Img) = \sqrt{\frac{\sum\limits_{v \in \mathcal{V}} \big(Ref(v) - Img(v)\big)^2}{\sum\limits_{v \in \mathcal{V}} Ref(v)^2}} \tag{3.33}$$

where $\sum\limits_{v \in \mathcal{V}}$ denotes sum over all voxels in *Ref* and *Img* images.

In the reconstruction quality measurements presented in this work *Ref* will denote known real emission density image and *Img* will denote tested image reconstruction output.

Chapter 4.

2D barrel PET simulation and system matrix calculation

As described in previous chapter, determining P model analytically is a difficult or even an impossible task. However, we may use different methods to estimate P integral equation (3.26) from Section 3.5. One of such methods is Monte-Carlo simulation. This chapter will present method of computing P for 2D PET barrel of detectors distributed on single or multiple rings.

The problem is defined as follows: for an input scanner layout describing placement of individual detectors in the 2D space we want to generate map of entries defined as $P(t \mid i)$, where $P(t \mid i)$ is the probability that detected event originating from the pixel i was detected by the TOR t.

We will be effectively computing approximation of $P(t \cap i)$ integrating $P(t, p, \theta)$ over pixel i and angle θ. Our simulation will take also into account cases where detectors occlude each other, opposite to simplified integral (3.26) presented in the previous chapter. $P(t \mid i)$ can be later derived from $P(t \cap i)$ using equation (3.29).

There exist generic simulation tools, e.g. GATE [58], suitable for performing detailed physics simulations specific to PET devices. Our solution is however tailored for performing simulation of two gamma quanta emission in exactly parallel back-to-back direction and detection modeled with attenuation law PSF equation (3.22), without taking into account scatter fraction, that can be reduced in J-PET to the similar amount as in the typical tomographs [59]. In combination with use of GPGPU, we are able to deliver the simulation results few orders of magnitude faster than GATE tools.

4.1. Generic CPU system matrix Monte-Carlo simulation

In order to compute approximation of $P(t \cap i)$ integral (3.27) we perform Monte-Carlo simulation of gamma quanta detection process described in Section 3.5.

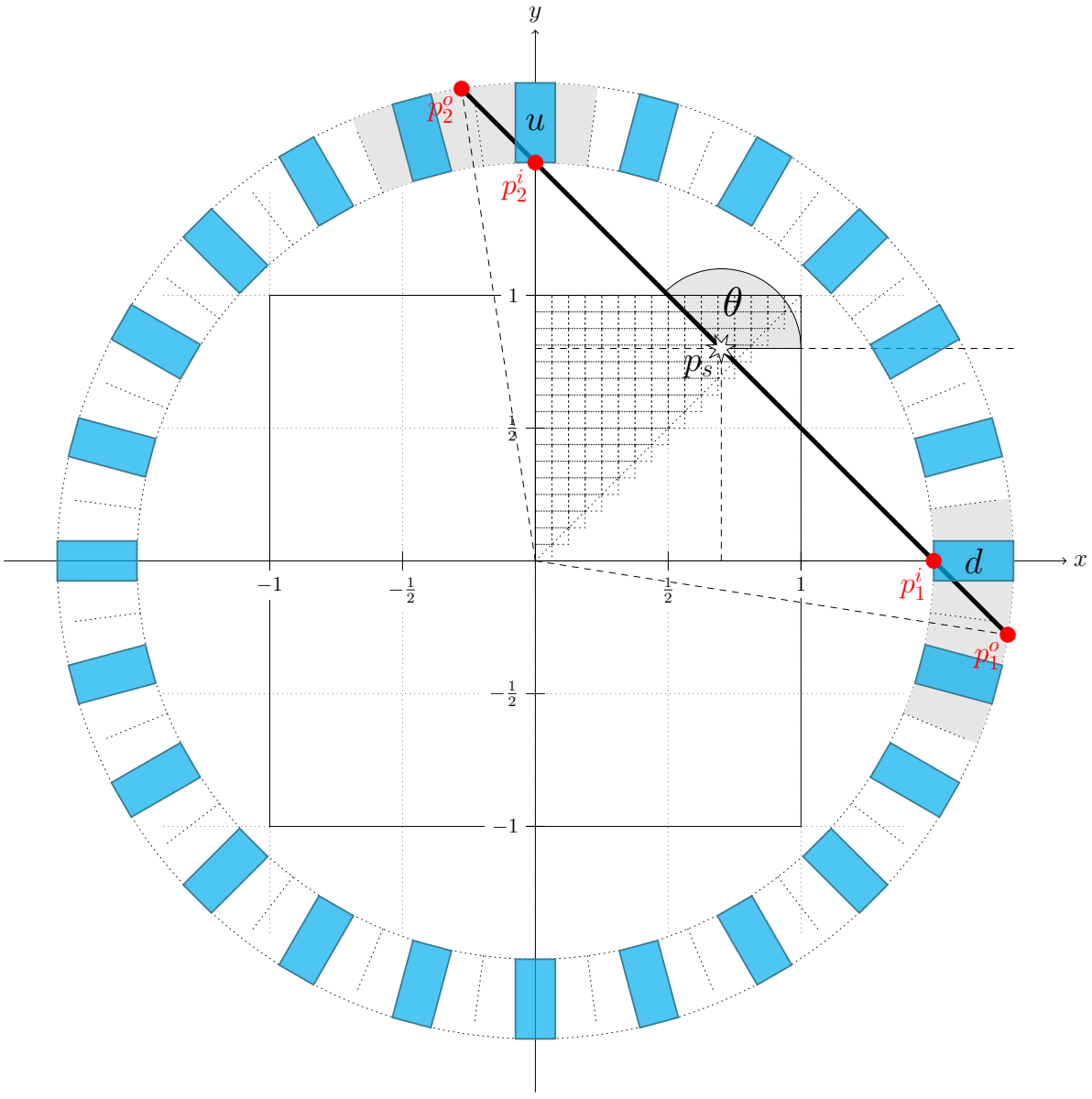


Figure 4.1.: Monte-Carlo simulation

For each pixel i we simulate emission of given number of gamma quanta with random direction angle θ and random emission point p_s belonging to pixel i as shown in Figure 4.1. The number of emissions must be high enough in order to provide good approximation. We simulate at least 10^6 up to $100 * 10^6$ emissions.

In order to reduce the number of operations and the overall simulation, time we leverage eight-fold detector symmetry to calculate only 1/8 of the whole pixel grid. Further reduction of calculations is done by observing the fact, that if either there is no intersection on one side or Monte-Carlo toss gives negative result, we can completely skip all the calculations for the other side of emission beam.

The simulation produces an output called by us *triangular sparse matrix*. Producing *full* sparse matrix requires duplicating each triangular matrix entry 8 times for each symmetry. While producing symmetric pixel entries is straightforward for each (x, y), and consists of simply changing the sign and reversing order, e.g. $(-y, x)$, computing counterpart symmetric detector to d in symmetry s is slightly more complex and requires Algorithm 5, where & denotes bitwise *and* operator and % denotes integer modulo operator.

Algorithm 5 Detector symmetric d detector for given symmetry s

function RINGSYMMETRICDETECTORINDEX(d, s, $N_{\text{detectors}}$)
  if s & $1 \neq 0$ **then**
    $d \leftarrow (N_{\text{detectors}} - d)$ % $N_{\text{detectors}}$
  end if
  if s & $2 \neq 0$ **then**
    $d \leftarrow ((N_{\text{detectors}} + \frac{N_{\text{detectors}}}{2}) - d)$ % $N_{\text{detectors}}$
  end if
  if s & $4 \neq 0$ **then**
    $d \leftarrow ((N_{\text{detectors}} + \frac{N_{\text{detectors}}}{4}) - d)$ % $N_{\text{detectors}}$
  end if
  return d
end function

We also use few methods to speed-up detector intersections resolution by given emission beam. The first method lets us identify possible intersections at early stage without exact polygon intersection testing. It works for single ring of detectors and relies on dividing the detector ring into sections, each enclosing single detector as shown in Figure 4.1. For each emission point p_s and angle θ, we first determine points intersecting inner and outer circle. With these points we can find the sections belonging to this intersection and the order of beam traversal from the most "inner" to the most "outer". The number of sections is less than 3 in most of the cases. Finally, we use exact polygon intersection calculation for each detector belonging to these few sections to ensure we have an intersection.

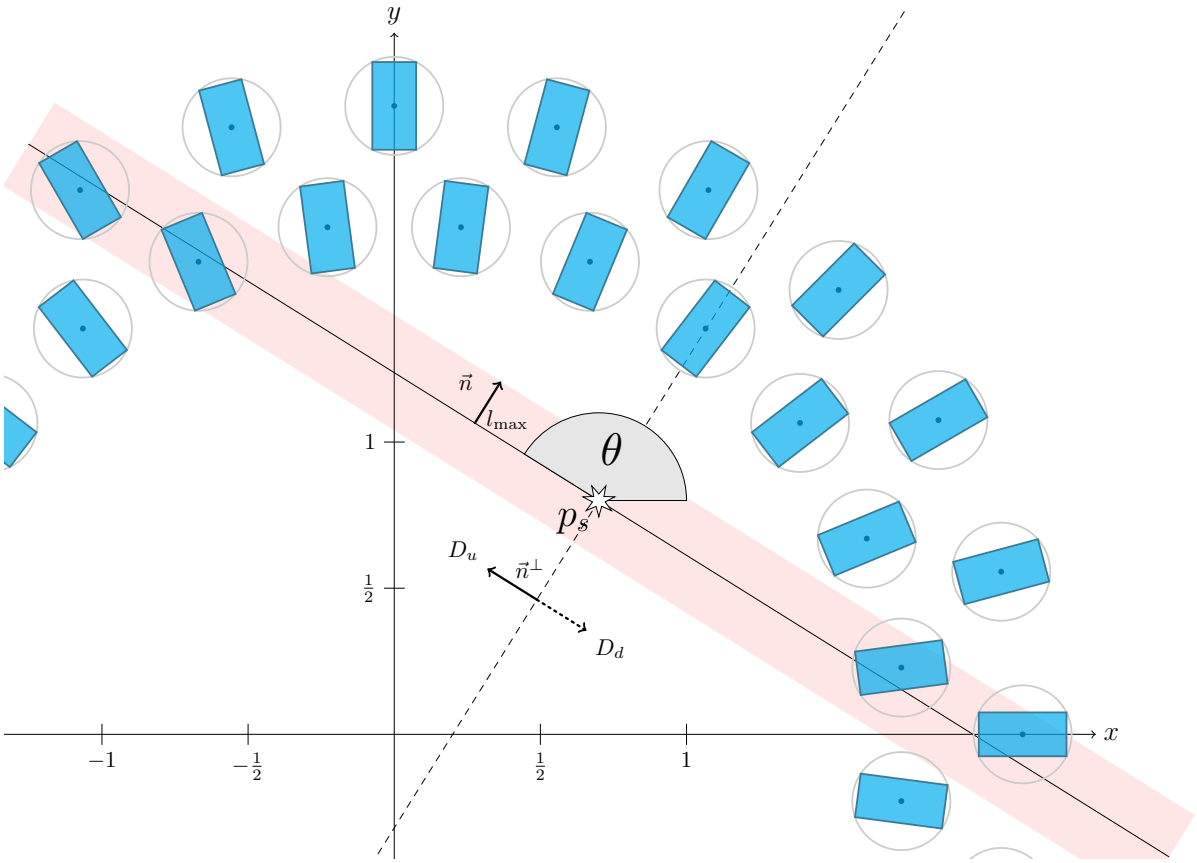


Figure 4.2.: Preliminary intersection check based on detector center to TOR distance

Nevertheless, single detector ring may be not desired layout. That is why we have developed another method for determining intersections on generic layout of detectors, e.g. multiple rings, depicted in Figure 4.2. This methods assumes that all detectors can be inscribed into circles, and each such circle does not intersect with other detectors circle. This constraint holds for J-PET scanner, where the detector circle is effectively defined by the photo-multiplier tube having circular scintillator attachment surface.

Assuming such detector inscription, we may perform rough test for intersection by simply measuring the distance of the circle center point to the emission beam – area marked red in Figure 4.2.

In order to compute this distance we first fix normal vector of emission beam direction $\vec{n} \stackrel{\text{def}}{=} (A, B) = (\sin(\theta), -\cos(\theta))$, next we determine $C = \vec{p_s} \cdot \vec{n}$. Once we

have $\vec{n}$ and C we can quickly compute distance of p to emission beam as

$$\text{distance}(p, \vec{n}, C) = \vec{p} \cdot \vec{n} - C \tag{4.1}$$

Such operation is very fast and consists only of 2 multiplications and 2 additions and can be expressed as only 2 *fused multiply-add* (FMA) instructions present on many architectures, including modern *x86* and GPU processors.

Using this distance measurement method, we filter the detectors testing if their circle p_c center point distance to the emission beam is less than R

$$D_{\text{close}}(\theta, p_s) = \left\{ d \in 0..N_{\text{detectors}} : \text{distance}(\vec{p_{c_d}}, \vec{n_\theta}, c_{\theta,p_s}) < R_d \right\} \tag{4.2}$$

All detectors failing this test cannot be intersected by the emission beam. Usually only few detectors pass this test.

Next we need to determine the order in which the gamma ray is traversing detectors, which is the order we will perform exact intersection testing and interaction simulation in. Under assumption that all detectors are inscribed in circles that not intersect each other, this order can be determined from the distance of the detector's center point p_c to line perpendicular to emission beam – denoted with dashed line in Figure 4.2.

This is explained in Figure 4.3 – if y-axis represents emission direction, for any circle intersected by y-axis in two points, one intersection point lies above the circle center and the second lies below. Because a and b circles are separated – not intersect each other, therefore $y(a) < y(a_u) < y(b_d) < y(b)$, so circle a is intersected before circle b only and only if circle $y(a) < y(b)$.

We fix vector $\vec{n}^\perp \overset{\text{def}}{=} (-B, A)$ perpendicular to $\vec{n}$ and $C^\perp = \vec{p_s} \cdot \vec{n}^\perp$ and sort the detectors which passed previous test by measure o

$$o(d) \overset{\text{def}}{=} \text{distance}(\vec{p_{c_d}}, \vec{n_\theta}^\perp, C^\perp_{\theta,p_s}) \tag{4.3}$$

The sign of $o(d)$ represents the particular gamma quanta in the quanta pair, either "upper" or "lower". The magnitude represents the single beam traversal order. Starting with detectors giving lowest $o(d)$, we determine the secant length with exact polygonal intersection and perform interaction simulation with attenuation model (3.23).

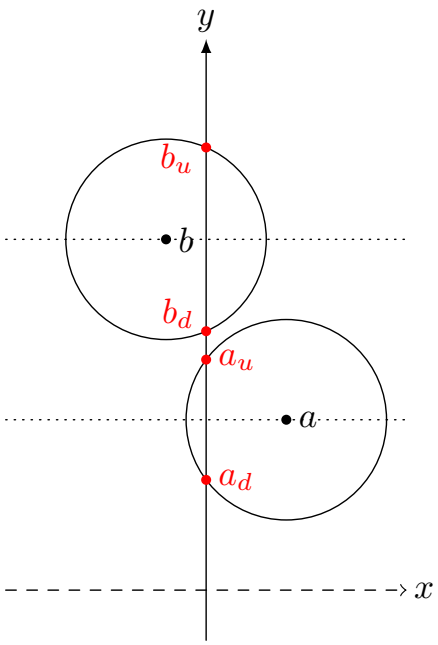


Figure 4.3.: Separate circle intersection order

Our exact polygon intersection test also employs use of normal $\vec{n}$ and c and leverages the fact that when intersection occurs with some face of polygon spanned by two sibling points $p_{(j)}$ and $p_{(j+1)}$ then

$$\text{sign}\left(\text{distance}(p_{(j)}, \vec{n}, C)\right) \neq \text{sign}\left(\text{distance}(p_{(j+1)}, \vec{n}, C)\right) \tag{4.4}$$

Once we encounter an intersection, we perform the interaction simulation using the attenuation model and the random value $r \in [0, 1)$. Next we compute logarithm $l = -l_s \ln(r)$ which gives us interaction depth l according to CDF (3.23). If l is below the intersection secant length then hit occurs, otherwise we assume gamma quanta did not hit the matter in the scintillator and we move to the next detector in order $o(d)$. This leads altogether to the Algorithms 6 and 7.

We were also exploring use of some more advanced spatial data structures such as quad-trees or KD-trees, but they were not fitting J-PET geometry very well, moreover using such data structures for massively parallel GPU implementation is problematic. Nevertheless, the second method working with any geometry shows just marginal performance drop comparing to the first tailored specifically for single ring of detectors, when benchmarked for scanner containing 192 detectors.

Algorithm 6 Monte-Carlo simulation for single system matrix pixel

function SMPIXELMONTECARLO(i, D, A_{model}, $N_{emissions}$)
  for $i \leftarrow 1, N_{emissions}$ **do**
    $e \leftarrow$ random $\in [0, \pi) \times [p_{\min x}, p_{\max x}) \times [p_{\min y}, p_{\max y})$ ▷ get random event
    $\vec{n}, C \leftarrow$ direction coef(e) ▷ emission direction coefficients
    $D_{\text{close}} \leftarrow \{d \in D : \text{distance}(d, \vec{n}, C) < R\}$ ▷ detectors close to the beam
    $D_u, D_d \leftarrow$ traverse order$(D_{\text{close}}, \vec{n}, C)$ ▷ upper and lower detectors
    for all $u, d \in D_u, D_d$ **do** ▷ traverse them
      if $\vec{n}, C$ intersects polygon $u \wedge d$ **then**
        intr $\leftarrow \{(u, \|\text{secant } u\|), (d, \|\text{secant } d\|)\}$ ▷ store intersection
        Quit loop.
      end if
    end for
    if $A_{model}(\text{intr})$ **then** ▷ interacted using attenuation model
      $M(u, d, i) \leftarrow M(u, d, i) + 1$ ▷ update $P(t \cap i)$
    end if
  end for
  return M ▷ return non-normalized $P(t \cap i)$ approximation
end function

Algorithm 7 Complete Monte-Carlo simulation for system matrix

function SMMONTECARLO($TORS$, A_{model}, $N_{emissions}$)
  $sparse \leftarrow \{\}$ ▷ create empty sparse matrix
  for $i \in \mathcal{I}$ where $i_y \geq i_x$ **do** ▷ 1/8 of pixels
    $M \leftarrow$ SMPixelMonteCarlo$(i, D, A_{model}, N_{emissions})$
    $sparse \leftarrow sparse \cup \{(M, i)\}$ ▷ merge pixel emissions
  end for
end function

4.1.1. Sparse partial system matrix representation

The output of the Monte-Carlo simulation consists of number of detected emissions per each pixel and TOR. As shown in Table 3.1 in previous chapter, storing this as plain matrix is not feasible. Therefore, we utilize special sparse partial system matrix representation explained in Figure 4.4.

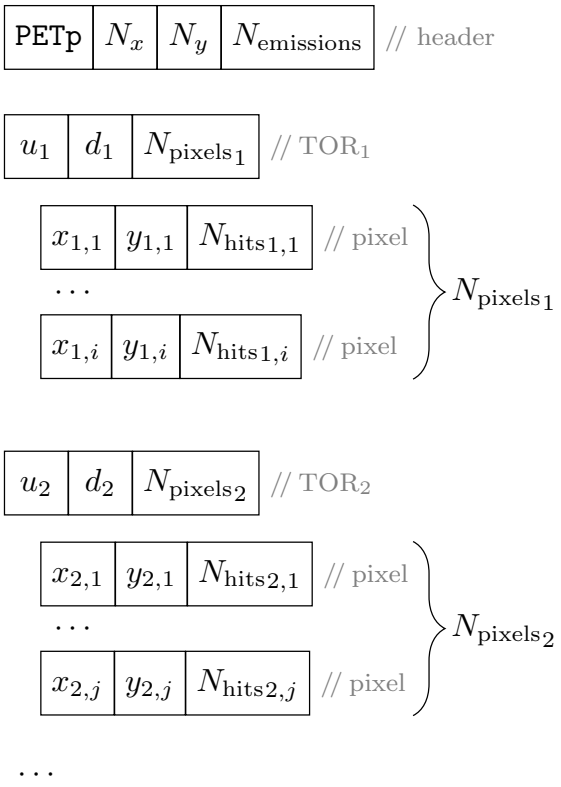


Figure 4.4.: TOR-major sparse matrix structure
($N_x \times N_y$ – image space dimensions, $N_{\text{emissions}}$ – simulated emissions)

4.1.2. Time of flight (TOF) consideration

TOF information Δ, representing measured position along TOR, may be injected into sparse matrix format quantizing Δ with step q_{step} and then treating quantized time value $\widehat{\Delta}$ as sub-TOR, so for TOF enabled matrix TOR can be represented by

$$\begin{aligned} t &= (u, d, \widehat{\Delta}), \\ \widehat{\Delta} &= \text{quantize}(\Delta, q_{\text{step}}) = \left\lfloor \frac{\Delta}{q_{\text{step}}} + \frac{1}{2} \right\rfloor \end{aligned} \tag{4.5}$$

Our J-PET simulation tools offer optional TOF mode for simulating system matrix, producing TOF system matrix for given q_{step} quantization step.

4.1.3. Temporary pixel-major storage

Sparse matrix representation is not suitable for storing temporary simulation results produced during Algorithm 6 execution. `SMPixelMonteCarlo` function main loop final update needs random access for each TOR. Therefore, we allocate space for all TORs. This can look like a waste of space, but since we process only one pixel at the time we need to keep only $\frac{N_{\text{detectors}}(N_{\text{detectors}}-1)}{2}$ values in the memory. Once values for currently processed pixel are completed, they are merged into space-efficient sparse format.

4.2. Parallelization

Simulation can be parallelized either by running `SMPixelMonteCarlo` simulation for several pixels at once or by running several simulations at once for single pixel.

The first approach is less memory space efficient, because it needs $T\frac{N_{\text{detectors}}(N_{\text{detectors}}-1)}{2}$ elements to be stored the memory, where T is number of parallel threads. The second is more space efficient – required space does not depend on number of threads, however it requires atomic memory access.

For generic CPU implementation we have chosen to use first approach, as modern CPUs do not have massive number of physical threads, so T is relatively low and we don't need to worry about available memory. Also first approach of running pixels separately ensures that there is no cache invalidation, that could likely occur when using second approach.

4.3. GPU accelerated implementation

J-PET Monte-Carlo simulation [60] is implemented on GPU using CUDA programming interface following second approach – all CUDA threads are processing single pixel. `SMPixelMonteCarlo` is implemented as single CUDA kernel launched sequentially per each pixel by supervisor CPU program.

All of the generic optimizations described above were carried into GPU implementation as well, thanks to C++11 support in CUDA compiler introduced beginning year 2015.

We do not worry about cache invalidation, simply because we do not use cache on GPU. We use fast built-in atomics to increment emission detection counts on TORs. For the number of detectors higher than 100, atomics collision occurrence is very unlike, therefore atomics do not slow down the algorithm – as proven by one of our benchmarks.

In order to reduce accesses to global memory, we keep all the geometry information in on-die shared memory, which is orders of magnitude faster than global memory. This memory is shared across all threads of single multi-processor, and while it is relatively small – up to 48 KB for NVIDIA GPUs, it is sufficient to store all necessary data structures. Therefore, all intersection calculations do not need any global memory access, which is only necessary for storing emission detection count, once for each simulated interaction.

Both our implementations – generic and GPU accelerated, use common *Scanner* class, that contains complete geometry description and methods for resolving intersections. This makes the GPU code lean and readable. However, in order to embed this class instance in the shared memory, we had to find a workaround, as it is not possible instantiate class with non-trivial constructor in shared memory using CUDA. That is why we instantiate it on the CPU and then we copy it using special helper to the GPU's shared memory. Additionally, we had to ensure that all geometry data is embedded directly in the class instance itself and we have same class structure memory layout on both CPU and GPU.

Since Monte-Carlo relies on efficient random number generator (RNG), we use our own Tausworthe RNG implementation presented in [49] using fast bitwise and fused multiply-add (FMA) operations. Each thread uses own RNG state kept in fast local register, therefore no memory or I/O access is needed for RNG. Local RNG states are seeded once with using global RNG on algorithm startup.

4.4. Results

4.4.1. Performance benchmark

No. of detectors	CPU[1,2]	GPU[3]	Speedup
48	410.37 ms	21.72 ms	18.89
96	479.43 ms	29.02 ms	16.52
192	697.22 ms	38.23 ms	18.24

[1] *GeForce GTX 980 Ti* 1 Ghz, 6 GB, 5 632 Gflops, 336 GB/s, CUDA 7.5 -O3

[2] Best compiler result among GCC 4.8, GCC 5.3 and ICC 16.0

[3] *Intel Xeon E5-2699v3* 3.2 GHz, 6 core, $\approx$ 307 Gflops, 51 GB/s, GCC 5.3 -O3

Table 4.1.: CPU vs GPU benchmark for system matrix Monte-Carlo ($10 * 10^6$ emissions per pixel, 200^2 image space, 4 mm pixel size, average time of simulation for single pixel)

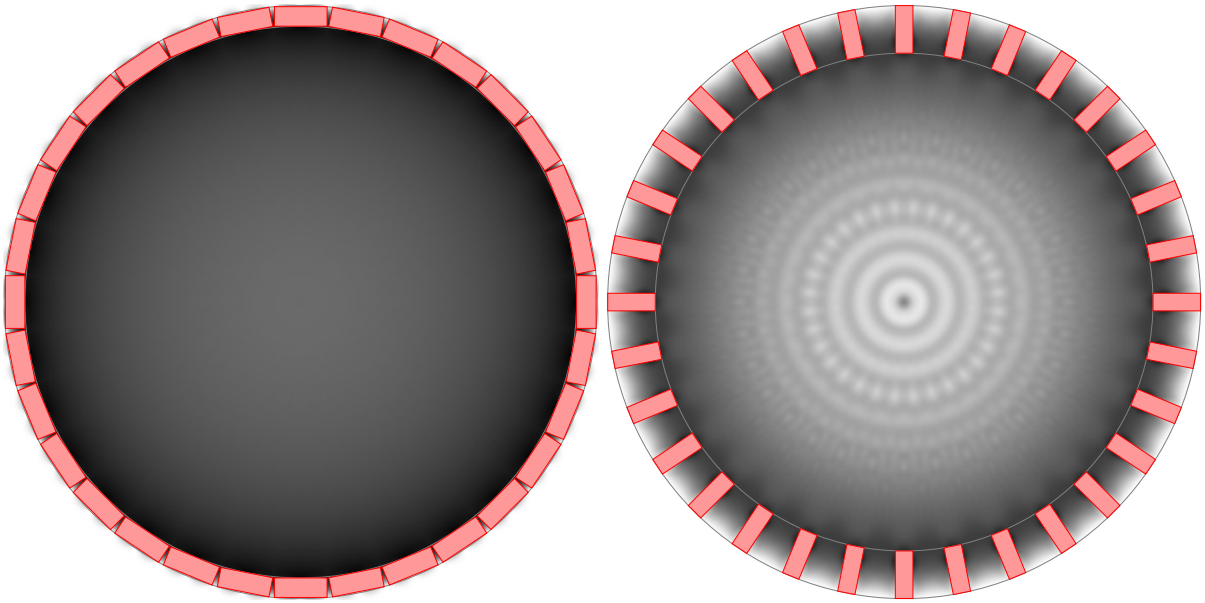
The time of per pixel simulation Algorithm 6 does not depend on number of pixels in system matrix. The relation of the overall simulation Algorithm 7 time to number of pixels is linear. Moreover, the simulation time increases with number of detectors, as shown in Table 4.1, which is expected. The main limitation of our implementation is a number of active blocks per *SMX*, caused by per-thread register limit, which allows *SMX* to run only two threads block, that is not enough to hide memory access latency.

Altogether it takes 4 minutes to compute system matrix for 192 detectors J-PET "big" barrel, 200^2 pixel grid and 10 million emissions from each pixel.

4.4.2. Simulated scanner sensitivity analysis

After performing system matrix Monte-Carlo simulation we may determine simulated scanner sensitivity $s(v)$ using equation (3.28). Our tools produce visualization for $s(v)$ additionally to system matrix output. This allows to investigate if given scanner's geometry covers completely desired field of view (FOV) – area lying inside the scanner and if the FOV does not contain "blind spots" where sensitivity is zero or near zero, meaning that emissions originating from such areas could not be detected at all by any of detector pairs.

Tightly packed scintillators layout as shown in Figure 4.5a, where each scintillator fits tight its neighbors, guarantee that "blind spots" will not appear. Such layout may be however either difficult to realize in practice, because we need to take into account space occupied by photomultipliers attached to scintillators and other elements such as cables or scintillators insulation foil. In such case gaps between scintillators manifest themselves as small pits or stripes on sensitivity map image as shown in Figure 4.5b.



(a) Tightly packed parallel scintillators **(b)** Loosely packed perpendicular scintillators

Figure 4.5.: Example sensitivity map for 2 layouts 32 detector 2D barrel (dark areas denote high sensitivity)

4.5. Phantom response simulation

Monte-Carlo may be not only applied to system matrix generation, but also to generate response for virtual phantom. A phantom is an object that can be measured by medical instrument in order to determine this instrument's properties and characteristics. Such phantom may be a physical object containing radio-emitter for PET measurement. In our simulation we use virtual phantom that exists only in our software.

Such virtual phantom consists of geometrical shapes, e.g. ellipses that have assigned center point position, axes length, rotation and emission intensity as shown in Figure 4.6.

```
ellipse  0.0   0.0    0.69   0.92   0     0.05
ellipse  0.0  -0.0184 0.6624 0.874  0     0.5
ellipse  0.22  0.0    0.11   0.31 -18     0.15
ellipse  0.0   0.35   0.21   0.25   0     1.0
ellipse  0.0  -0.1    0.046  0.046  0     0.75
ellipse -0.08 -0.605  0.046  0.023  0     0.75
ellipse  0.06 -0.605  0.023  0.046  0     0.75
ellipse  0.54 -0.451  0.03   0.2  -24.12  0.75
```

Figure 4.6.: Example for Shepp-like phantom description

Generating phantom response is similar to system matrix Monte-Carlo simulation, but instead of taking random point from currently processed pixel, we take random point from emission shapes according to their emission intensity and we perform the same emission detection simulation.

Currently there exists only implementation for CPU. Since generating the phantom response require significantly less total emission simulations, there is no need to develop optimized GPU implementation at this moment. Our implementation is able to produce additional image output that can serve as preview and also as reference image when measuring NRMSE (3.33).

Chapter 5.

2D barrel PET system matrix based MLEM reconstruction

Once system matrix is generated for given PET detector geometry and configuration, we may perform image reconstruction for specific response data. This response data may be either produced by real physical scanner device or separate simulation, e.g. virtual phantom simulation from previous Chapter 4.

System matrix based reconstruction is bin-mode reconstruction, therefore performing it requires computing ρ estimate from TOR bin t with count $n(t)$ using 2D version of equation (3.15) introduced in Section 3.3.1

$$\rho(i)^{(t+1)} = \sum_{t \in \mathcal{T}} \frac{n(t)\, P(t \mid i)\, \rho(i)^{(t)}}{\sum\limits_{j \in \mathcal{I}} P(t \mid j)\, s(j)\, \rho(j)^{(t)}} \quad \text{for } i \in \mathcal{I} \tag{5.1}$$

Since system matrix carries $P(t \cap i)$ not $P(t \mid i)$ values, using substitution from equation (3.29) in equation (5.1) with P we get formula used in our algorithm

$$\begin{aligned} \rho(i)^{(t+1)} &= \sum_{t \in \mathcal{T}} \frac{n(t) \frac{P(t \cap i)}{s(i)}\, \rho(i)^{(t)}}{\sum\limits_{j \in \mathcal{I}} \frac{P(t \cap j)}{s(i)}\, s(j)\, \rho(j)^{(t)}} \\ &= \frac{1}{s(i)} \sum_{t \in \mathcal{T}} \frac{n(t)\, P(t \cap i)\, \rho(i)^{(t)}}{\sum\limits_{j \in \mathcal{I}} P(t \cap j)\, \rho(j)^{(t)}} \end{aligned} \tag{5.2}$$

5.1. Generic CPU implementation

Implementing (5.2) directly yields Algorithm 8 of complexity $\mathcal{O}(\#\mathcal{T}\#\mathcal{I}^2)$ where $\#\mathcal{T}$ denotes number of TORs and $\#\mathcal{I}$ denotes number of pixels. This complexity can be reduced to $\mathcal{O}(\#\mathcal{T}\#\mathcal{I})$ by reordering loops as shown in Algorithm 9 proposed in [61].

Algorithm 8 Naive 2D barrel reconstruction

$\rho_{\text{new}} \leftarrow 0$ ▷ start with zero ρ_{new}
for $i \in \mathcal{I}$ **do**
  for $t \in \mathcal{T}$ **do**
    denom $\leftarrow 0$ ▷ compute denominator
    for $j \in \mathcal{I}$ **do**
      denom $\leftarrow$ denom $+ P(t \cap j)\,\rho(j)$
    end for
    $\rho_{\text{new}}(i) \leftarrow \rho_{\text{new}}(i) + \frac{n(t)\,P(t \mid i)\,\rho(i)}{\text{denom}}$
  end for
end for
for $i \in \mathcal{I}$ **do** ▷ correct ρ against sensitivity
  $\rho_{\text{new}}(i) \leftarrow \frac{\rho_{\text{new}}(i)}{s(i)}$
end for

Algorithm 9 Re-ordered 2D barrel reconstruction

$\rho_{\text{new}} \leftarrow 0$ ▷ start with zero ρ_{new}
for $t \in \mathcal{T}$ **do**
  denom $\leftarrow 0$ ▷ compute denominator
  for $i \in \mathcal{I}$ **do**
    denom $\leftarrow$ denom $+ P(t \cap i)\,\rho(i)$
  end for
  for $i \in \mathcal{I}$ **do**
    $\rho_{\text{new}}(i) \leftarrow \rho_{\text{new}}(i) + \frac{n(t)\,P(t \cap i)\,\rho(i)}{\text{denom}}$
  end for
end for
for $i \in \mathcal{I}$ **do** ▷ correct ρ against sensitivity
  $\rho_{\text{new}}(i) \leftarrow \frac{\rho_{\text{new}}(i)}{s(i)}$
end for

Further optimization can be obtained from sparse matrix representation, that keeps only non-zero $P(t \cap i)$ values in TOR-major order resulting in final Algorithm 10 of complexity $\mathcal{O}(\max(\#\text{pixels}(\mathcal{T}))\mathcal{I})$ where $\#\text{pixels}(\mathcal{T})$ denotes number of pixel entries for given TOR in sparse matrix structure described in Figure 4.4.

Algorithm 10 Optimized 2D barrel reconstruction

$\rho_{\text{new}} \leftarrow 0$ ▷ start with zero ρ_{new}
for $t \in \mathcal{T}$ **do**
  denom $\leftarrow 0$ ▷ compute denominator
  for $i \leftarrow$ pixels(t) **do** ▷ iterate through non-zero P pixels
    denom $\leftarrow$ denom $+ P(t \cap i)\,\rho(i)$
  end for
  for $i \leftarrow$ pixels(t) **do** ▷ iterate through non-zero P pixels
    $\rho_{\text{new}}(i) \leftarrow \rho_{\text{new}}(i) + \frac{n(t)\,P(t \cap i)\,\rho(i)}{\text{denom}}$
  end for
end for
for $i \in \mathcal{I}$ **do** ▷ correct ρ against sensitivity
  $\rho_{\text{new}}(i) \leftarrow \frac{\rho_{\text{new}}(i)}{s(i)}$
end for

5.2. GPU accelerated implementation

GPU accelerated reconstruction is implemented using CUDA programming interface. CUDA parallelization relies on processing each t bin separately by each CUDA thread and it is relatively simple as the bin-mode reconstruction is computationally not very demanding.

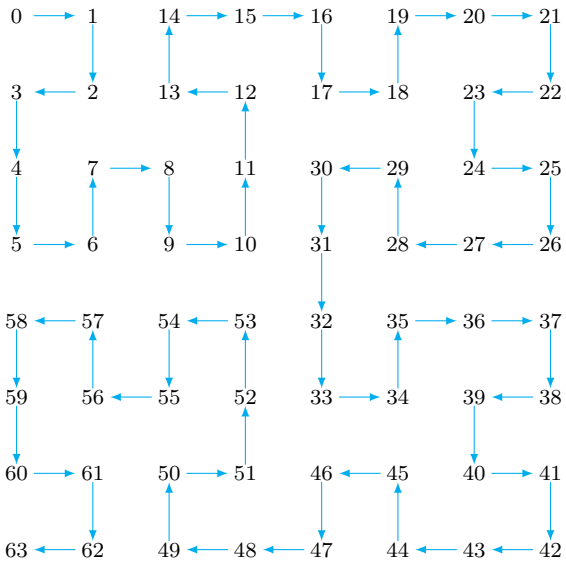


Figure 5.1.: Hilbert curve 64×64 texture 2D spatial to linear memory mapping

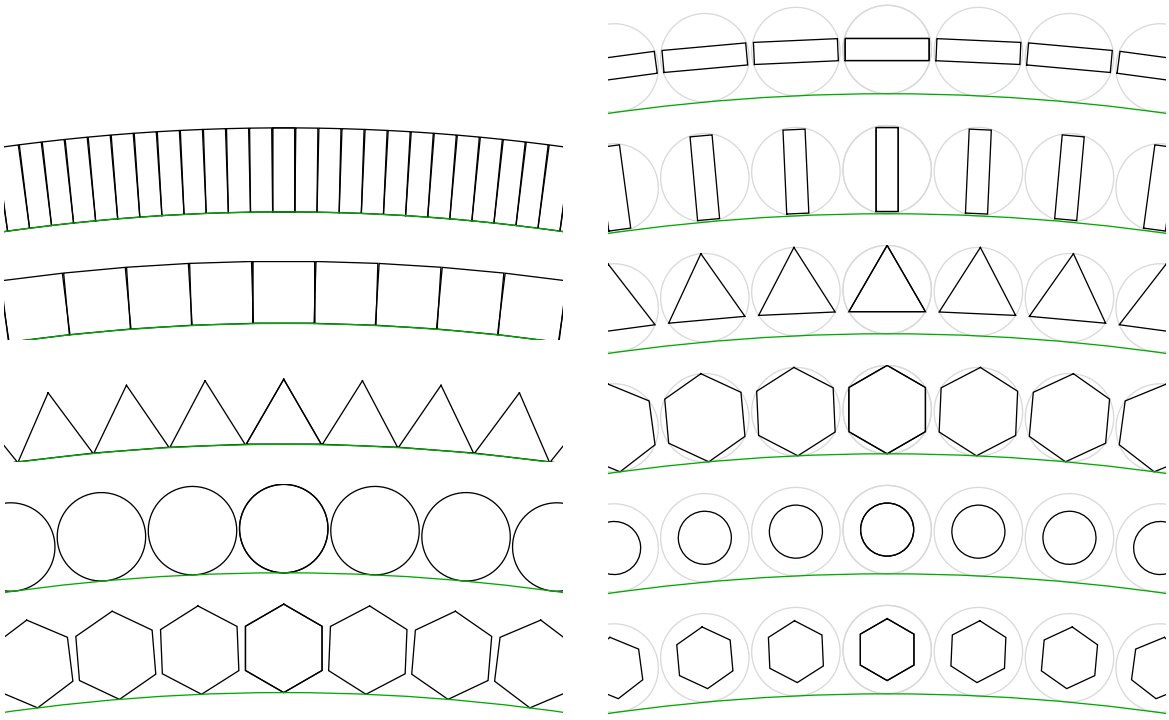
Because of that there are just a few GPU specific optimizations used in this implementation. Built-in atomics are used for updating ρ_{new}. Access to previous iteration

ρ is provided through hardware 2D texture – cached and optimized for coherent memory spatial access, usually using Z-order, Hilbert curve, depicted in Figure 5.1, or similar spatial to linear memory mappings [62, 63].

5.3. Results

5.3.1. Scintillator shapes and layout evaluation

Our 2D barrel reconstruction was used in early stages of PET project to discover scintillator shapes and detectors layouts having good theoretical resolution and sensitivity. Scintillator shapes were chosen to match the samples available in our laboratory. All of them had different cross-section area and thus sensitivity.



(a) tightly packed scintillators, variable number **(b)** 140 scintillators, inscribed in 20 mm circle

Figure 5.2.: Scintillator shapes and arrangement

We have taken into consideration two PET scanners with single ring of detectors of 45 cm radius – one with tightly packed scintillators, as shown in Figure 5.2a, second with constant number of scintillators, as shown in Figure 5.2b. Parameters presented

in Table 5.1 were used to produce the system matrix for each shape. System matrix and simulation used TOF step of 1 cm to produce TOF bins according to equation (4.5).

Shape	Parameters	No. of detectors[1]
Rectangle	5 mm (width) × 19 mm (depth)	564 / 140
Square	13.85 mm × 13.85 mm	204 / 140
Triangle	Side: 17 mm	164 / 140
Hexagon	Side: 18 mm	152 / 140
Circle	Radius: 10 mm	140 / 140

[1] tightly packed scintillators / constant number of scintillators

Table 5.1.: Scintillator shape parameters and number of detectors

For each shape we were simulating a response for Shepp-Logan phantom, until we have reached $160 * 10^6$ coincidences, and an accompanying system matrix characterized by high statistics – $200 * 10^6$ emissions for each pixel.

Our evaluation was using very optimistic TOF step of 1 cm, whereas the current J-PET prototype has TOF measurement error's standard deviation of 2.7 cm along TOR. Moreover, in real life we expect scans with $10 * 10^6$ coincidences, so presented here image reconstruction results shall be only considered in the context of scintillator shape effectiveness, but not as expected image reconstruction quality for working J-PET scanner. Such results will be presented in the next chapters.

In order to evaluate shape and layout effectiveness, we were measuring the goodness of the reconstructed image after each iteration using Normalized Mean Squared Error (NRMSE). The results of this measurement are presented in Figures 5.3 and 5.5.

In case of tightly packed scintillators, depicted in Figure 5.2a, it is not clearly visible in reconstruction images, shown in Figure 5.4, which shape has the advantage over the others. This can be however determined from the plot in Figure 5.3 and leads to the conclusion that reconstruction quality depends on the number of scintillators, which is proportional to the scanner angular resolution. There is one exception – triangular shape providing slightly better resolution than square. This can be justified by the fact that triangle is effectively thinner, so it can provide better angular resolution even with slightly smaller number of scintillators.

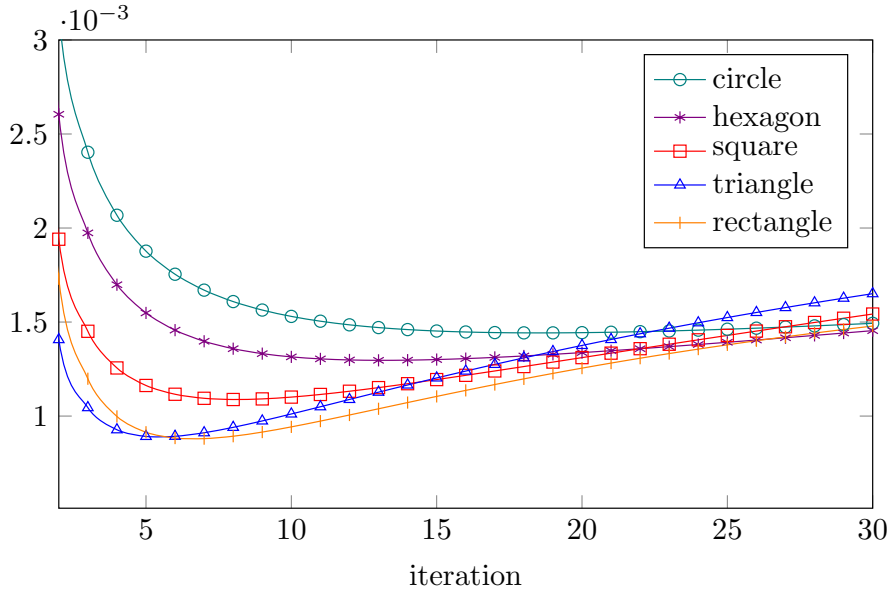


Figure 5.3.: NRMSE across 30 iterations for tightly packed shapes (140 detectors, 1 cm TOF bin size, 128 × 128 image space)

(a) Reference (emitted)

(b) Circular detectors (10 mm radius)

(c) Hexagonal detectors (10 mm side)

(d) Square detectors (14 mm ×14 mm)

(e) Triangular detectors (10 mm side)

(f) Rectangular detectors (5 mm ×19 mm)

Figure 5.4.: Reconstruction image after 10 iterations using tightly packed detector shapes (140 detectors, 1 cm TOF bin size, 128 × 128 image space)

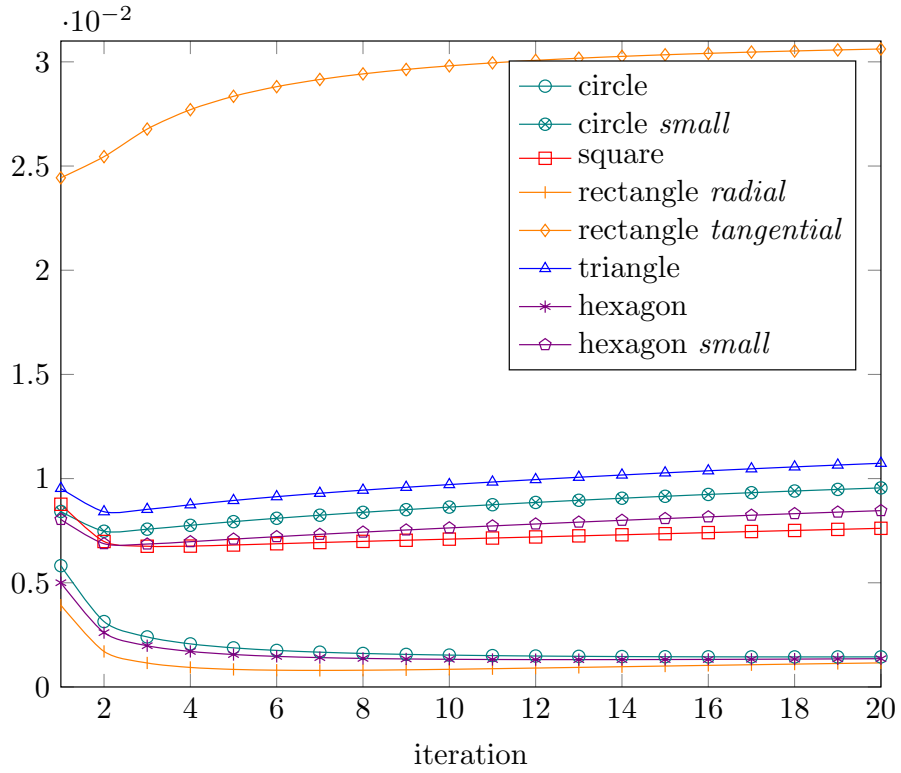
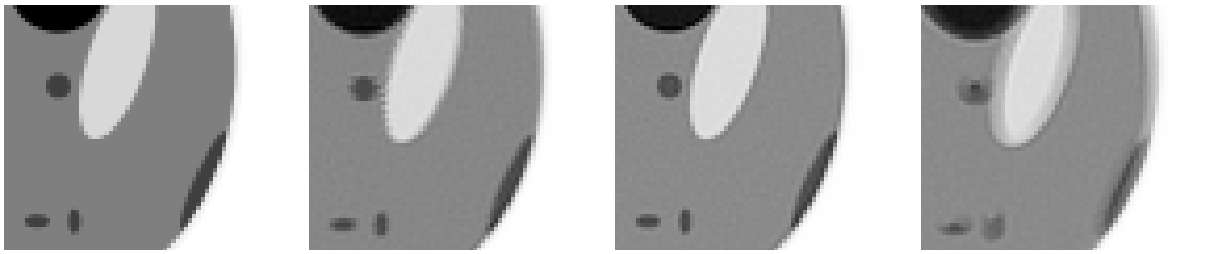


Figure 5.5.: NRMSE across 20 iterations for inscribed shapes
(140 detectors, 1 cm TOF bin size, 128 × 128 image space)



(a) Reference (emitted)

(b) Triangular detector (10 mm side)

(c) Radial rectangular detector (5 mm ×19 mm)

(d) Tangential rectangular detector (19 mm ×5 mm)

Figure 5.6.: Reconstruction image after 10 iterations using selected inscribed detector shapes
(140 detectors, 1 cm TOF bin size, 128 × 128 image space)

Another interesting fact observable in Figure 5.3 is that after few iterations we reach best quality, then the quality degrades with each next iteration. This phenomenon was already observed by others [20].

Filling the available space on the detector ring may not be possible in real-life and it is not possible in case of J-PET prototype where available space is restricted by photo-multiplier tube dimensions. Therefore, another set of measurements was taken, accounting scintillators inscribed into circle having photomultiplier dimensions (20 mm radius) which matches closer our prototype setup. This configuration is depicted in Figure 5.2b.

The evaluation presented in Figure 5.5 has let us make a decision for shape and orientation of scintillators installed in the first prototypes specified in Table 5.2 and depicted in Figures 5.7 and 5.8 – radially directed i.e. longer side along the radius, rectangular detectors of 9 mm $\times$15 mm and 7 mm $\times$19 mm cross-section dimensions were used for the first two physical prototypes of J-PET scanners.

Codename	Radius	No. of detectors	Sc. shape	Sc. dimensions
small barrel	18 cm	24 (1 layer)	Rectangle	$0.9 \times 1.5 \times 30.0$ cm
big barrel	57.5 cm	192 (3 layers)	Rectangle	$0.7 \times 1.9 \times 50.0$ cm
1st layer	42.5 cm	48	Rectangle	–
2nd layer	46.75 cm	48	Rectangle	–
3rd layer	57.5 cm	96	Rectangle	–

Table 5.2.: Configuration of first two J-PET prototypes

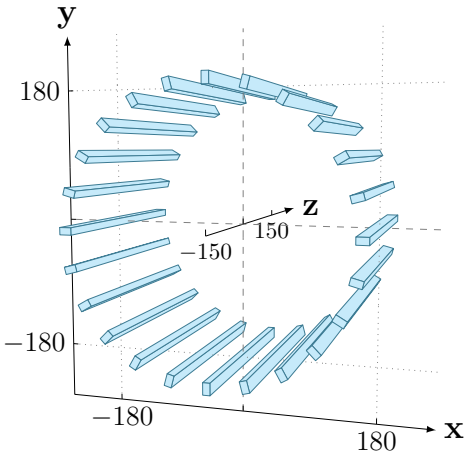


Figure 5.7.: Built J-PET "small barrel" prototype layout

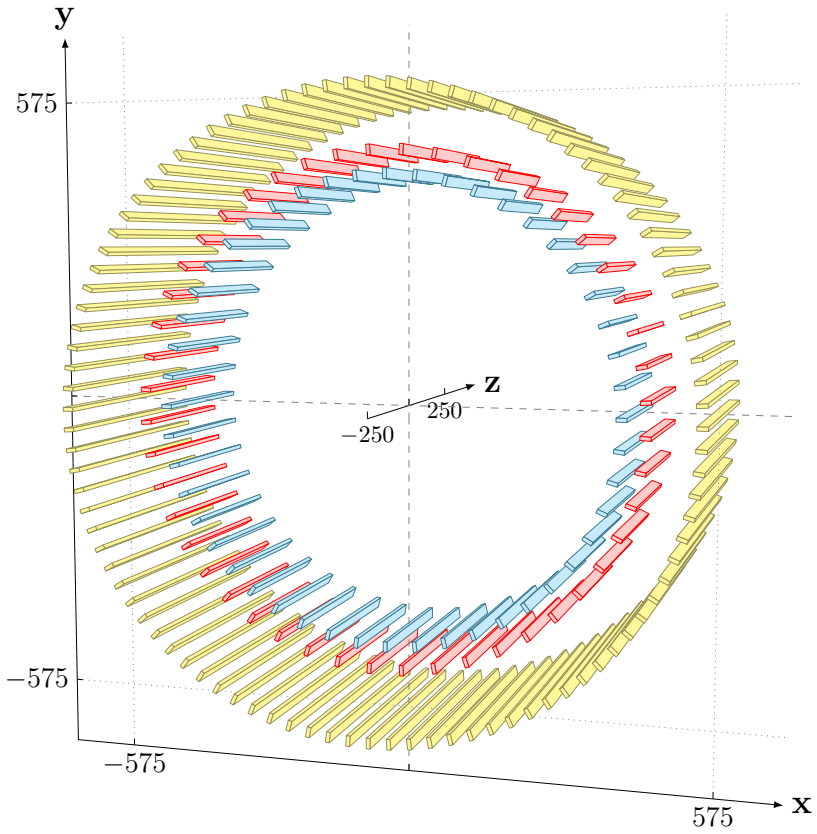


Figure 5.8.: Built J-PET "big barrel" prototype layout

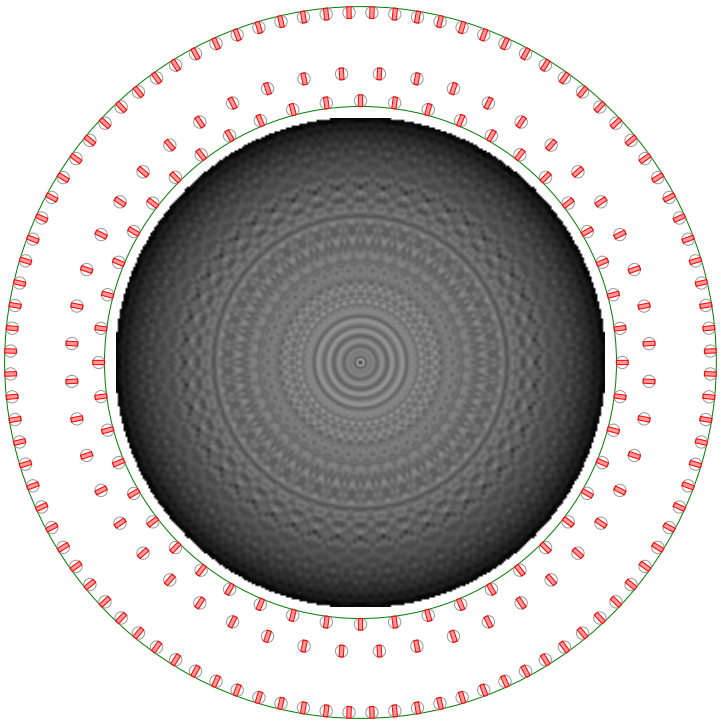


Figure 5.9.: Built J-PET "big barrel" prototype sensitivity map in $x - y$ plane

5.3.2. Performance benchmark

Bin-mode reconstruction is not computationally demanding and number of operations depends strictly on product of number of image pixels and number of TORs. As shown in Table 5.3, using GPU accelerated implementation, it is possible to execute 1000 iterations in 17 seconds for 2 mm resolution, while usually less than 20 iterations are sufficient to produce good quality image.

That is why just few optimizations used for GPU implementation and it does not bring much improvement relatively to CPU implementation. It was never an objective to produce highly optimized GPU implementation, due to non-demanding nature of bin-more reconstruction, opposite to list-mode algorithms presented in the next chapters.

Image Size	Pixel Size	CPU[1,2]	GPU[3]	Speedup
100^2	8 mm	6.19 ms	0.96 ms	6.4
200^2	4 mm	17.77 ms	3.06 ms	5.8
400^2	2 mm	74.49 ms	17.10 ms	4.4
800^2	1 mm	327.66 ms	102.49 ms	3.2

[1] *GeForce GTX 980 Ti* 1 Ghz, 6 GB, 5 632 Gflops, 336 GB/s, CUDA 7.5 `-O3`

[2] Best compiler result among GCC 4.8, GCC 5.3 and ICC 16.0

[3] *Intel Xeon E5-2699v3* 3.2 GHz, 6 core, $\approx$ 307 Gflops, 51 GB/s, GCC 5.3 `-O3`

Table 5.3.: 2D "big" barrel bin-mode reconstruction CPU vs GPU benchmark (average iteration time of 50 iterations in milliseconds)

5.3.3. Performance estimation in *Roof-line* model

2D bin-mode reconstruction has very low $Q_{\text{alg}} = 0.5$ and is clearly memory bound for all of the computing platforms shown in Table 2.1.

Operation	**FLOP**	**Mem.Op.**
pixel	-	2 loads
pixel weight	-	2 loads
ρ	-	1 texture load
denominator	2	
numerator	1	
ρ_{new}	-	1 atomic store
Total	3	6
$\boldsymbol{Q_{\text{alg}}}$	0.5	

Table 5.4.: 2D barrel reconstruction operation count per mean pixel (algorithm 10 inner loop)

Number of floating point operations in Algorithm 10 is minimal in comparison to 6 memory loads and stores as shown in Table 5.4. For such strictly memory bound algorithm only possible way for optimization is improving memory access. This is done in this case by using texture unit and its fast cache to read current ρ.

Chapter 6.

2D strip PET list-mode reconstruction

In this chapter we will focus on 2D detector geometry described by two parallel scintillators (strips). 2D strip PET reconstruction is a specific subproblem of general 3D reconstruction of the complete J-PET scanner. 2D strip PET geometry can be modeled as two parallel line segments of length L at the distance $2R$, with neglected thickness (Figure 6.1). Time information is measured by two pairs of two photomultiplier tubes attached to each end of both scintillators. This is a minimal configuration required to form a single detector and perform the reconstruction. Apart from being the elementary part of the J-PET scanner, this configuration could be used in principle as a cheap 2D scanning device on its own.

For each event the detector response $\widetilde{\mathbf{e}}$ is a tuple of four values

$$\widetilde{\mathbf{e}} = (\widetilde{T}_{ul}, \widetilde{T}_{ur}, \widetilde{T}_{dl}, \widetilde{T}_{dr}) \tag{6.1}$$

each representing time of signal appearance on each attached photomultiplier.

Gamma quanta hit positions in our geometry can be estimated by two quantities, where $c_{sci.}$ is the effective speed of light in a scintillator (12.6 cm/ns):

$$\widetilde{z_u} = \frac{1}{2} c_{sci.} \left(\widetilde{T}_{ul} - \widetilde{T}_{ur} \right), \quad \widetilde{z_d} = \frac{1}{2} c_{sci.} \left(\widetilde{T}_{dl} - \widetilde{T}_{dr} \right) \tag{6.2}$$

Combining time measurements from two scintillators, we can express the position of the gamma quanta emission point along the line segment joining upper and lower hit points as a difference $\Delta\widetilde{l}$, representing twice a distance from the segment's midpoint

$$\Delta\widetilde{l} = \frac{1}{2} c \left(\left(\widetilde{T}_{ul} + \widetilde{T}_{ur} \right) - \left(\widetilde{T}_{dl} + \widetilde{T}_{dr} \right) \right) \tag{6.3}$$

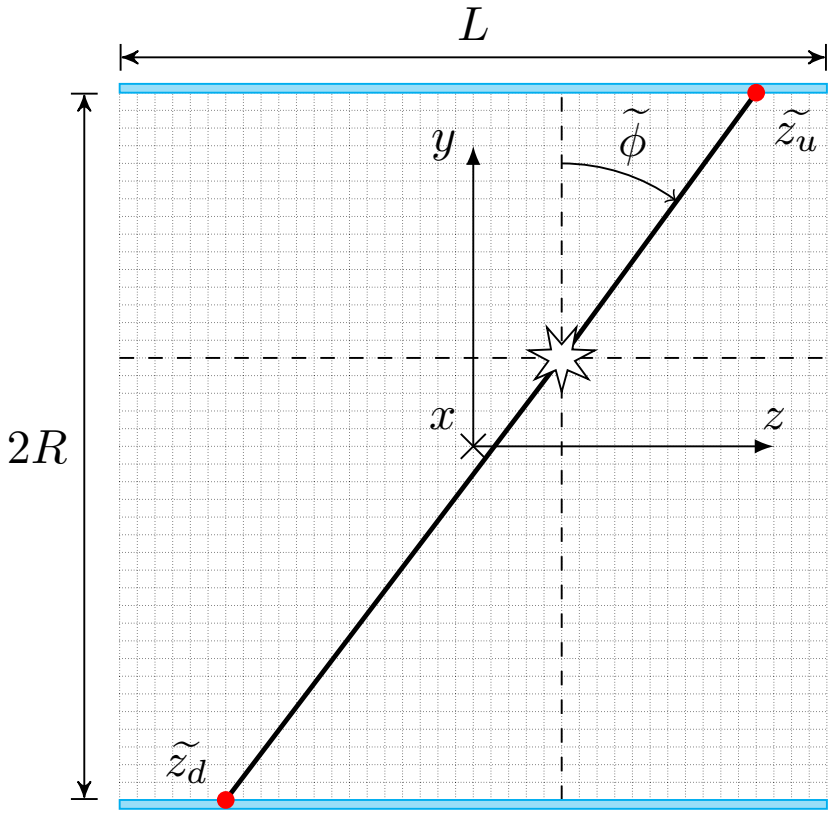


Figure 6.1.: Strip detector geometry

where c is speed of light in the air and $\Delta\widetilde{l}$ is the difference of distances of the reconstructed point (y, z) from upper and lower detection points.

From those measurements the emission position and angle can be reconstructed directly

$$\begin{aligned}
\tan\widetilde{\phi} &= \frac{\widetilde{z_u} - \widetilde{z_d}}{2R} \\
\widetilde{y} &= -\frac{1}{2}\frac{\Delta\widetilde{l}}{\sqrt{1+\tan^2\widetilde{\phi}}} = \frac{2R\Delta\widetilde{l}}{\sqrt{\widetilde{z_u} - \widetilde{z_d} + 4R^2}} \\
\widetilde{z} &= \frac{1}{2}\left(\widetilde{z_u} + \widetilde{z_d} + 2y\tan\widetilde{\phi}\right) \\
&= \frac{1}{2}\left(\widetilde{z_u} + \widetilde{z_d} + \frac{(\widetilde{z_u} - \widetilde{z_d})\Delta\widetilde{l}}{\sqrt{\widetilde{z_u} - \widetilde{z_d} + 4R^2}}\right)
\end{aligned} \tag{6.4}$$

where $2R$ denotes distance between the scintillators. In this calculation attenuation model (3.23) for scintillator thickness and geometry errors are neglected. We later show that this does not noticeably degrade the reconstruction quality.

The calculation above is however subject to measurement errors

$$\widetilde{z}_y = z_y + \varepsilon_{z_y},\ y = u, d \qquad \widetilde{\Delta l} = \Delta l + \varepsilon_{\Delta l} \tag{6.5}$$

where the errors ε are normally distributed with correlation matrix C. The magnitude of the measurement errors depends on place where the gamma hit the scintillator $C = C(z_u, z_d)$

$$C = \begin{pmatrix} \sigma_z^2(z_u) & 0 & \varsigma(z_u) \\ 0 & \sigma_z^2(z_d) & -\varsigma(z_d) \\ \varsigma(z_u) & -\varsigma(z_d) & \sigma_{\Delta l}^2(z_u, z_d) \end{pmatrix} \tag{6.6}$$

where

$$\begin{aligned} \sigma_z^2(z) &= E\left[\varepsilon_{u,d}^2(z)\right], \\ \sigma_{\Delta l}^2(z_u, z_d) &= E\left[\varepsilon_{\Delta l}^2(z_u, z_d)\right], \\ \varsigma(z) &= E[\varepsilon_{z_u}(z)\varepsilon_{\Delta l}(z, z_d)] = -E[\varepsilon_{z_d}(z)\varepsilon_{\Delta l}(z_u, z)] \end{aligned} \tag{6.7}$$

In our current implementation we assume $\varsigma(z) = 0$, as the preliminary measurements of early two strip prototype show that the correlation between two detector strip errors is minimal, effectively making C to be diagonal. Also, the dependence of errors on z turns out to be weak, so we assume that the magnitude of errors is constant across the whole strip.

The 2D strip PET reconstruction is done iteratively using the list-mode version of the MLEM algorithm as described in Section 3.4.2. Each iteration of this algorithm is defined by 2D version of equation (3.18)

$$\rho(i)^{(t+1)} = \sum_{\widetilde{\mathbf{e}} \in \widetilde{\mathbf{E}}} \frac{P(\widetilde{\mathbf{e}} \mid i)\, \rho(i)^{(t)}}{\sum\limits_{j \in \mathcal{V}} P(\widetilde{\mathbf{e}} \mid j)\, s(j)\, \rho(j)^{(t)}} \quad \text{for } i \in \mathcal{I} \tag{6.8}$$

where $\rho(i)$ denotes sought emission density, $P(\widetilde{\mathbf{e}} \mid i)$ denotes *reconstruction kernel* that represents the probability that an event detected as $\widetilde{\mathbf{e}}$ originates from pixel i and $s(i)$ denotes *sensitivity* of the pixel i.

6.1. Analytic kernel approximation for strip PET scanner

We will now derive analytical approximation $P(\widetilde{\mathbf{e}} \mid i)$ presented first in [64, 65]. First, we start with $s(\mathbf{e})$ – a probability that event $\mathbf{e}$ will be detected. We assume that every event reaching the detector is detected so the $s(\mathbf{e})$ is given solely by the geometrical constraints

$$s(\mathbf{e}) = s(z_u, z_d, \Delta l) = \begin{cases} 1 & z_u \in [-\frac{L}{2}, \frac{L}{2}] \wedge z_d \in [-\frac{L}{2}, \frac{L}{2}] \\ 0 & \text{otherwise} \end{cases} \tag{6.9}$$

This is somewhat more complicated in the image space

$$s(y, z, \phi) = \begin{cases} 1 & \tan\phi \in \left[\max\left(-\frac{\frac{1}{2}L + z}{R - y}, \frac{-\frac{1}{2}L + z}{R + y} \right), \min\left(\frac{\frac{1}{2}L - z}{R - y}, \frac{\frac{1}{2}L + z}{R + y} \right) \right] \\ 0 & \text{otherwise} \end{cases} \tag{6.10}$$

With that we can determine the sensitivity for image point (y, z)

$$s(y, z) = \pi^{-1} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \mathrm{d}\phi \; s(y, z, \phi) = \pi^{-1}(\phi_{max} - \phi_{\min}) \tag{6.11}$$

where

$$\begin{aligned} \phi_{min} &= \arctan \max\left(-\frac{\frac{1}{2}L + z}{R - y}, \frac{-\frac{1}{2}L + z}{R + y} \right), \\ \phi_{max} &= \arctan \min\left(\frac{\frac{1}{2}L - z}{R - y}, \frac{\frac{1}{2}L + z}{R + y} \right) \end{aligned} \tag{6.12}$$

As discussed in the previous section the errors are normally distributed

$$P(\widetilde{\mathbf{e}} \cap \mathbf{e}) = s(\mathbf{e}) \frac{\det^{\frac{1}{2}} C(\mathbf{e})}{(2\pi)^{\frac{3}{2}}} \exp\left(-\frac{1}{2} (\widetilde{\mathbf{e}} - \mathbf{e})^T C^{-1}(\mathbf{e}) (\widetilde{\mathbf{e}} - \mathbf{e}) \right) \tag{6.13}$$

where

$$\Delta\mathbf{e} = \mathbf{e}(z,y,\phi) - \mathbf{e}(\widetilde{z},\widetilde{y},\widetilde{\phi}) = \begin{pmatrix} z + (R-y)\tan\phi - \widetilde{z} - (R-\widetilde{y})\tan\widetilde{\phi} \\ z - (R+y)\tan\phi - \widetilde{z} + (R+\widetilde{y})\tan\widetilde{\phi} \\ -2y\sqrt{1+\tan^2\phi} + 2\widetilde{y}\sqrt{1+\tan^2\phi} \end{pmatrix} \tag{6.14}$$

With these definitions sought $P(\widetilde{\mathbf{e}} \,|\, i)$ becomes

$$P(\widetilde{\mathbf{e}} \,|\, i) = \frac{P(\widetilde{\mathbf{e}} \cap i)}{s(i)} \tag{6.15}$$

where

$$P(\widetilde{\mathbf{e}} \cap i) = \pi^{-1} \int\limits_{z,y\in i} \int\limits_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \mathrm{d}\phi \, P(\widetilde{\mathbf{e}} \cap (z,y,\phi)), \tag{6.16}$$

$$s(i) = \int\limits_{z,y\in i} s(z,y) \tag{6.17}$$

We will now construct an approximation for the formula (6.16), starting by calculating

$$P(\widetilde{\mathbf{e}} \cap z,y) = \pi^{-1} \int\limits_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \mathrm{d}\phi \, P(\widetilde{\mathbf{e}} \cap (z,y,\phi)) \tag{6.18}$$

The first approximation we make is assuming that the correlation matrix C is depending weakly on $\mathbf{e}$ and can be approximated by its value at $\widetilde{\mathbf{e}}$. The integral (6.18) becomes then

$$P(\widetilde{\mathbf{e}} \cap z,y) = \pi^{-1} \frac{\det^{\frac{1}{2}} C(\widetilde{\mathbf{e}})}{(2\pi)^{\frac{3}{2}}} \int\limits_{\phi_{\min}}^{\phi_{\max}} \mathrm{d}\phi \, \exp\left(-\frac{1}{2}(\widetilde{\mathbf{e}}-\mathbf{e})^T C^{-1}(\widetilde{\mathbf{e}})(\widetilde{\mathbf{e}}-\mathbf{e})\right) \tag{6.19}$$

Assuming $3\sigma_z \ll L$ and that we will be evaluating our kernel only for points (y,z) lying inside detector and $\widetilde{\mathbf{e}}$ whose $\phi_{\min} \ll \phi_{\widetilde{\mathbf{e}}} \ll \phi_{\max}$ we can replace integral limits

$$P(\widetilde{\mathbf{e}} \cap z,y) \approx \pi^{-1} \frac{\det^{\frac{1}{2}} C(\widetilde{\mathbf{e}})}{(2\pi)^{\frac{3}{2}}} \int\limits_{-\infty}^{\infty} \mathrm{d}\phi \, \exp\left(-\frac{1}{2}(\widetilde{\mathbf{e}}-\mathbf{e})^T C^{-1}(\widetilde{\mathbf{e}})(\widetilde{\mathbf{e}}-\mathbf{e})\right) \tag{6.20}$$

When $\phi_{\tilde{\mathbf{e}}}$ of some $\tilde{\mathbf{e}}$ is close to $\phi_{\min}$ or $\phi_{\max}$, the integral (6.20) with replaced limits is producing too high value, because the essential part of the Gaussian volume crosses $\phi_{\min}$ or $\phi_{\max}$ limit, yet is taken into integral value. However, when $\phi_{\min} \ll \phi_{\tilde{\mathbf{e}}} \ll \phi_{\max}$, the most part of the Gaussian volume lies inside the limits, hence the approximation produces correct value in such case.

The effect is depicted in Figure 6.2. We will later show that the fraction of responses with $\phi_{\tilde{\mathbf{e}}}$ close to the limits is relatively small and does not have high impact on overall approximation error.

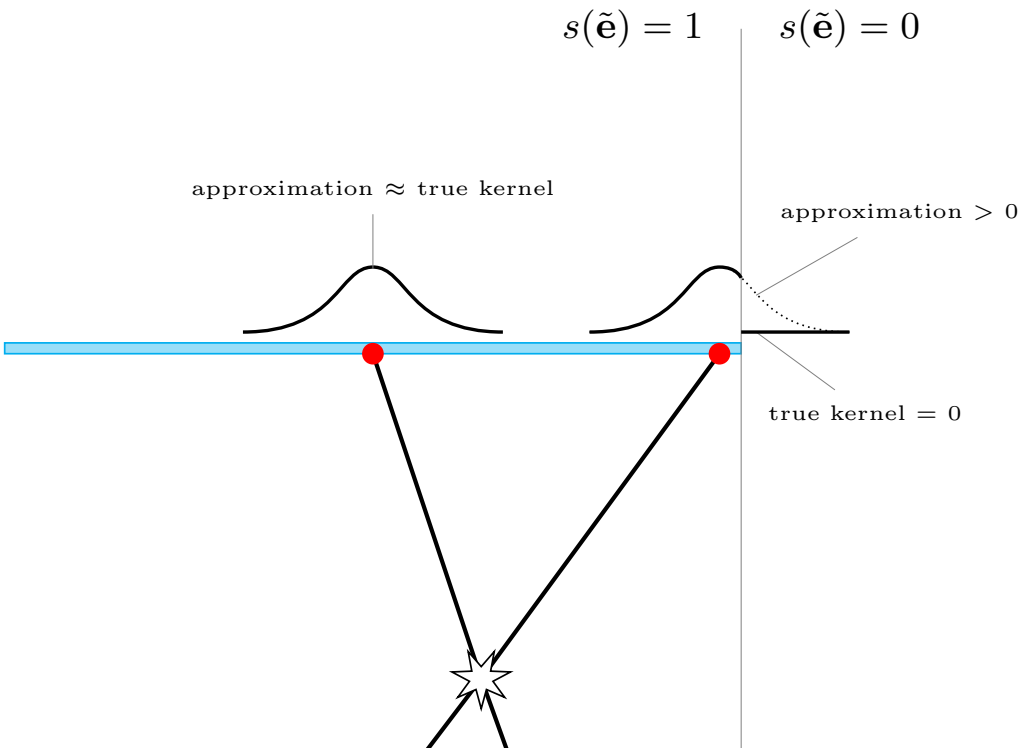


Figure 6.2.: Strip detector approximation vs true kernel relative to angle

We will now approximate integral (6.20) using the saddle-point approximation. Therefore we first expand $\Delta\mathbf{e}$ in ϕ

$$\Delta\mathbf{e} \approx \vec{o}\Delta\phi^2 + \vec{a}\Delta\phi + \vec{b} \quad \text{for} \quad \Delta\phi \stackrel{\text{def}}{=} \phi - \widetilde{\phi} \tag{6.21}$$

with

$$\vec{o} = \begin{pmatrix} -(\Delta y + \widetilde{y} - R)\tan\widetilde{\phi}\cos^{-2}\widetilde{\phi} \\ -(\Delta y + \widetilde{y} + R)\tan\widetilde{\phi}\cos^{-2}\widetilde{\phi} \\ -(\Delta y + \widetilde{y})\cos^{-1}\widetilde{\phi}(1 + 2\tan^2\widetilde{\phi}) \end{pmatrix}, \tag{6.22}$$

$$\vec{a} = \begin{pmatrix} -(\Delta y + \widetilde{y} - R)\cos^{-2}\widetilde{\phi} \\ -(\Delta y + \widetilde{y} + R)\cos^{-2}\widetilde{\phi} \\ -2(\Delta y + \widetilde{y})\cos^{-1}\widetilde{\phi}\tan\widetilde{\phi} \end{pmatrix}, \tag{6.23}$$

$$\vec{b} = \begin{pmatrix} \Delta z - \Delta y\tan\widetilde{\phi} \\ \Delta z - \Delta y\tan\widetilde{\phi} \\ -2\Delta y\cos^{-1}\widetilde{\phi} \end{pmatrix} \tag{6.24}$$

where

$$\Delta y \stackrel{\text{def}}{=} y - \widetilde{y} \quad \text{and} \quad \Delta z \stackrel{\text{def}}{=} z - \widetilde{z} \tag{6.25}$$

After inserting (6.21) into the exponent of (6.19) we obtain the expression

$$\frac{1}{2}\left(\vec{o}\Delta\phi^2 + \vec{a}\Delta\phi + \vec{b}\right) C^{-1} \left(\vec{o}\Delta\phi^2 + \vec{a}\Delta\phi + \vec{b}\right) \tag{6.26}$$

which we truncate to the quadratic order

$$\left(\vec{o}C^{-1}\vec{b} + \frac{1}{2}\vec{a}C^{-1}\vec{a}\right)\Delta\phi^2 + \vec{a}C^{-1}\vec{b}\Delta\phi + \frac{1}{2}\vec{b}C^{-1}\vec{b} \tag{6.27}$$

After differentiating with respect to $\Delta\phi$ we obtain the equation for the minimum

$$\left(2\vec{o}C^{-1}\vec{b} + \vec{a}C^{-1}\vec{a}\right)\Delta\phi + \vec{a}C^{-1}\vec{b} = 0 \tag{6.28}$$

with the solution

$$\Delta\phi_{\text{min}} = -\frac{\vec{b}C^{-1}\vec{a}}{\vec{a}C^{-1}\vec{a} + 2\vec{o}C^{-1}\vec{b}} \tag{6.29}$$

Denoting $\lambda \stackrel{\text{def}}{=} \Delta\phi - \Delta\phi_{\min}$ we rewrite the (6.27) as

$$\frac{1}{2}\left(\vec{a}C^{1}\vec{a} + 2\vec{o}C^{-1}\vec{b}\right)\lambda^2 + \frac{1}{2}\left(\vec{b}C^{-1}\vec{b} - \frac{(\vec{a}C^{-1}\vec{b})^2}{\vec{a}C^{1}\vec{a} + 2\vec{o}C^{-1}\vec{b}}\right) \tag{6.30}$$

Finally, we obtain

$$\begin{aligned} P(\widetilde{\mathbf{e}} \cap z, y) \approx \frac{\det^{\frac{1}{2}} C(\widetilde{\mathbf{e}})}{(2\pi)^{\frac{3}{2}}} \exp\left(-\frac{1}{2}\left(\vec{b}C^{-1}\vec{b} - \frac{\left(\vec{b}C^{-1}\vec{a}\right)^2}{\vec{a}C^{-1}\vec{a} + 2\vec{o}C^{-1}\vec{b}}\right)\right) \\ \pi^{-1}\int_{-\infty}^{\infty} \mathrm{d}\lambda \, \exp\left(-\frac{1}{2}\lambda^2\left(\vec{a}C^{-1}\vec{a} + 2\vec{o}C^{-1}\vec{b}\right)\right) \end{aligned} \tag{6.31}$$

and performing the Gaussian integration we get

$$\begin{aligned} P(\widetilde{\mathbf{e}} \cap z, y) \approx \frac{\det^{\frac{1}{2}} C}{2\pi\sqrt{\vec{a}C^{-1}\vec{a} + 2\vec{o}C^{-1}\vec{b}}}\pi^{-1} \\ \exp\left(-\frac{1}{2}\left(\vec{b}C^{-1}\vec{b} - \frac{\left(\vec{b}C^{-1}\vec{a}\right)^2}{\vec{a}C^{-1}\vec{a} + 2\vec{o}C^{-1}\vec{b}}\right)\right) \end{aligned} \tag{6.32}$$

We still need to perform the integration over the pixel in (6.16) and (6.17). Instead of that we will just approximate the integrals by the value of (6.32) and (6.11) at the center point (y_i, z_i) of the pixel i and its volume $V(i)$

$$P(\widetilde{\mathbf{e}} \cap i) \approx V(i)\, P(\widetilde{\mathbf{e}} \cap y_i, z_i)\,, \quad s(i) \approx V(i)\, s(y_i, z_i) \tag{6.33}$$

and

$$P(\widetilde{\mathbf{e}} \,|\, i) \approx \frac{P(\widetilde{\mathbf{e}} \cap y_i, z_i)}{s(y_i, z_i)} \tag{6.34}$$

For each event, we will limit calculating formula (6.32) to three sigma region around the reconstruction point, estimated by the quadratic form ellipse equation

$$\vec{b}C^{-1}\vec{b} = R^2 \tag{6.35}$$

where R is defined as the three sigma ($R = 3$).

6.2. Consideration for geometry errors

Analytic approximation kernel given by equation (6.32) does not consider scintillator hit position geometry error (Figure 3.1), only measurement errors are taken into account. We will show that for current J-PET setup, measurement errors play vital role and geometry errors can be ignored.

Lets assume for the moment that all gamma quanta emissions are perpendicular to 19 mm thick scintillator used in J-PET big barrel as shown in Table 5.2. We will estimate geometry error $\sigma_{\text{geom}\Delta l}$ relatively to measurement error $\sigma_{\Delta l} = 40mm$ that represents J-PET prototype time resolution [14]. Knowing that coincidence occurred and both gamma quanta have hit respective scintillators, conditional probability density of an interaction can be modeled after equation (3.22) and equation (3.23) as

$$\begin{aligned} f_{\text{hit}}(x) &= \begin{cases} \dfrac{\frac{1}{100}e^{-\frac{x}{100}}}{1-e^{-\frac{19}{100}}} & \text{for } 0 \leq x \leq 19 \\ 0 & \text{otherwise} \end{cases} \\ &\approx \begin{cases} 0.058\, e^{-\frac{x}{100}} & \text{for } 0 \leq x \leq 19 \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{6.36}$$

that is almost flat in $[0, 19]$ and can be approximated by uniform distribution

$$\widetilde{f_{\text{hit}}}(x) = \begin{cases} \frac{1}{19} & \text{for } 0 \leq x \leq 19 \\ 0 & \text{otherwise} \end{cases} \tag{6.37}$$

If each of two coincidence hit geometric errors $\varepsilon_{\text{hit}_{u,d}}$ have an uniform distribution over the interval $[-\frac{19}{2}\,\text{mm}, \frac{19}{2}\,\text{mm}]$, then their sum

$$\varepsilon_{\text{geom}\Delta l} = \varepsilon_{\text{hit}_u} + \varepsilon_{\text{hit}_d} \tag{6.38}$$

have a symmetric triangular distribution over the interval $[-19\,\text{mm}, 19\,\text{mm}]$ and $\sigma_{\text{geom}\Delta l} \approx 7.8\,\text{mm}$ and the combined $\sigma_{\Delta l+\text{geom}\Delta l}$ is only 2% higher than $\sigma_{\Delta l}$

$$\begin{aligned} \sigma_{\Delta l+\text{geom}\Delta l} &= \sqrt{\sigma_{\Delta l}^2 + \sigma_{\text{geom}\Delta l}^2} \\ &\approx \sqrt{40^2 + 7.8^2} \\ &\approx 40.8\,\text{mm} \end{aligned} \tag{6.39}$$

Relaxing assumption that gamma quanta emissions are perpendicular, secant length expressed relatively to angle ϕ and scintillator thickness l as

$$\text{secant}(\phi, l) = \frac{l}{\cos\phi} \tag{6.40}$$

does not vary much depending on emission angle for current J-PET setup. For big barrel J-PET prototype using 50 cm long scintillators and $45cm$ radius maximum angle is arctan $\frac{50}{90} \approx 30^\circ$ giving the maximum secant length only 15% greater than scintillator thickness. This may only marginally raise $\sigma_{\text{geom}\Delta l}$ and combined $\sigma_{\Delta l + \text{geom}\Delta l}$ in equation (6.39).

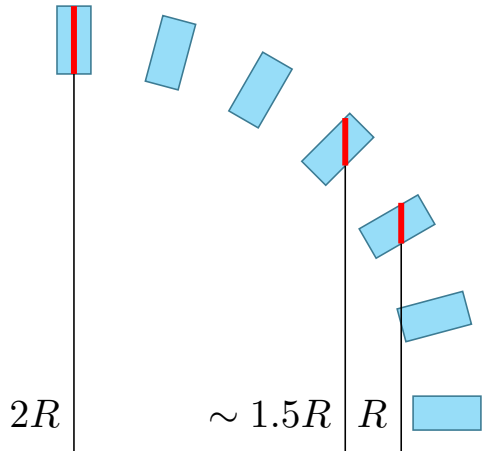
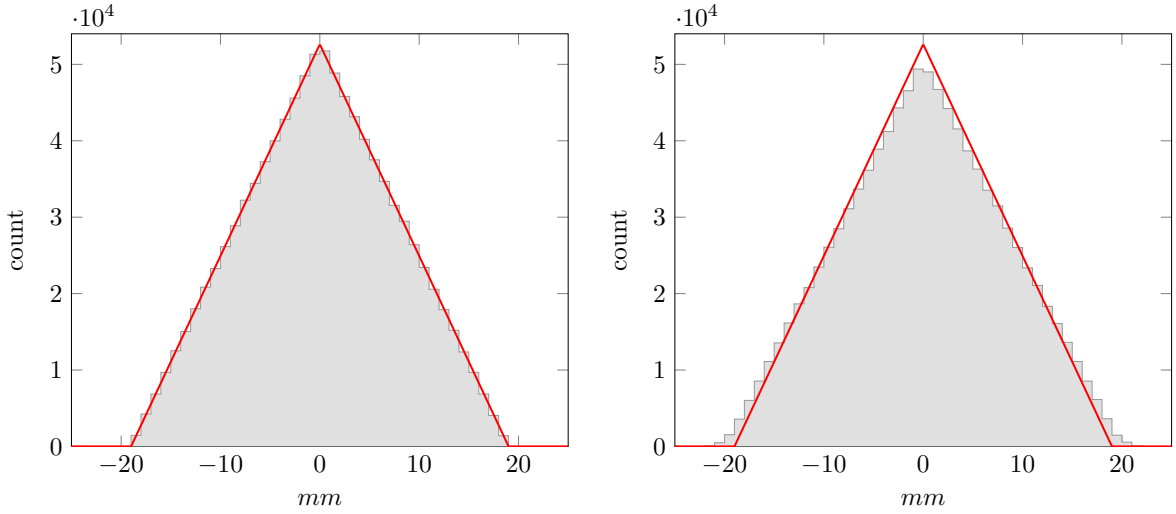


Figure 6.3.: Secant length relative to TOR distance to center

The calculation above is done for central TORs having $2R$ distance between scintillators. TORs lying far from center have maximum secant length higher, e.g. TOR lying at FOV border with length R, depicted in Figure 6.3, have arctan $\frac{50}{45} \approx 50^\circ$ and maximum secant length 55% greater than secant length for perpendicular ϕ. However, at the same time the secant length for perpendicular ϕ and length R – right red line segment in Figure 6.3, is half of the one for $2R$, so geometric errors are even lower for TORs lying on the FOV border, showing altogether that geometry errors may be ignored for current J-PET setup.

6.2.1. Simulation of geometry errors

In order to validate calculations above, we have performed simulations of 2 strip PET detector system with $R = 415mm$, $L = 500mm$ and scintillator height of $19mm$, representing one "big" barrel detector pair. 1 million responses for emissions from central $(0, 0)$ point source were collected for each simulation. First simulation had fixed emission direction ϕ angle, perpendicular to detectors, representing case described by equation (6.36). The resulting $\Delta\widetilde{l}$ distribution, depicted in Figure 6.4a, was almost perfect triangular distribution with parameters $(-19mm, 19mm)$ (marked with red line for reference) and $\sigma_{\text{geom}\Delta l} = 7.75mm$. This has confirmed our calculations.



(a) Fixed perpendicular ϕ **(b)** Isotropic ϕ

Figure 6.4.: $\Delta\widetilde{l}$ histogram $R = 415mm$, $L = 500mm$, bin size = 1 mm
grey area – simulated distribution,
red outline – $(-19\,\text{mm}, 19\,\text{mm})$ triangular distribution

Second simulation was conducted with relaxed constraint about emission angle, so ϕ angle was a random direction. The resulting $\Delta\widetilde{l}$ distribution, depicted in Figure 6.4b, was nearly triangular distribution with $\sigma_{\text{geom}\Delta l} = 8.16\,\text{mm}$. This has confirmed that the emission angle has negligible impact on $\Delta\widetilde{l}$ geometric error in "big" barrel configuration case and altogether the $\Delta\widetilde{l}$ geometric errors have minimal impact on overall $\Delta\widetilde{l}$ error distribution.

Using the second simulation, we were also able to estimate $\widetilde{z_u}$, $\widetilde{z_d}$ geometric errors, for which analytic derivation is not so trivial. This was done indirectly by calculating

distribution for $\widetilde{z_u} + \widetilde{z_d}$. Since the emission point $(0, 0)$ is central, therefore $z_u + z_d = 0$ and hence $\widetilde{z_u} + \widetilde{z_d} = \varepsilon_u + \varepsilon_d$.

The result $\widetilde{z_u} + \widetilde{z_d}$ distribution depicted in Figure 6.5 had $\sigma_{\text{geom}\, z_u + z_d} = 2.62\,\text{mm}$, so assuming $\sigma_{z_{u,d}} = 10\,\text{mm}$ for "big" barrel configuration [14], we get

$$\sigma_{\text{geom}\, z_{u,d}} = \frac{\sigma_{\text{geom}\, z_u + z_d}}{\sqrt{2}} = 1.85\,\text{mm}, \tag{6.41}$$

$$\begin{aligned} \sigma_{z_{u,d}+\text{geom}\, z_{u,d}} &= \sqrt{\sigma^2_{z_{u,d}} + \sigma^2_{\text{geom}\, z_{u,d}}} \\ &\approx \sqrt{10^2 + 1.85^2} \\ &\approx 10.17\,\text{mm} \end{aligned} \tag{6.42}$$

which shows that the combined $\sigma_{z_{u,d}+\text{geom}\, z_{u,d}}$ is 1.7% higher than $\sigma_{z_{u,d}}$.

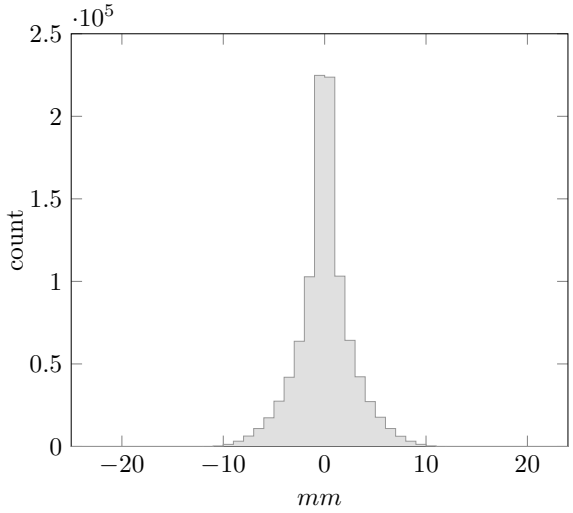


Figure 6.5.: $\widetilde{z_u} + \widetilde{z_d}$ histogram $R = 415\,\text{mm}$, $L = 500\,\text{mm}$, bin size $= 1\,\text{mm}$ (isotropic ϕ, grey area – simulated distribution)

6.3. Analytic kernel formula validation

In order to validate our analytic kernel formula (6.32), that uses several approximations in its derivation, we have performed three Monte-Carlo simulations of 2D strip detector response to a point source at three (z, y) positions – $(0, 0)$, $(10\,\text{cm}, 0)$, $(10\,\text{cm}, 10\,\text{cm})$. For each position 10^9 coincidences were simulated, producing distri-

bution of $P(\widetilde{\mathbf{e}} \cap z, y)$ for each $\widetilde{\mathbf{e}} = (\widetilde{z}, \widetilde{y}, \widetilde{\phi})$, where $\widetilde{z}$ and $\widetilde{y}$ were binned as 50×50 pixel grid, and $\tan \widetilde{\phi}$ was stored in 200 bins.

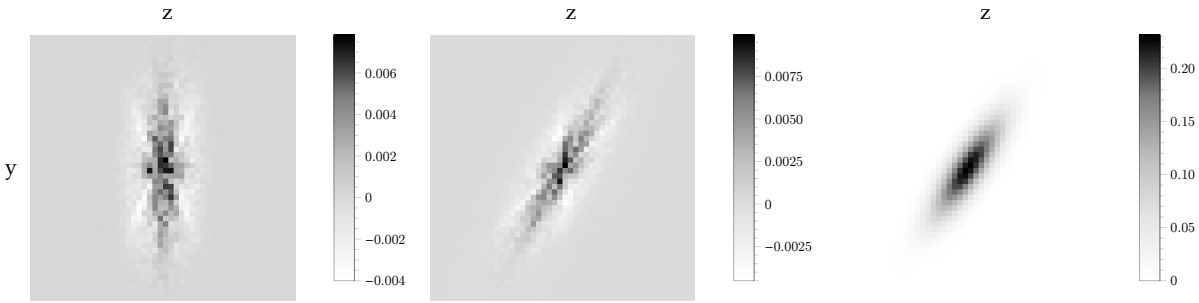
Next, we have calculated error as a normalized difference between analytic kernel value $P(\widetilde{\mathbf{e}} \cap z, y)$ multiplied by Jacobian J – coming from replacing $\widetilde{\mathbf{e}} = (\widetilde{z_u}, \widetilde{z_d}, \Delta\widetilde{l})$ with $\widetilde{\mathbf{e}} = (\widetilde{z}, \widetilde{y}, \tan\widetilde{\phi})$ according to (6.4) and integrating over pixel and $\tan\phi$ bins, and simulated bin value $\mathrm{sim}(\widetilde{\mathbf{e}}, z, y)$

$$\mathrm{error}(\widetilde{\mathbf{e}}, z, y) = \frac{J\,P(\widetilde{\mathbf{e}} \cap z, y) - \mathrm{sim}(\widetilde{\mathbf{e}}, z, y)}{J\,P(\widetilde{\mathbf{e}} \cap z, y)} \tag{6.43}$$

where

$$J = 4R\ \mathrm{pixel}_{\mathrm{size}}^2\ \mathrm{tan}_{\mathrm{size}}\ \sqrt{1 + \tan^2} \tag{6.44}$$

Our formula agreed with the simulation with a maximum error below 1% for $\phi_{\min} \ll \phi_{\widetilde{\mathbf{e}}} \ll \phi_{\max}$. For $\phi_{\widetilde{\mathbf{e}}}$ angles close to $\phi_{\min}$ and $\phi_{\max}$ our formula was producing to high values, shown in Figures 6.6c, 6.7c and manifested by black caps near the top and bottom border in Figure 6.8. This is expected because of (6.20) approximation. Nevertheless, there were just few tangent bins for angles close $\phi_{\min}$ and $\phi_{\max}$, so the overall error remained low as shown in Figures 6.6a, 6.6b, 6.7a and Figure 6.8a.

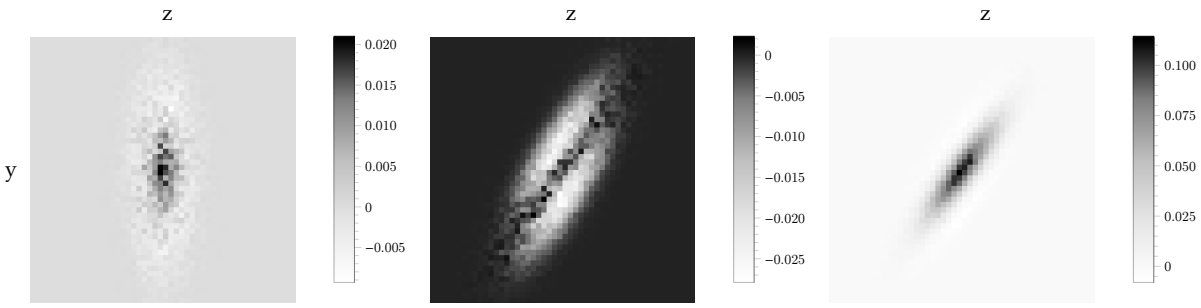


(a) $\tan\phi = 0$ **(b)** $\tan\phi \prec \tan\phi_{\max}$ **(c)** $\tan\phi = \tan\phi_{\max}$

Figure 6.6.: Relative error of analytic kernel to 2D simulated kernel in $(0, 0)$ for selected $\tan\phi$ (positive value – analytic value too high, negative – too low)

Next we conducted simulation of 3D two strip detector taking into account geometry and scintillator interaction model. The agreement for selected tangent bins is shown in Figure 6.7 and its changes along ϕ tangent are shown in Figure 6.8b.

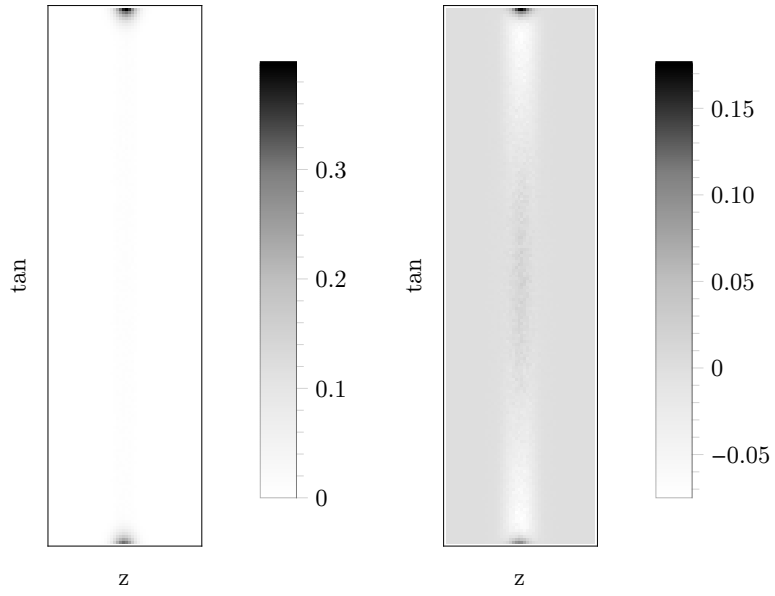
It is visible in Figure 6.8b that the agreement for $\phi_{\min} \ll \phi_{\widetilde{\mathbf{e}}} \ll \phi_{\max}$ in case of 3D is not so perfect as in case of 2D shown in Figure 6.8a. Particularly the analytic kernel is



(a) $\tan\phi = 0$ **(b)** $\tan\phi \prec \tan\phi_{\max}$ **(c)** $\tan\phi = \tan\phi_{\max}$

Figure 6.7.: Relative error of analytic kernel to 3D simulated kernel in $(0,0)$ for selected $\tan\phi$ (positive value – analytic value too high, negative – too low)

producing too small values for greater angles, but not very close to $\phi_{\min}$ or $\phi_{\max}$, as shown in Figure 6.7b. This comes straight from interaction model dependent on equation (6.40), which is not taken into account in our analytic kernel approximation. Still, the overall error in case of 3D is below 3% for $\phi_{\min} \ll \phi_{\tilde{\mathbf{e}}} \ll \phi_{\max}$.



(a) 2D simulation **(b)** full 3D simulation

Figure 6.8.: Relative error of analytic kernel to simulated kernel in $(0,0)$ along $\tan\phi$ (positive value – analytic value too high, negative – too low)

6.4. Generic implementation

The iteration step described by formula (6.8) can be implemented as described below

Algorithm 11 2D strip list-mode reconstruction step

function STRIPLMRECONSTRUCTIONSTEP(ρ, ρ_{new})
  $\rho_{\text{new}} \leftarrow 0$ ▷ start with zero ρ_{new}
  for all $\widetilde{\mathbf{e}} \in \widetilde{\mathbf{E}}$ **do**
    denom $\leftarrow 0$ ▷ calculate denominator
    for $i \in \texttt{ellipse}(\widetilde{\mathbf{e}})$ **do**
      $\text{cache}(i) \leftarrow P(\widetilde{\mathbf{e}} \mid i)\, \rho(i)$ ▷ cache product of kernel and image
      $\text{denom} \leftarrow \text{denom} + \text{cache}(i)\, s(i)$
    end for
    for $i \in \texttt{ellipse}(\widetilde{\mathbf{e}})$ **do**
      $\rho_{\text{new}}(i) \leftarrow \rho_{\text{new}}(i) + \frac{\text{cache}(i)}{\text{denom}}$
    end for
  end for
end function

Loops **for all** $\widetilde{\mathbf{e}} \in \texttt{ellipse}(\widetilde{\mathbf{e}})$ iterate over all pixels in the 3σ ellipse of the event $\widetilde{\mathbf{e}}$. To calculate pixels contributing to this ellipse, we first need to determine its bounding box in pixel space. Once bounding box is calculated, we loop only through pixels inside this bounding box. Each pixel is then tested if its center point resides inside or outside of the ellipse. Only then the whole kernel is calculated. The results are cached and used subsequently in the second loop.

The CPU implementation follows essentially the Algorithm 11. We use OpenMP to parallelize the outer loop over the events. Each thread writes to its own copy of ρ_{new} array. All copies are merged at the end of the iteration. Currently we do not take direct advantage of the AVX/SSE instruction set aside of automatic vectorization provided by *Intel C++ Compiler* – vectorizing for example transcendental functions.

6.5. GPU accelerated implementation

Construction of GPU accelerated implementation for J-PET 2D strip detector, originally presented in [66], begins with a naive GPU implementation based on reference CPU implementation where each thread processes all pixels of single event, so few thousands of events are processed simultaneously by hardware threads.

Such approach has however serious drawback on GPU hardware, which is essentially a vector computer. On the NVIDIA CUDA architecture that we use, the threads are collected in batches of 32 threads called *warps*. All threads in a warp must execute same instruction in parallel (SIMD). In the naive implementation each thread is processing a different event with different number of pixels. That results in a double loop with loops bounds different across the threads of a warp. This leads to severe *thread divergence* and, as we have discovered, carries a much higher penalty then naively expected. One would expect that the execution time of a warp, would be approximately the time needed to execute the longest loops, but as it turned out it is much higher. This effect is explained by us in detail in [46]. Additionally, we cannot cache visited pixels and their kernel results since there are not enough registers or shared memory to store such information given each thread processes separate event.

This can be partially circumvented using different pixel calculation scheduling called by us *warp granularity*, where whole warp of 32 threads calculates a single event, depicted in Figure 6.9. Each thread in a warp processes a single pixel for the same event and there is no divergence. Different events are processed by different warps that run independently. This leads to better utilization of available shared memory, registers and improves memory access coherence. However, processor cycles are still wasted by the threads whose pixels lie outside of 3σ ellipse (marked white in Figure 6.9).

Single event is calculated in two passes. First, we calculate denominator of (6.8). This pass iterates through all pixels inside 3σ ellipse bounding box. Each pixel is tested with 3σ ellipse equation. Only pixels belonging to the ellipse are taken into account in denominator calculation (marked grey in Figure 6.9).

During the first pass of *warp granularity* we get an opportunity to cache visited pixels in shared memory and kernel (6.32) value in registers, so the second pass can loop only through pixels visited already without a need to test them for ellipse inclusion and use kernel value calculated once and stored in the register. This is possible as long as the maximum number of pixels belonging to 3σ ellipse is relatively small, which is the case when pixel size is 2 mm or higher.

Calculation of the denominator requires adding the contributions from the 32 threads of the warp. We have done this using the new shuffle instructions introduced in Kepler architecture. This gave a notable performance boost over standard reduction algorithm using shared memory [67].

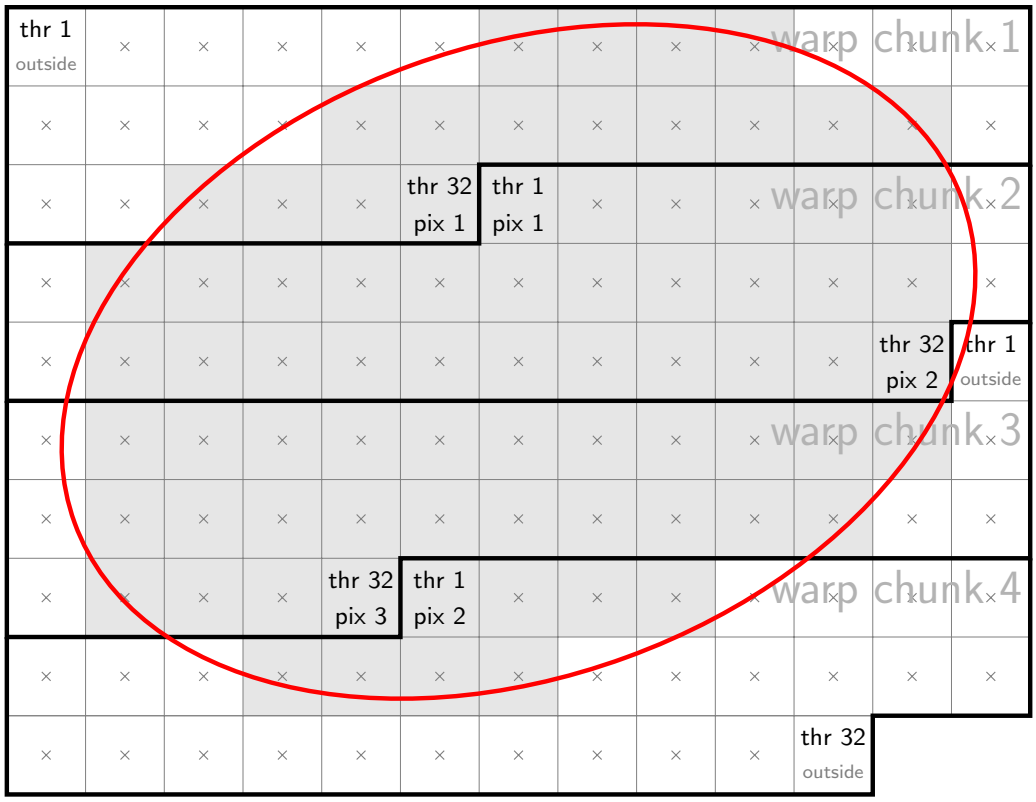


Figure 6.9.: Warp granularity (whole event processed by single warp)

Final optimization is to access ρ (previous iteration image buffer) as texture. This produces noticeable performance boost by using hardware GPU texture unit cache and special 2D access optimized memory layout. However, it can be observed that memory access still takes around 35% of overall iteration time after optimizations.

6.6. Results

6.6.1. Performance benchmark

Time of flight list-mode reconstruction is more computationally demanding than bin-mode, which can be observed comparing times presented in Table 5.3. This is expected and this is the reason that the most of the existing devices still use bin-mode. Number of operations in Algorithm 11 depends on a number of responses (events) and the size of σ in pixels. Relation to number of events is linear, therefore it will be not presented here.

Relation to pixel-size is linear in case of CPU, but it is not in case of GPU, that shows some advantage when using smaller pixels, thus larger image space, as shown in Table 6.1. This comes from the fact that there will be relatively less trailing warp chunks in GPU *warp granularity* with many idle threads – white squares in "*warp chunk 4*" depicted in Figure 6.9, when pixels are smaller. Also some chunks belonging to 3σ bounding box will be completely excluded from the 3σ ellipse, so $i \in \texttt{ellipse}(\widetilde{\mathbf{e}})$ condition will be false for all warp threads, making GPU quickly skip to the next chunk.

Image Size	Pixel Size	CPU[1,2]	GPU[3]	Speedup
213 × 125	4 mm	985 ms	49 ms	20
425 × 250	2 mm	3 978 ms	161 ms	25
850 × 500	1 mm	15 537 ms	585 ms	27

[1] *Intel Xeon E5-2699v3* 3.2 GHz, 6 core, ≈ 307 Gflops, 51 GB/s, ICC 16.0 `-O3`
[2] Best compiler result among GCC 4.8, GCC 5.3 and ICC 16.0
[3] *Intel Xeon E5-2699v3* 3.2 GHz, 6 core, ≈ 307 Gflops, 51 GB/s, GCC 5.3 `-O3`

Table 6.1.: 2D "big" barrel list-mode reconstruction of CPU vs GPU benchmark (10^6 responses, average iteration time of 5 iterations in milliseconds)

6.6.2. Performance estimation in *Roof-line* model

2D strip reconstruction with its estimated in Table 6.2 $Q_{\text{alg}} \approx 40$ is performance bound on CPU with $Q_{\text{cpu}} \approx 30$ and memory bound on GPU having $Q_{\text{gpu}} \approx 60$ factor. However GPU implementation employs texture units with fast cache, hence "correcting" Q_{alg} for that may make it actually very close to Q_{gpu} and peak compute performance. It is also important to ensure that most of the memory accesses are coalesced. This is addressed by warp granularity improving memory access coalescence.

Cost of each transcendental function is equated with 8 elementary FLOP in Table 6.2. This comes from the fact that elementary operations require 4 cycles per warp in latest CUDA architectures, where transcendental functions require 32 cycles. Still these functions are relatively fast on GPU platforms, much faster than if we stored them in some pre-computed look-up array or texture. This is because GPU devices were optimized for computer graphics in first place, where trigonometric and exponential functions are important for calculating rendered graphics lighting.

Operation	FLOP	Mem.Op.
warp-pixel	8	-
ellipse-check	16	-
kernel	76	-
add/mul	60	-
exp	$1 * 8$	-
sqrt	$1 * 8$	-
denom reduce	8	-
rho calculate	6	-
sensitivity read	-	1
via texture	-	1
rho read/update	-	3
via texture	-	1
via atomic	-	1
outer loop	106/20	1/20
- event load	-	1
- bbox	66	-
- add/mul	26	-
- cos/atan	$2 * 8$	-
- sqrt	$3 * 8$	-
- miscellaneous	40	-
Total	~ 120	~ 3
$Q_{\mathbf{alg}}$	~ 40	

Table 6.2.: 2D strip reconstruction operation count per pixel (algorithm 11 inner loop, assumed 20 pixels for each event)

6.6.3. Detailed performance estimation

We made a detailed breakdown for warp granularity algorithm by replacing parts of code with constants and measuring iteration time for remaining active parts. The time difference produced the numbers shown in Table 6.3a. This has led of course to invalid reconstruction results, however it had no impact on the active parts performance. The breakdown shows that memory latency cannot be hidden and balanced out completely by arithmetic operations. Moreover, during kernel execution, many of the global memory accesses are still not coalesced, so *Load/Store* (L/S) unit needs to

perform more than one transaction to fetch data to warp registers (128 byte cache L2 cache line on NVIDIA *Kepler* architecture). For example, if bounding box length is not divisible by 32, L/S unit performs additional transaction for elements on the edge of the bounding box.

It is possible to replace sensitivity memory read with calculation of analytic approximation from the pixel center (6.33). However we use instead a texture holding multi-sampled thus slightly better sensitivity approximation. Reading this texture has comparable cost to analytic calculation. Remaining two memory accesses read current ρ and update new ρ are necessary and cannot be avoided.

Operation	**Time**
Iteration total time	612 ms
Kernel calculation	52 ms
Atomicadd load/store	199 ms
Rho global memory access	117 ms
Shared memory load/store	56 ms
Iterator calculation	20 ms
Check ellipse	119 ms
remaining operations	49 ms

(a) Warp granularity

Operation	**Time**
Iteration total time	955 ms
Kernel calculation	354 ms
Atomicadd load/store	238 ms
Rho global memory access	131 ms
remaining operations	232 ms

(b) Thread granularity

Table 6.3.: 2D strip algorithm iteration time with selected algorithm parts contribution (single iteration, 10^7 events)

6.6.4. PSF FWHM measurements

We will now try to determine 2D strip reconstruction performance characteristics using measures introduced in Section 3.6, according as close as possible to NEMA standard [57] – with one exception, that we will be using MLEM reconstruction instead of filtered back-projection (FBP) required by NEMA.

To do so we have simulated 10^5 coincidences with emission points generated uniformly from the source of 1 mm diameter disc placed $(0, 1\,\text{cm})$ above z-axis. The two gamma were assumed to be exactly collinear, the detection was modeled with attenuation law equation (3.22) with 19 mm thick and 30 cm long detectors. The scatterings were not taken into consideration in the simulation.

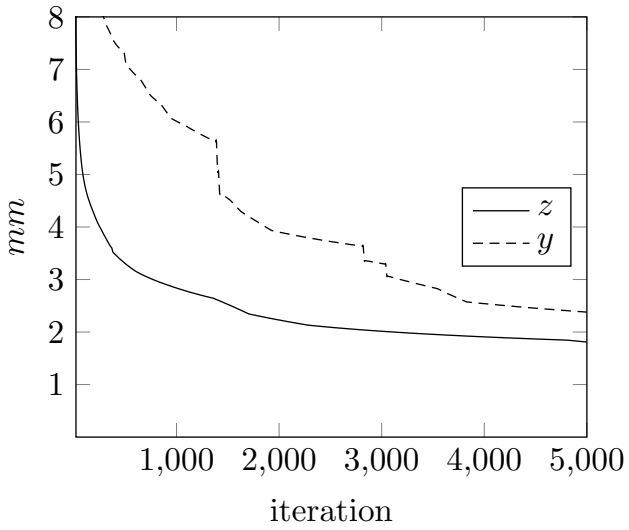
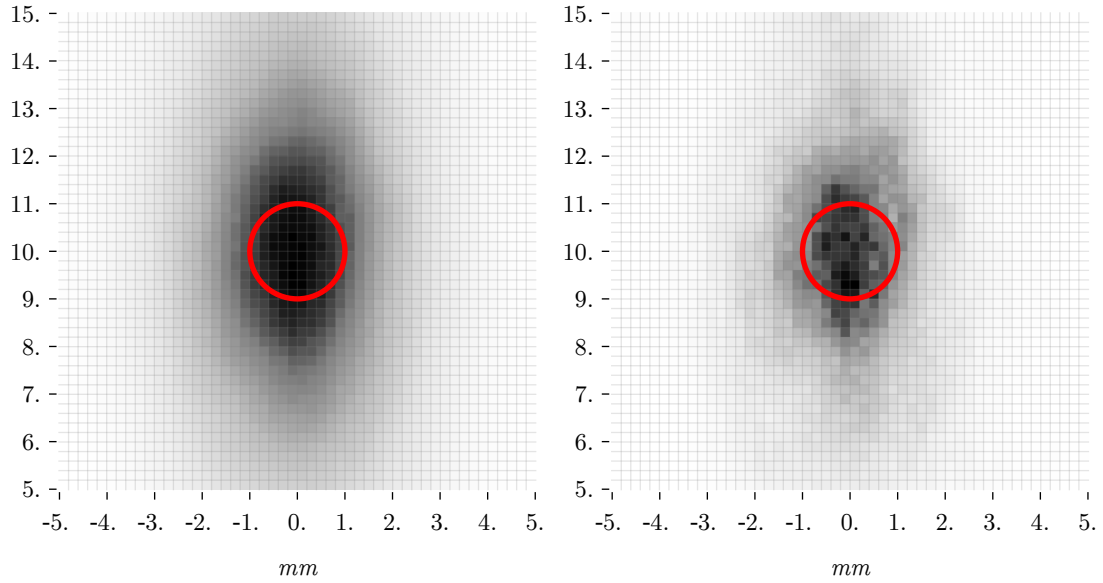


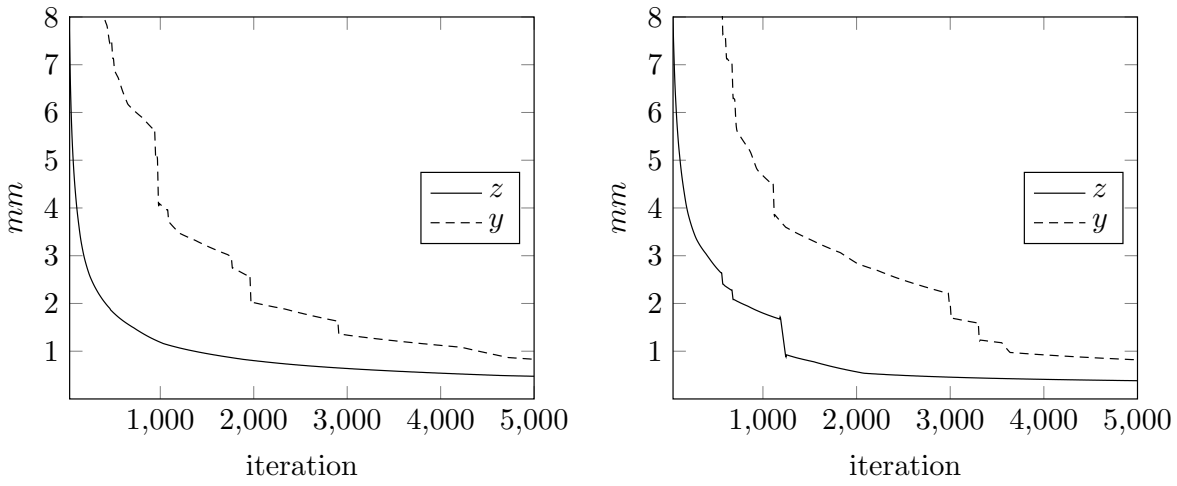
Figure 6.10.: "big" 2 strip J-PET virtual phantom PSF FWHM plot
($R = 42\,\text{cm}$, $L = 50\,\text{cm}$, 10^5 coincidences, $(0, 1\,\text{cm})$ position)



(a) 200 iteration **(b)** 1000 iteration

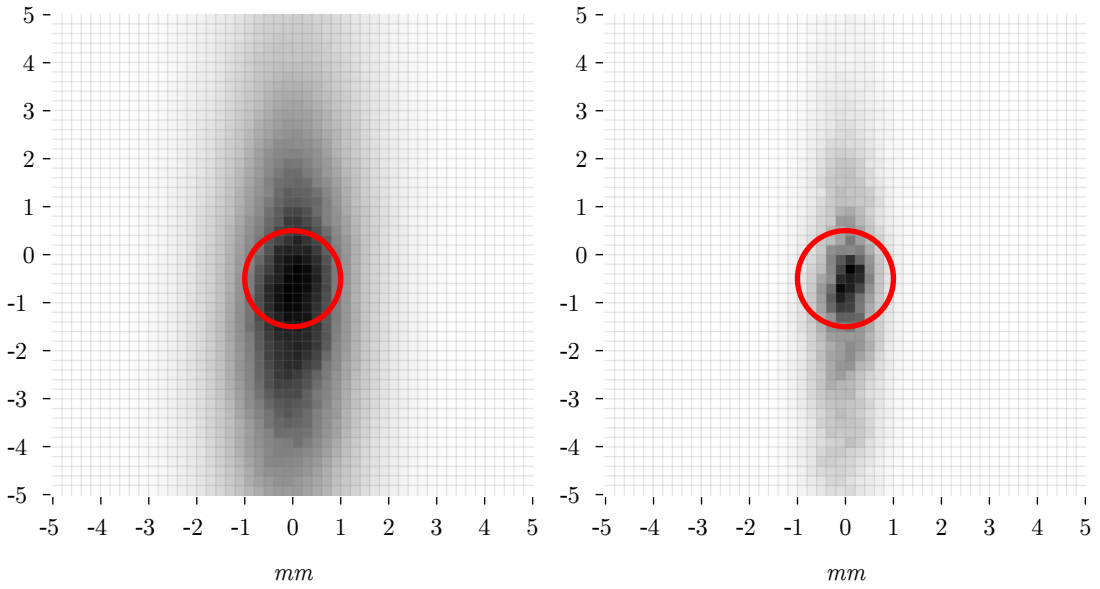
Figure 6.11.: "big" 2 strip J-PET virtual phantom image reconstruction
($R = 42\,\text{cm}$, $L = 50\,\text{cm}$, 0.2 mm pixel size, phantom position marked with circle)

The PSF FWHM for this simulation is shown in Figure 6.10 and detailed images for few selected iterations are shown in Figure 6.11. The convergence of PSF image reconstruction is slow. This fact was already observed by others [68]. After around 3500 iterations there is no further improvement in FWHM and the noise starts to appear on the resulting images. The steps visible on PSF FWHM plots come from the changes of pixel holding maximum value.



(a) $(0, 0)$ central position (17 147 coincidences) **(b)** $\sim (10\,\text{cm}, 10\,\text{cm})$ position (25 061 coincidences)

Figure 6.12.: 2 strip detector prototype real measurement PSF FWHM plot



(a) 200 iteration **(b)** 1000 iteration

Figure 6.13.: 2 strip detector prototype reconstruction images for central phantom position ($R = 20.5\,\text{cm}$, $L = 30\,\text{cm}$, 0.2 mm pixel size, source placement marked with circle)

There are also preliminary 2 strip detector prototype scan results presented in [69]. These were done with real point-like source of $\sim 1\,\text{mm}$ diameter active area. The source was placed in 9 positions. Below in Figure 6.12 we present results for two example positions and sample images for first central position in Figure 6.13.

6.6.5. Quality measurement

As a pre-requisite for the quality estimation, we have conducted two simulations for virtual Shepp-Logan phantom - first producing 1 million coincidences, second producing 10 million, assuming $\sigma_{z_{u,d}} = 10\,\text{mm}$, $\sigma_{\Delta l} = 40mm$ time resolution [14]. Quality estimation presented in Table 6.14 was performed by comparing the emitted density phantom image 6.15a, 6.16a to the reconstructed image using normalized root mean square error (NRMSE), defined in Section 3.6.3.

It is important to note that input for the reconstruction is what has been measured with errors, depicted in Figures 6.15c, 6.16c. We neither can measure what was real emission density shown in Figures 6.15a, 6.16a, nor the exact detected emission density shown in Figures 6.15b, 6.16b.

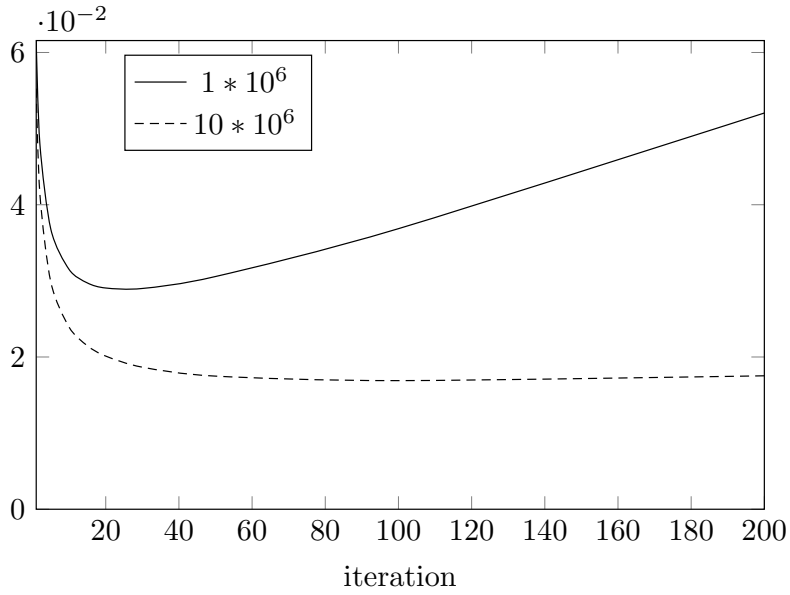
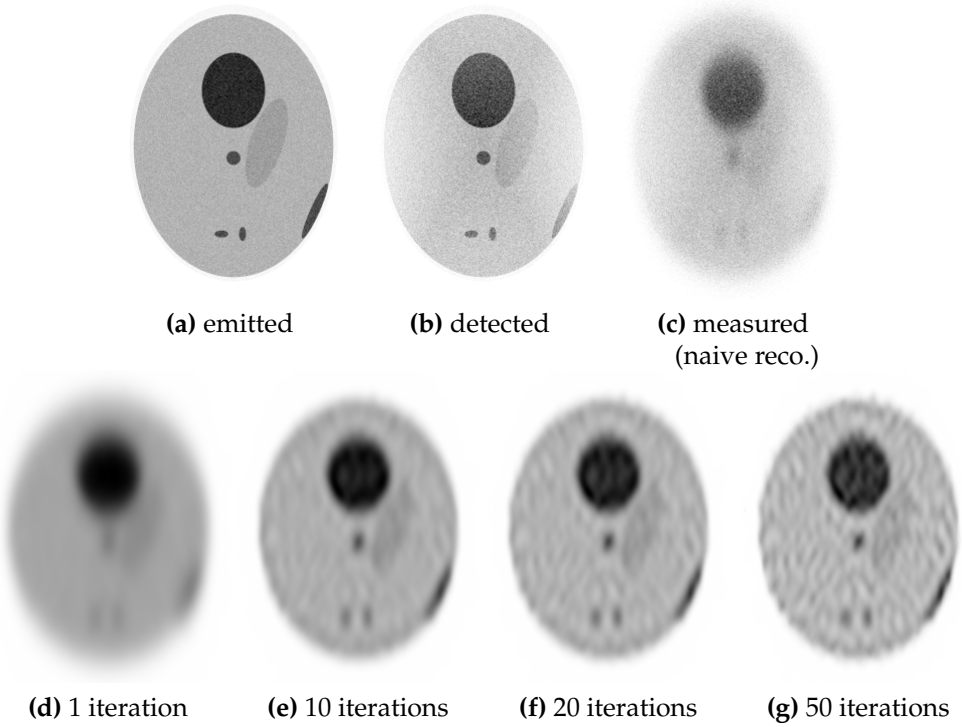


Figure 6.14.: "big" 2 strip J-PET Shepp-Logan phantom reconstruction NRMSE plot ($R = 42\,\text{cm}$, $L = 50\,\text{cm}$, NRMSE as a function of the number of MLEM iterations, for two numbers of coincidences)

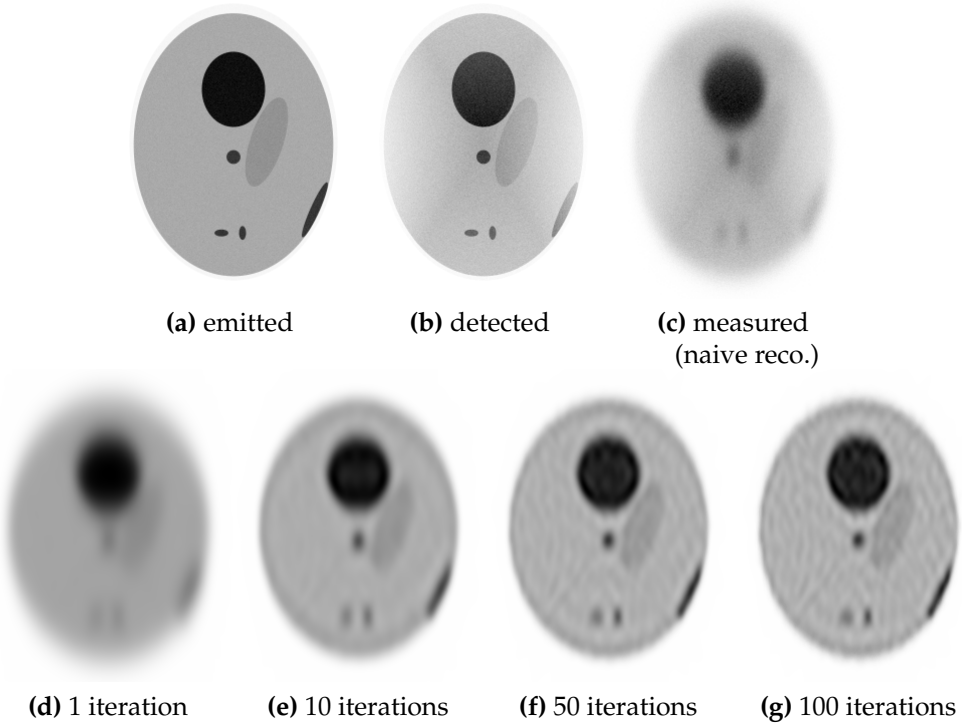
It can be observed that number of coincidences has visible impact on image quality. Too low number of coincidences cause noise to manifest itself soon after performing 20 iterations, reflected by NRMSE raise in Figure 6.14. Higher number of coincidences enables to perform more iterations without the noise manifestation. Altogether our MLEM TOF method shows to be effective in reconstructing smaller details barely visible on "naive reconstruction", that could not be extracted by simple image filtering.



(a) emitted **(b)** detected **(c)** measured (naive reco.)

(d) 1 iteration **(e)** 10 iterations **(f)** 20 iterations **(g)** 50 iterations

Figure 6.15.: J-PET Shepp-Logan virtual phantom simulation and image reconstruction (10^6 coincidences)



(a) emitted **(b)** detected **(c)** measured (naive reco.)

(d) 1 iteration **(e)** 10 iterations **(f)** 50 iterations **(g)** 100 iterations

Figure 6.16.: J-PET Shepp-Logan virtual phantom simulation and image reconstruction ($10 * 10^6$ coincidences)

Chapter 7.

3D J-PET reconstruction

Supplied with the knowledge and the experience presented in previous chapters, we are now able to form the final statistical model and algorithm in order to perform full 3D reconstruction.

Presented reconstruction is a combination or a "hybrid" of two subproblem solutions presented respectively for $x-y$ 2D barrel plane and $z-y$ 2D strip plane. The 2D barrel model represents complex geometry, while 2D strip model embeds time measurement errors.

7.1. Combining 2D barrel and 2D strip reconstruction

Our objective is to find $P\left(\widetilde{\mathbf{e}} = (t, \widetilde{z_u}, \widetilde{z_d}, \widetilde{\Delta l}) \,|\, v\right)$, necessary for the 3D version of J-PET reconstruction, where $\widetilde{\mathbf{e}}$ is a single response, t is a detector pair index, $\widetilde{z_u}$, $\widetilde{z_d}$ are measured positions along the detectors (6.2), $\widetilde{\Delta l}$ is a measured position difference between the detectors (6.3), and v represents a voxel.

In order to do so, we need to integrate $P(\widetilde{\mathbf{e}} \,|\, p, \Omega)$, a probability of registering an emission from the point p in the direction Ω as a response $\widetilde{\mathbf{e}}$, over all emission directions Ω and all points p of the voxel v

$$P\left(\widetilde{\mathbf{e}} = (t, \widetilde{z_u}, \widetilde{z_d}, \widetilde{\Delta l}) \,|\, v\right) = \int\limits_{p \in v} \int \mathrm{d}p \; \mathrm{d}\Omega \, P(\widetilde{\mathbf{e}} \,|\, p, \Omega) \tag{7.1}$$

We may express the inner integral in spherical coordinate system originating from the point p

$$P\Big(\widetilde{\mathbf{e}} = (t, \widetilde{z_u}, \widetilde{z_d}, \widetilde{\Delta l}) \,|\, v\Big) = \int\limits_{p \in v} \int\limits_0^{\pi} \int\limits_0^{\pi} \mathrm{d}p \;\sin\theta \,\mathrm{d}\theta \,\mathrm{d}\phi \, P(\widetilde{\mathbf{e}} \,|\, p, \theta, \phi) \tag{7.2}$$

We fix the orientation of the coordinate system, so detector pair t lies along its "equator", making ϕ an emission angle in 2D strip plane and θ an emission angle in 2D barrel plane, as shown in Figure 7.1.

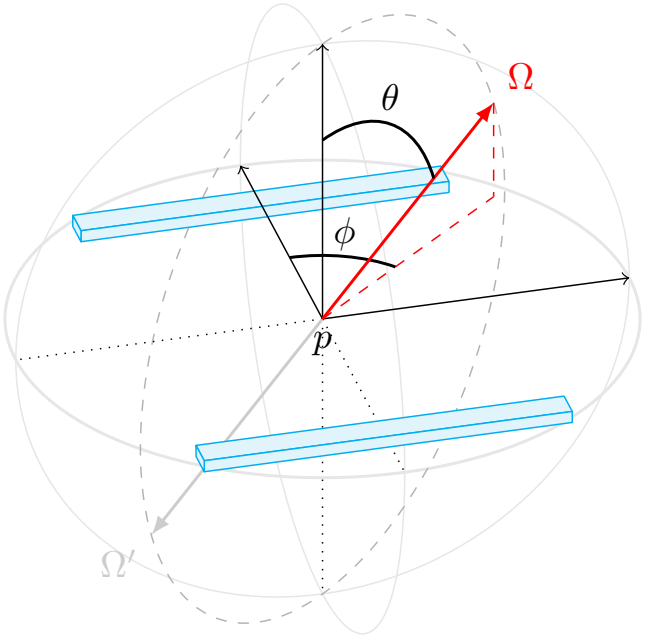


Figure 7.1.: Spherical coordinate system orientation
(p denotes emissions point, Ω, Ω' denote quanta emission directions)

Next, we can assume J-PET scanner geometry related uncertainties and time measurement errors, introduced in Chapter 3, are independent, so

$$P\Big(\widetilde{\mathbf{e}} = (t, \widetilde{z_u}, \widetilde{z_d}, \widetilde{\Delta l}) \,|\, p, \theta, \phi\Big) = P(t \,|\, p, \theta, \phi)\, P\Big(\widetilde{z_u}, \widetilde{z_d}, \widetilde{\Delta l} \,|\, t, p, \theta, \phi\Big) \tag{7.3}$$

where $P(t \,|\, p, \theta, \phi)$ denotes probability of interaction with TOR t detector pair, depending on intrinsic detector response function (IDRF) introduced in Section 3.5, and $P\Big(\widetilde{z_u}, \widetilde{z_d}, \widetilde{\Delta l} \,|\, t, p, \theta, \phi\Big)$ denote 2D strip measurement error model (6.13) for TOR t.

Now we make an approximation that interaction probability depends weakly on ϕ, as shown in Section 6.2, and that time measurement errors do not depend on θ, so

$$P\left(t, \widetilde{z_u}, \widetilde{z_d}, \widetilde{\Delta l} \,|\, v\right) \approx \int\limits_{p \in v} \int\limits_0^\pi \int\limits_0^\pi \mathrm{d}p \;\sin\theta \;\mathrm{d}\theta \;\mathrm{d}\phi \, P(t \,|\, p, \theta) \, P\left(\widetilde{z_u}, \widetilde{z_d}, \widetilde{\Delta l} \,|\, t, p, \phi\right) \tag{7.4}$$

$$= \int\limits_{p \in v} \mathrm{d}p \int\limits_0^\pi \sin\theta \;\mathrm{d}\theta \, P(t \,|\, p, \theta) \int\limits_0^\pi \mathrm{d}\phi \, P\left(\widetilde{z_u}, \widetilde{z_d}, \widetilde{\Delta l} \,|\, t, p, \phi\right) \tag{7.5}$$

J-PET detectors are thin enough, so orientation depicted in Figure 7.1 ensures that effective θ range where $P(t \,|\, p, \theta)$ is non zero is very close to 90°, therefore assuming $\sin\theta \approx 1$ when $P(t \,|\, p, \theta) \neq 0$, we can calculate the inner itegrals

$$\approx \int\limits_{p \in v} \mathrm{d}p \; P(t \,|\, p) \; P\left(\widetilde{z_u}, \widetilde{z_d}, \widetilde{\Delta l} \,|\, t, p\right) \tag{7.6}$$

Next, we make the same approximation we made before in equation (6.33), taking single 2D strip kernel value at voxel's central point p_v for all points in the voxel

$$\approx \int\limits_{p \in v} \mathrm{d}p \; P(t \,|\, p) \; P\left(\widetilde{z_u}, \widetilde{z_d}, \widetilde{\Delta l} \,|\, t, p_v\right) \tag{7.7}$$

$$= P\left(\widetilde{z_u}, \widetilde{z_d}, \widetilde{\Delta l} \,|\, t, p_v\right) \int\limits_{p \in v} \mathrm{d}p \, P(t \,|\, p) \tag{7.8}$$

and because of the first approximation (7.4), $P(t \,|\, p)$ does not depend on point p's z-coordinate as long p is in the FOV, this leads us to the final approximation

$$P\left(t, \widetilde{z_u}, \widetilde{z_d}, \widetilde{\Delta l} \,|\, v\right) \approx H(v) \, P(t \,|\, i) \, P\left(\widetilde{z_u}, \widetilde{z_d}, \widetilde{\Delta l} \,|\, t, p_v\right) \tag{7.9}$$

where $H(v)$ denotes constant defining height (along z-axis) of voxel v, $P(t \,|\, i)$ denotes 2D system matrix value for TOR t and pixel $i = (v_x, v_y)$, $P\left(\widetilde{z_u}, \widetilde{z_d}, \widetilde{\Delta l} \,|\, t, p_v\right)$ denotes 2D strip conditional probability, calculated at voxel's central point p_v as

$$P\left(\widetilde{z_u}, \widetilde{z_d}, \widetilde{\Delta l} \,|\, t, p_v\right) = \frac{P_t\left(\widetilde{z_u}, \widetilde{z_d}, \widetilde{\Delta l} \cap p_v\right)}{s_t(p_v)} \tag{7.10}$$

that is 2D strip kernel value (6.32) divided by 2D strip frame sensitivity (6.11), similarly to (6.34), both parametrized by TOR t, having specific detectors distance.

2D strip kernel expects $(\widetilde{z}, \widetilde{y}, \widetilde{\phi})$ arguments, hence before running actual reconstruction all $\widetilde{\mathbf{e}} = (t, \widetilde{z_u}, \widetilde{z_d}, \widetilde{\Delta l})$ are converted into $\widetilde{\mathbf{e}} = (u, d, \widetilde{z}, \widetilde{y}, \widetilde{\phi})$ using equation (6.4), where u, d denote TOR t detectors indices. With this representation 3D P can be expressed as

$$P\left(\widetilde{\mathbf{e}} = (u, d, \widetilde{z}, \widetilde{y}, \widetilde{\phi}) \,|\, v\right) = P(u, d \,|\, v)\; P\left(\widetilde{z}, \widetilde{y}, \widetilde{\phi} \,|\, u, d, \mathrm{proj}_{u,d} v\right) \tag{7.11}$$

where $\mathrm{proj}_{u,d}(v) = (z, y)$ denotes projection of voxel v central point to 2D strip plane for u, d detector pair.

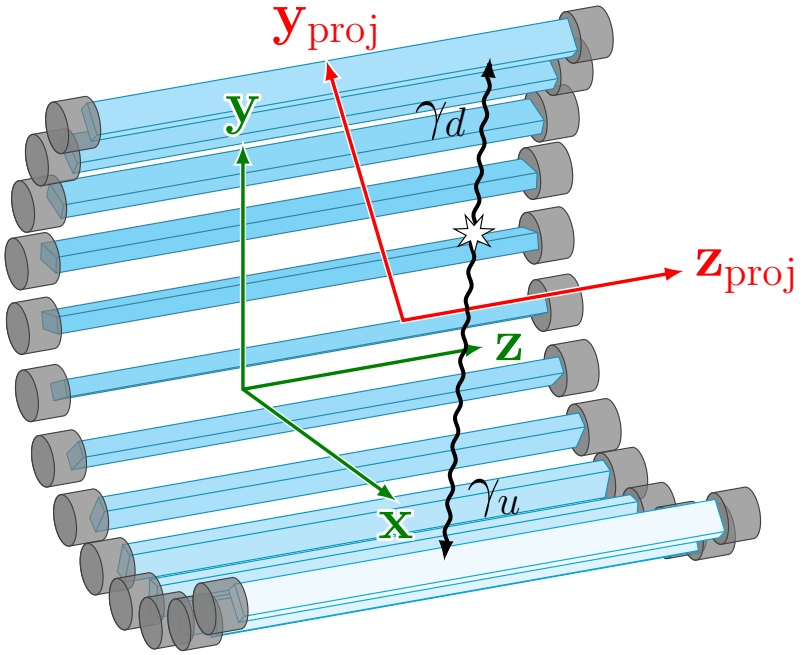


Figure 7.2.: Projection from 3D space (*green*) to 2D detector pair space (*red*)

2D strip plane is spanned by detector pair u, d specific to each coincidence. Therefore 2D strip $\mathbf{z}_{\mathrm{proj}}$, $\mathbf{y}_{\mathrm{proj}}$ axes are specific to detector pair u, d and are not identical to 3D space $\mathbf{z}$, $\mathbf{y}$ axes as shown in Figure 7.2. Having $(x_u, y_u, 0)$ and $(x_d, y_d, 0)$ detector pair central points, the 2D strip space is defined by

$$\begin{aligned}
\mathrm{origin}_{u,d} &= \left(\frac{x_u + x_d}{2}, \frac{y_u + y_d}{2}, 0\right) \\
\vec{z}_{u,d} &= (0, 0, 1) \\
\vec{y}_{u,d} &= \left(\frac{|x_u - x_d|}{\sqrt{(x_u - x_d)^2 + (y_u - y_d)^2}}, \frac{|y_u - y_d|}{\sqrt{(x_u - x_d)^2 + (y_u - y_d)^2}}, 0\right)
\end{aligned} \tag{7.12}$$

7.2. Generic CPU implementation

3D reconstruction Algorithm 12 performs list-mode MLEM described by (3.18). It is a "fusion" of Algorithms 10 and 11 performing image reconstruction for 2D barrel and 2D strip.

Algorithm 12 3D hybrid list-mode reconstruction step

function 3DRECONSTRUCTIONSTEP(ρ, ρ_{new})
  $\rho_{\text{new}}(v) \leftarrow 0$
  for all $\widetilde{\mathbf{e}} \in \widetilde{\mathbf{E}}$ **do** ▷ iterate through all scan responses
    $t = (u, d) \leftarrow \texttt{TOR}(\widetilde{\mathbf{e}})$
    denom $\leftarrow 0$ ▷ reset denominator to zero
    for $i \in \texttt{pixels}(t) \cap \texttt{bbox}_{3\sigma}(\widetilde{\mathbf{e}})$ **do** ▷ iterate 3σ bounding box pixels
      weight $\leftarrow P(t \mid i)$ ▷ load system matrix value
      $y \leftarrow \text{proj}_{u,d}(i)$ ▷ projection to TOR t frame
      for $z \in \texttt{planes}(\texttt{bbox}_{3\sigma}(\widetilde{\mathbf{e}}))$ **do** ▷ iterate planes along z-axis
        $v \leftarrow (i_x, i_y, z)$
        denom $\leftarrow$ denom $+$weight $\; s(v)\, P(\widetilde{\mathbf{e}} \mid t, z, y)\, \rho(v)$
      end for
    end for
    for $i \in \texttt{pixels}(t) \cap \texttt{bbox}_{3\sigma}(\widetilde{\mathbf{e}})$ **do** ▷ iterate 3σ bounding box pixels
      weight $\leftarrow P(t \mid i)$ ▷ load system matrix value
      $y \leftarrow \text{proj}_{u,d}(i)$ ▷ projection to TOR t frame
      for $z \in \texttt{planes}(\texttt{bbox}_{3\sigma}(\widetilde{\mathbf{e}}))$ **do** ▷ iterate planes along z-axis
        $v \leftarrow (i_x, i_y, z)$
        $\rho_{\text{new}}(v) \leftarrow \rho_{\text{new}}(v) + \dfrac{\text{weight}\, P(\widetilde{\mathbf{e}} \mid t, z, y)\, \rho(v)}{\text{denom}}$ ▷ update ρ_{new}
      end for
    end for
  end for
end function

In Algorithm 12, $P(t \mid i)$ denotes pre-calculated 2D system matrix value, thus is effectively a memory access. This value is loaded once per pixel and can be reused in the innermost loop iterating over the planes along z-axis. Also, $s(v)$ denotes pre-calculated full 3D barrel sensitivity, read from the memory and coming from the 3D simulation described in following Section 7.4.

$P(\widetilde{\mathbf{e}} \mid t, z, y)$ denotes 2D strip analytic kernel value (6.34). This is effectively calculated as unconditional kernel (6.32) divided by sensitivity $s_t(z, y)$ (6.11), necessary to form conditional probability. In opposition to 2D strip implementation, $s_t(z, y)$ is not pre-calculated and cached, but calculated on-fly. This is because we would need to

pre-calculate it for all TORs – detector pairs, but such pre-calculated sensitivity would barely fit into the available memory.

Inner loop constructs actual voxel $v = (i_x, i_y, z)$ out of pixel i and plane z. This voxel is used to lookup current ρ and update next ρ_{new} value.

Some other expressions are also pre-calculated prior running actual reconstruction iterations. An intersection $\texttt{pixels}(t) \cap \texttt{bbox}_{3\sigma}(\widetilde{\mathbf{e}})$, representing system matrix pixels belonging to TOR t that are in 3σ bounding box, is pre-calculated and stored as simple pixel list index range. Therefore $i \in \texttt{pixels}(t) \cap \texttt{bbox}_{3\sigma}(\widetilde{\mathbf{e}})$ loop effectively iterates over the elements of the pixel list associated with each $\widetilde{\mathbf{e}}$ response. Same applies to response z plane range denoted as $\texttt{planes}(\texttt{bbox}_{3\sigma}(\widetilde{\mathbf{e}}))$, this is pre-calculated and stored as simple voxel z coordinate range assigned to each $\widetilde{\mathbf{e}}$.

Pixel list and weights are rewritten into structure of arrays (SoA) upon loading sparse matrix depicted in Figure 4.4. Since 3D z coordinate is identical to 2D u, d projected strip space, only y is calculated as a result of the projection of pixel i's central point onto $\vec{y}_{u,d}$ anchored to $\text{origin}_{u,d}$ (7.12), denoted as $\text{proj}_{u,d}(i)$ in Algorithm 12.

7.3. GPU accelerated implementation

GPU implementation uses warp-granularity scheduling, similar to the one shown in Figure 6.9, to process all $\widetilde{\mathbf{e}}$ TOR pixels from Algorithm 12 using a single warp. This makes some memory loads coalesced, especially TOR pixel access and $P(t \mid i)$ system matrix read. However, unlike 2D strip GPU implementation, z planes are not processed in parallel. Actually, processing them in parallel slightly degrades performance, as some additional calculations are necessary to determine voxel v from warp thread index.

Some of the 2D strip GPU implementation optimizations could not be ported to 3D implementation due to limitations of shared memory and registers. Most noticeable difference from Algorithm 11 is that $P(\widetilde{\mathbf{e}} \mid z, y)$ cannot be cached in the kernel variable, simply because there are not enough registers. The total number of voxels processed, thus $P(\widetilde{\mathbf{e}} \mid z, y)$ calls necessary to calculate denominator, is $\texttt{width}(\texttt{TOR}(\widetilde{\mathbf{e}}))$ times higher than in 2D strip case, where $\texttt{width}(t)$ denotes width for given TOR t in pixels, and depending on resolution can be 3 up to 20 pixels for J-PET scanner geometries. This

results in loading $\rho(v)$ and calling $P(\widetilde{\mathbf{e}}\,|\,z,y)$ calculation twice as often as in 2D strip GPU implementation.

On the other hand, using on-fly calculation for $s_t(z,y)$ does not bring any penalty on GPU, as arctan is an optimized hardware instruction. In fact, it even helps to hide global memory access latency for system matrix and barrel sensitivity reads. Finally 3D reconstruction GPU implementation uses 3D textures for current ρ and $s(v)$ lookup, which improves memory access coalescence and uses fast texture cache, reducing memory access latency.

7.4. Simulation of 3D J-PET scanner

In order to be able to generate an input for 3D reconstruction we had to extend existing 2D barrel simulation to the full 3D. This has also let us to use 3D version of sensitivity for the image reconstruction. While it is possible to use 2D sensitivity in Algorithm 12, using 3D version ensures proper correction against sensitivity that noticeably changes along z-axis for same v position in x-y barrel plane. This is done for the cost of more frequent memory access, for each voxel instead of each pixel in TOR.

The 3D version of the simulation is in the essence a minimal extension to Algorithm 6. Instead of taking random (x,y,θ) where $(x,y) \in i$ – simulated pixel and $\theta \in [\,0,\pi)$, we take random (x,y,z,θ,ϕ) where $(x,y,z) \in v$ – simulated voxel and $\theta,\phi \in [0,\pi)$ are angle components of spherical coordinates of a random point on unit sphere, representing random direction.

Prior starting 2D barrel intersection testing described in Algorithm 6, we first check if one of the emitted quanta did not escape through one of the open sides of the barrel. This is done by finding intersection points of the emission beam crossing the surface of an infinitely long cylinder, having its axis equal to J-PET scanner z-axis, and a radius of J-PET barrel radius. As depicted in Figure 7.3, if one of the intersection points lies out of the barrel – $z(p_1) < -\frac{L}{2} \vee z(p_1) > \frac{L}{2} \vee z(p_2) < -\frac{L}{2} \vee z(p_2) > \frac{L}{2}$, we consider the emission to escape the barrel and we skip to the next random emission event.

Otherwise, we continue checking the 2D barrel intersection for (x,y,θ) using 2D version Algorithm 6. Finally, instead testing interaction with 2D secant length l, we use corrected length $l_{\text{corr}} = l/\cos\phi$ according to equation (6.40). This gives us an

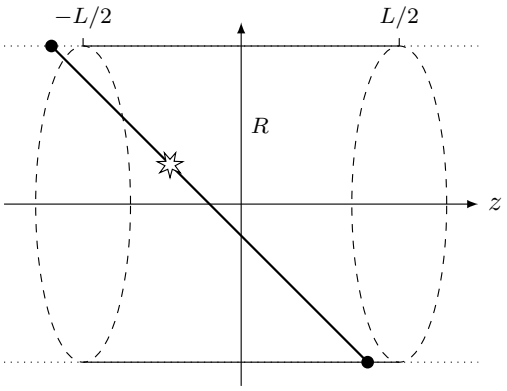


Figure 7.3.: Upper emission escaping through left open side of the 3D barrel

approximation for 3D secant length, that does not take into account the case when the beam crosses the side of the scintillator attached to PMT, as shown in Figure 7.4.

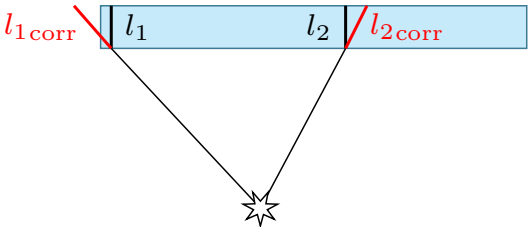


Figure 7.4.: Emission angle correction for 3D

7.5. Results

7.5.1. Performance benchmark

3D reconstruction Algorithm 12 processes average TOR width (measured in pixels) times more voxels than 2D strip Algorithm 11. So the performance is expected to drop proportionally to the average TOR width. Actual performance drops about 4× more for GPU implementation than this as shown in Table 7.1. This can be however easily explained. First of caching kernel values cannot be ported as number of values would not fit in the available fast register memory, so all kernel computation operations are basically done twice comparing to 2D version. Moreover, 3D reconstruction operates on much bigger image space and memory comparing to 2D version. Therefore 3D

memory accesses are less likely coalescent and hitting the cache when using 3D texture than in 2D strip implementation.

The CPU generic implementation performance shows much bigger degradation comparing to the 2D version. There is no cause for concern though, as CPU implementation just acts as reference and the severe degradation comes from the fact that we calculate sensitivity for 2D strip on-fly and arctan function is slow on CPU. Moreover, CPU has no 3D texture or spatially optimized hardware cache, therefore the linear memory cache hit is much less likely than in 2D strip implementation.

Image Size	Pixel Size	CPU[1,2]	GPU[3]	Speedup
100^2	8 mm	5 140 ms	49 ms	105
200^2	4 mm	27 322 ms	250 ms	109
400^2	2 mm	176 862 ms	3 442 ms	51
800^2	1 mm	1 708 898 ms	29 512 ms	57

[1] *Intel Xeon E5-2699v3* 3.2 GHz, 6 core, ≈ 307 Gflops, 51 GB/s, ICC 16.0 `-O3`

[2] Best compiler result among GCC 4.8, GCC 5.3 and ICC 16.0

[3] *GeForce GTX 980 Ti* 1 Ghz, 6 GB, 5 632 Gflops, 336 GB/s, CUDA 7.5 `-O3`

Table 7.1.: 3D "big" barrel list-mode reconstruction CPU vs GPU benchmark (average iteration time of 5 iterations in milliseconds)

7.5.2. Performance estimation in *Roof-line* model

3D hybrid reconstruction ($Q_{\text{alg}} \approx 27$) is memory bound on both CPU ($Q_{\text{cpu}} \approx 30$) and GPU ($Q_{\text{cpu}} \approx 60$). This comes from the fact that 2D barrel part shown in Table 5.4 is heavily memory bound and 2D strip part is just nearly performance bound. Further optimization could employ replacing two 2D barrel TOR pixels and their weights with some analytic approximation, based on the distance to the line segment spanned across TOR detectors. Replacing sensitivity read with some analytic approximation is however not feasible as current J-PET prototypes geometry is non-trivial with its loosely packed scintillators as shown in Figure 5.9, therefore even if it was possible, calculating such sensitivity on-fly would be more costly than looking it up from the texture.

Operation	FLOP	Mem.Op.
warp-pixel	8	-
2D TOR pixels	-	1
2D barrel pixel weights	-	1
2D strip projection	8	-
2D strip kernel	$76 * 4$	-
denom reduce	$8 * 4$	-
rho calculate	$6 * 4$	-
sensitivity read	-	$4 * 1$
- via texture	-	$4 * 1$
rho read/update	-	$4 * 2$
- via texture	-	$4 * 1$
- via atomic	-	$4 * 1$
outer loop	4	2/10
- event load	-	2
- miscellaneous	40	-
Total	~ 380	~ 14
$Q_{\mathbf{alg}}$	~ 27	

Table 7.2.: 3D hybrid reconstruction operation count per pixel
(algorithm 12 inner loop, assumed 10 pixels, 40 voxels and 4 z-planes for each event)

7.5.3. PSF FWHM measurements

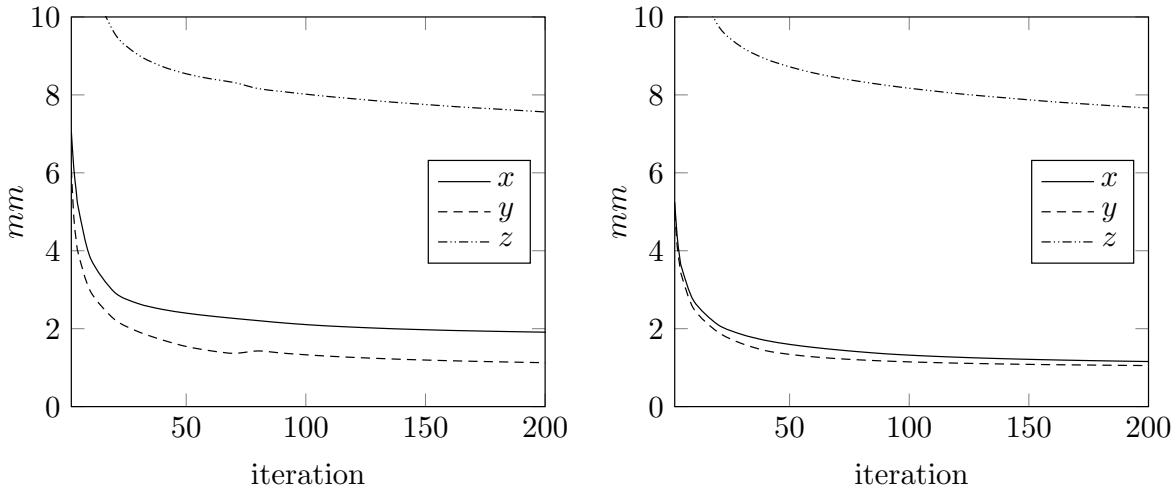
We have used a data coming from two sources as an input for our image reconstruction in order to determine PSF FWHM. The former source was the GATE simulations described in [8, 58]. The latter is our own Monte-Carlo simulation.

We have generated emission points uniformly from the source, 1 mm diameter sphere in this case, and then direction of emission also uniformly. The two gamma were assumed to be exactly collinear, the detection was modeled with attenuation law equation (3.22) and the scatterings were not taken into consideration, opposite to GATE simulation.

	Source					
	(1 cm, 0, 0)			(10 cm, 0, 0)		
Iteration	**x-axis**	**y-axis**	**z-axis**	**x-axis**	**y-axis**	**z-axis**
1	9.059	8.803	25.279	7.402	7.162	25.319
5	5.081	3.909	13.409	3.546	3.335	13.562
10	3.811	2.956	11.071	2.708	2.504	11.257
50	2.399	1.543	8.542	1.597	1.341	8.721
100	2.105	1.330	8.018	1.321	1.148	8.171
200	1.910	1.127	7.562	1.157	1.053	7.664

Table 7.3.: PSF FWHM as a function of the number of MLEM iterations (FWHM values in mm along each of 3 axes, for two source positions)

Assuming $\sigma_{z_{u,d}} = 10\,\text{mm}$, $\sigma_{\Delta l} = 40mm$ time resolution again [14] for both sources, both have produced almost identical results, therefore without loss of generality we present below only plots and numbers for the latter.



(a) (1 cm, 0, 0) **(b)** (10 cm, 0, 0)

Figure 7.5.: PSF FWHM as a function of the number of MLEM iterations

PSF FWHM is rapidly decreasing during first 50 iterations as shown in Figure 7.5, after 200 iterations there is almost no decrease alone x and y axis, however FWHM still decreases along z axis. This can be explained by slow PSF convergence of 2D strip kernel part of 3D reconstruction as shown in Figure 6.10.

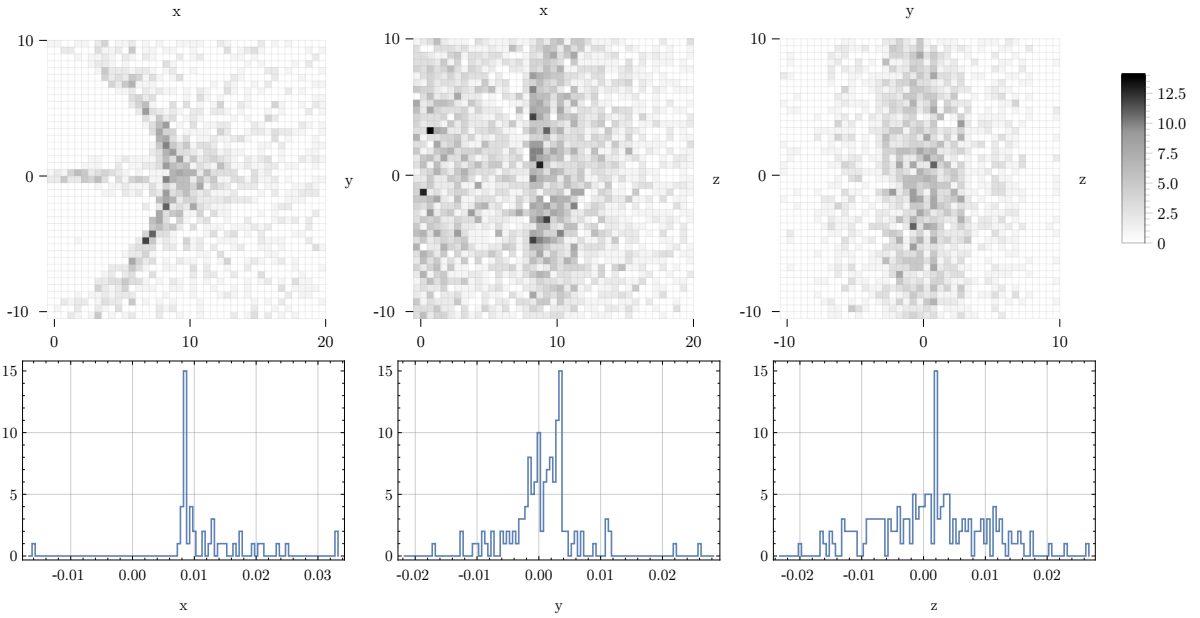


Figure 7.6.: Naive reconstruction of the 1mm source at $(10, 0, 0)$ mm (cuts taken thru the maximum)

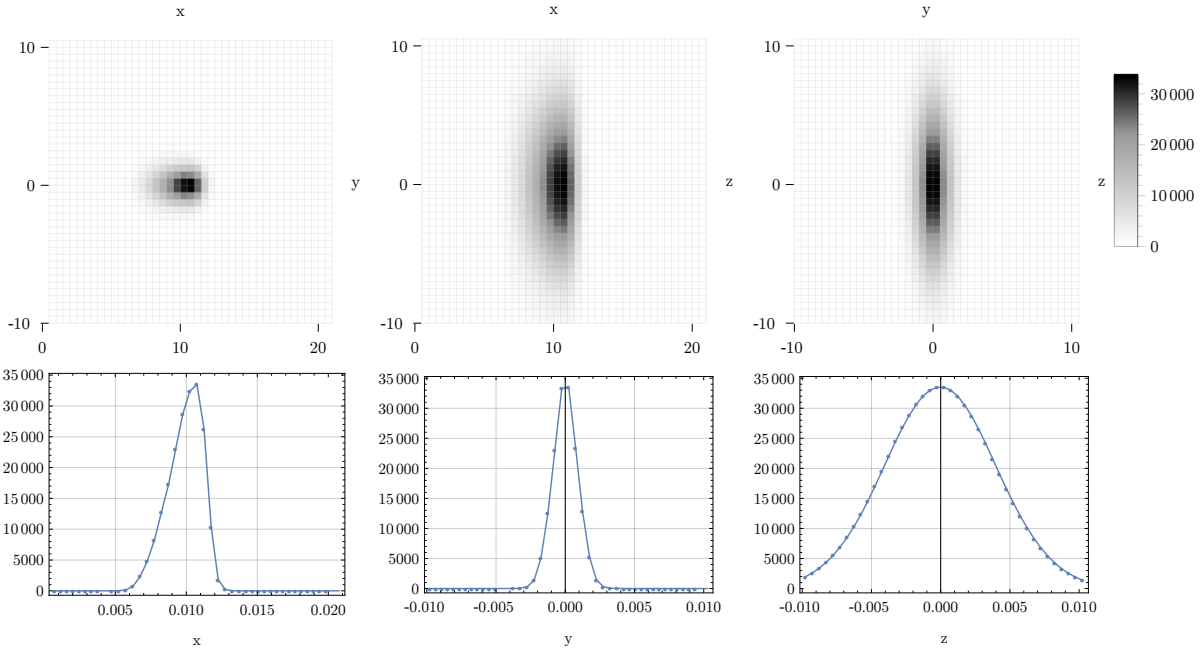
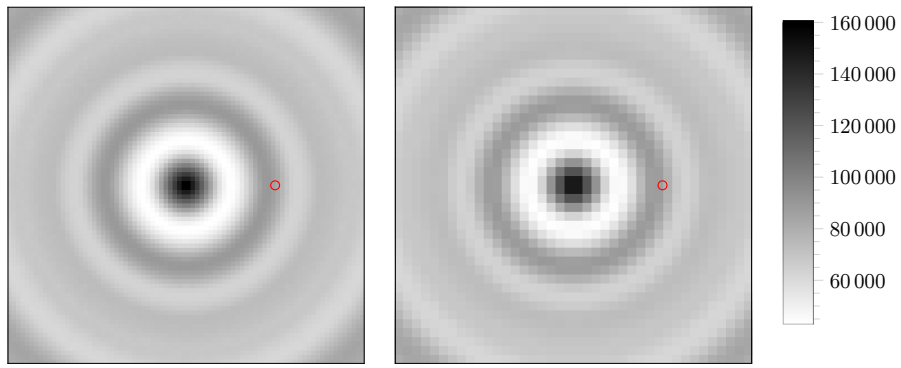


Figure 7.7.: MLEM reconstruction after 100 iterations of the 1 mm source at $(10, 0, 0)$ mm (cuts taken thru the maximum)

The asymmetrical PSF shape in $x - y$ axis visible on left images in Figure 7.6 and 7.7 comes from the erratic J-PET scanner sensitivity around the source placement $(1\,\text{cm}, 0, 0)$ depicted in Figure 7.8.



(a) 0.5 mm **(b)** 1 mm

(c) 2 mm **(d)** 4 mm

Figure 7.8.: Sensitivity of 40×40 mm central region in the central plane of the J-PET scanner (images for different pixel sizes, $(1\,\text{cm}, 0, 0)$ source placement marked with circle)

7.5.4. Quality measurements

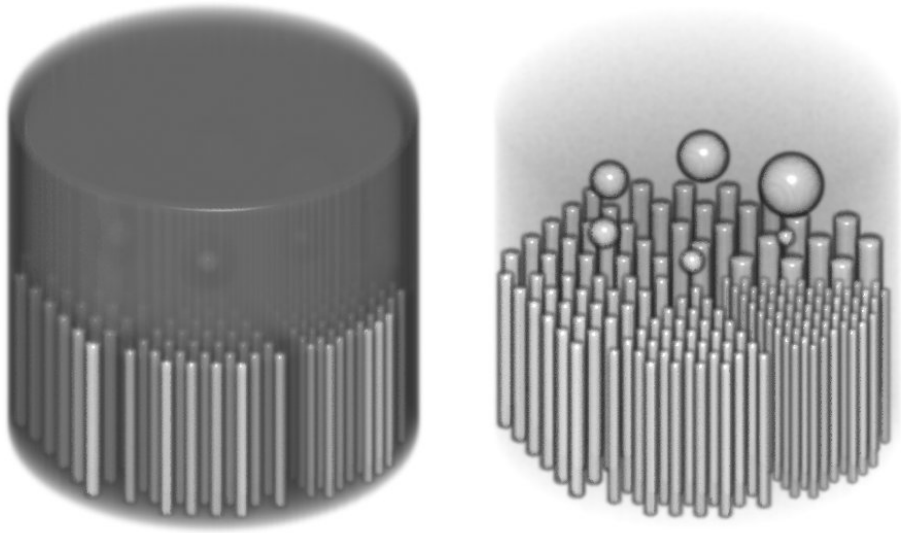
In order to perform 3D reconstruction image quality measurements, we have created virtual phantom model based on *Deluxe Jaszczak Phantom* according to description provided in Table 7.4 and [70, 71] – depicted in Figure 7.9. Next we have performed 4 simulations for the response of the J-PET "big" barrel 3D PET scanner to this virtual phantom, assuming $\sigma_{z_{u,d}} = 10\,\text{mm}$, $\sigma_{\Delta l} = 40mm$ time resolution [14]. The phantom was placed in two orientations – phantom cylinder axis along scanner z-axis and y-axis. 1 and 10 million detected coincidences scans were generated for each orientation.

These simulations have yielded NRMSE plot shown in Figure 7.10. 10 million represents an expected number of collected coincidences during real long medical PET examination. Therefore results for 10 million coincidences reflect the expected reconstruction image quality for future medical examinations performed using J-PET scanner. Result for 1 million coincidences was given to show what happens if MLEM method is given not enough statistics and can represent brief "live-preview" medical PET examination.

Altogether shown results represent the quality comparable to commercial PET devices available on the market [33, 34, 36], which lets us believe that J-PET scanner may be a cheaper but comparably good alternative to expensive crystal PET scanners.

Element	**Dimensions**
Cylinder	
diameter	20.4 cm
height	18.6 cm
Rods and spheres	
rod diameters	4.8, 6.4, 7.9, 9.5, 11.1, 12.7 mm
rod height	8.8 cm
sphere diameters	9.5, 12.7, 15.9, 19.1, 25.4, 31.8 mm
spheres vertical position	12.7 cm

Table 7.4.: Deluxe Jaszczak Phantom specification according to [70, 71]



(a) proportional density **(b)** increased transparency

Figure 7.9.: Virtual phantom based on *Deluxe Jaszczak Phantom* 3D rendering

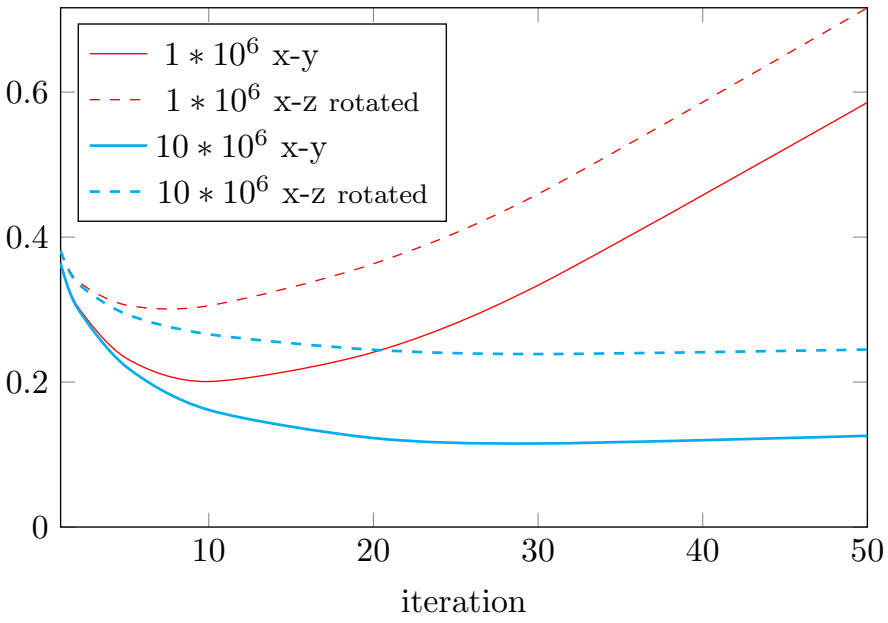
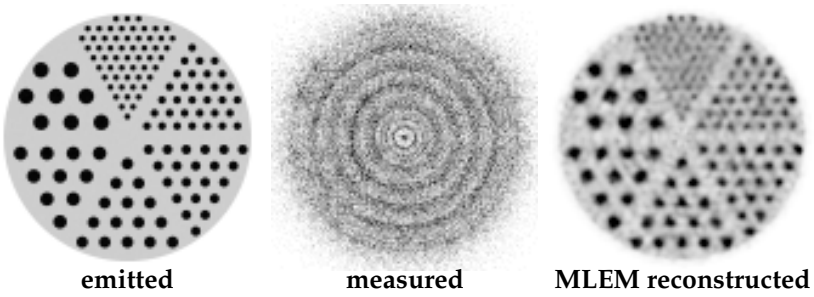
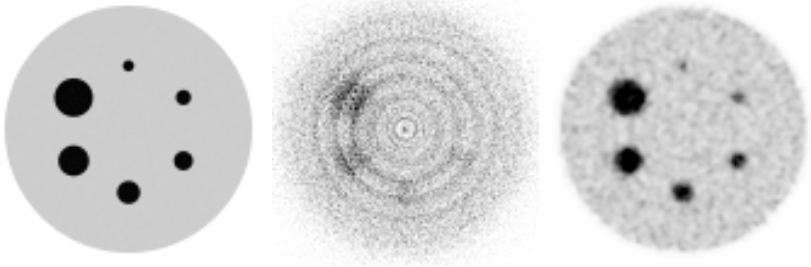


Figure 7.10.: J-PET "big" barrel 3D PET scanner NRMSE plot for Jaszczak phantom (NRMSE as a function of the number of MLEM iterations, for two numbers of coincidences and two phantom orientations)

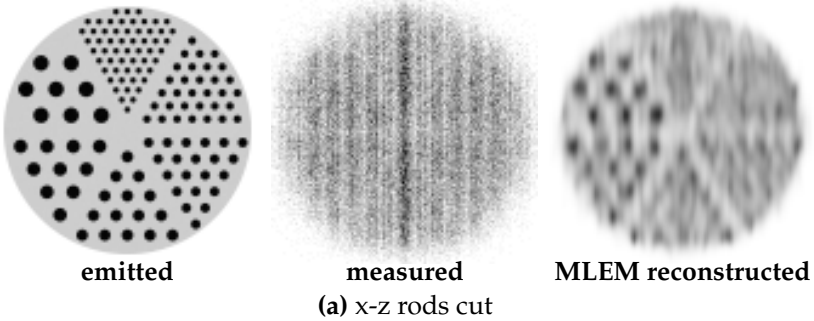


(a) x-y rods cut

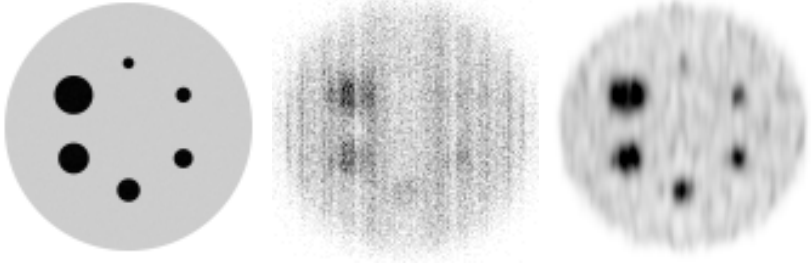


(b) x-y spheres cut

Figure 7.11.: 3D reconstruction results – $10 * 10^6$ coincidences, 20 iterations (phantom cylinder axis orientation along z-axis)

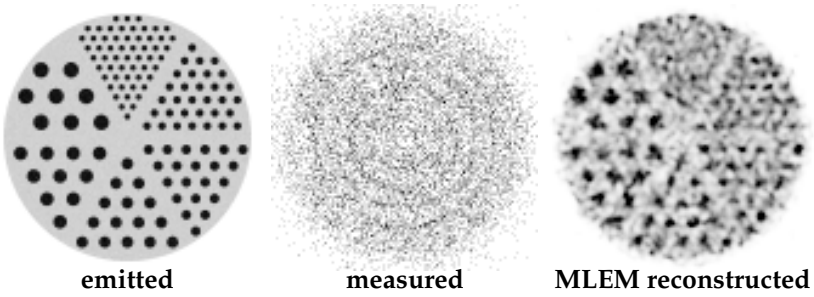


(a) x-z rods cut

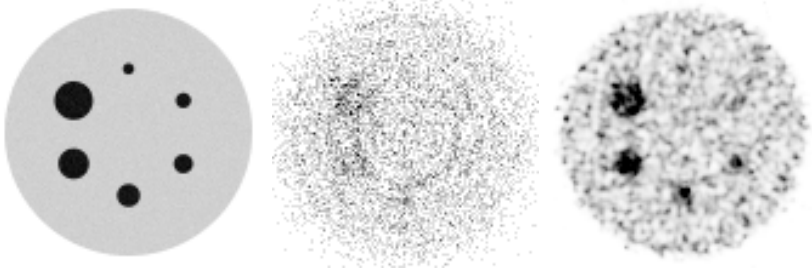


(b) x-z spheres cut

Figure 7.12.: 3D reconstruction results – $10 * 10^6$ coincidences, 20 iterations (phantom cylinder axis orientation along y-axis)

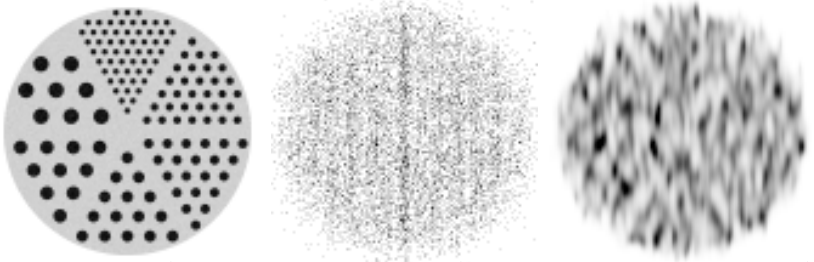


(a) x-y rods cut



(b) x-y spheres cut

Figure 7.13.: 3D reconstruction results – $1 * 10^6$ coincidences, 20 iterations (phantom cylinder axis orientation along z-axis)



(a) x-z rods cut

(b) x-z spheres cut

Figure 7.14.: 3D reconstruction results – $1 * 10^6$ coincidences, 20 iterations (phantom cylinder axis orientation along y-axis)

Chapter 8.

Conclusions and future directions

J-PET is a novel device not only in terms of use of plastic scintillator strips implying its unique geometry, but also in terms of modern electronics and at last, but not least, algorithms employing parallel computing techniques and GPGPU devices.

In the previous chapters of this dissertation, I presented several algorithms and techniques that form a foundation for image reconstruction software for J-PET scanner. These algorithms not only work and produce results with comparable quality to existing devices available on the market, but their performance makes them suitable for almost realtime on-site measurement visualization and analysis as they comply with the constraint of producing results below one minute. This makes J-PET device not only potentially cheaper and easier to manufacture, but also convenient to use.

8.1. Future directions

While all algorithms presented in this work and accompanying source code represent working proof-of-concept, they can be a subject for further improvements in terms of performance and functionality. First of all, the software tools produced by source code accompanying this work and described thoroughly in appendices, provide bare command line interface. Such interface is best for prototyping, analysis and research, but for production use it has to be coupled with some dedicated visualization modules.

Another future direction is exploring the usage of other computing devices for purpose of optimized simulation and image reconstruction. Currently optimized J-PET GPGPU code uses exclusively CUDA programming interface that is suitable

only for NVIDIA device family. CUDA was chosen intentionally as NVIDIA devices represent best *FLOPs/dollar* ratio today and CUDA interface is most stable, explored and reliable GPGPU interface. We made some initial attempts to employ other computing accelerators such as *Intel Xeon Phi*, but the initial performance results were nowhere close to NVIDIA devices. This comes either straight from NVIDIA devices highest FLOPs peak performance comparing to the others, but also from easy and straightforward CUDA programming paradigm. Other parallelization and vectorization techniques require either relying on compiler automatic optimizations or OpenMP constructs, which did not work very well in case of J-PET, or use of manual intrinsics which is laborious time-consuming process as we have learnt in ϕ^4 project [49]. CUDA implicit parallelization produces best results in shortest time span.

Porting the GPGPU source code to OpenCL standard, which follows same paradigm as CUDA and works for devices coming from other manufacturers, could be considered as well. Recent OpenCL 2.1 standard introduces support for C++ [72], which could allow to port our code easily to OpenCL. However, all current OpenCL implementations still do not support version 2.1 of the standard, while just a few support older version 2.0. Using older OpenCL implementation would require rewriting most of the code parts into plain C. This would be again very time-consuming process, that gives no guarantee that we receive better results using NVIDIA competing devices, such as AMD GPU cards. Therefore it makes sense to explore this direction once device manufacturers will provide complete support for OpenCL 2.1, that will let compile and run our code without many modifications on other GPGPU devices.

It has to be noted that we made some attempt to produce SPIR [73] binary code, runnable on OpenCL compatible devices, out of our C++ source code base using patched LLVM Clang compiler. This solution was however very fragile and produced code that worked only for some trivial input. The changes were also not accepted into Clang base, simply because at the time Khronos was working on supporting C++ officially in OpenCL.

Altogether, we believe that further evolution of programming interfaces, computing paradigms and device architectures in the upcoming years will make easier to exploit computing power available in massively parallel computing accelerators, that will drive further evolution of PET imaging and other medical imaging techniques. We hope this work can be inspiration for future works and provides good understanding for J-PET scanner and its image reconstruction methods.

Appendix A.

Accompanying project source code

The source code accompanying this dissertation is a result of over three year's work and represents evolution of the various ideas arose during the research on J-PET simulation and reconstruction. While the codebase is complex, we did our best to employ best programming techniques and good programming habits making the code readable and reusable to anyone.

All PET, image reconstruction, simulation and optimization algorithms were developed solely by the authors and were not copied or derived from any existing implementations.

A.1. Getting project source code from *Git* repository

This project source code was versioned from the very beginning using *Git* version control system (VCS) and it is hosted on Jagiellonian University server at

`https://sorbus.if.uj.edu.pl/pet/tools` [1]

The repository access is restricted, but will be provided to anyone willing to contribute to or review this work by requesting the access writing to author's email

`adam.strzelecki@uj.edu.pl`

[1] Will render as *404 Page Not Found* for not-logged user.

A.2. Build prerequisites and CUDA environment

This project was developed and tested on *Linux* and *Mac OS X*. It should compile and run on most *UNIX/POSIX* compatible platforms. This project may run with no or minimal modifications on *Microsoft Windows*, but no official support is provided for this platform.

This project does not use any third party libraries, especially no PET, image specific or code optimization libraries, with an exception for

- command line interface arguments parsing provided by *cmdline* minimalistic single header library [74],
- unit tests runner framework provided by *Catch* header only library [75],
- optional geometry description generation relying by *Boost* geometry library [76].

Minimal CPU-only configuration has following build prerequisites:

1. *UNIX* compatible build environment, tested on *Ubuntu 14.04 LTS Linux* and *Mac OS X 10.11*
2. C++11 compatible compiler i.e. *GCC* 4.6, *Clang* 3.2 or *ICC* 13
3. *CMake* [77] 2.8 for build script generation
4. *GNU Make* 3.8 for building using `Makefile`

Optional prerequisites, e.g. required for GPGPU optimized modules:

1. CUDA 7.5 (automatically detected by `cmake`), necessary for GPGPU modules
2. *Ninja* 1.4 for faster re-builds (with `cmake -G Ninja`)
3. `libpng` headers and libraries for PNG output
4. *Boost* 1.58 for optional geometry calculation
5. *QtCreator* 3.1 programming environment for comfortable development

A.3. Preparing and building project using *CMake*

PET tools project relies on *CMake* [77] for platform independent build process bootstrap. In most of the cases it is sufficient to execute `cmake` within the root of the project to generate platform specific build files.

```
-- The CXX compiler identification is Clang 3.7.0
-- Check for working CXX compiler using: Ninja
-- Check for working CXX compiler using: Ninja -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Found OpenMP: -fopenmp=libomp
-- OpenMP found
-- Found CUDA: /usr/local/cuda (found version "7.5")
-- The C compiler identification is AppleClang 7.0.2.7000181
-- Found ZLIB: /usr/lib/libz.dylib (found version "1.2.5")
-- Found PNG: /usr/local/lib/libpng.a (found version "1.5.23")
-- Boost version: 1.58.0
-- Disabled CACHE_ELLIPSE_PIXELS
-- Using WARP GRANULARITY
-- Using MAX_PIXELS_PER_THREAD=12
-- Using MAX_THREADS_PER_BLOCK=512
-- Disabled NORMAL_PHANTOM
-- Enabled USE_KERNEL
-- Disabled USE_RHO_PER_WARP
-- Disabled USE_SENSITIVITY
-- Disabled USE_STATISTICS
-- Using WARP_SIZE=32
-- Configuring done
-- Generating done
-- Build files have been written to: /Users/UJ/Projects/PET
```

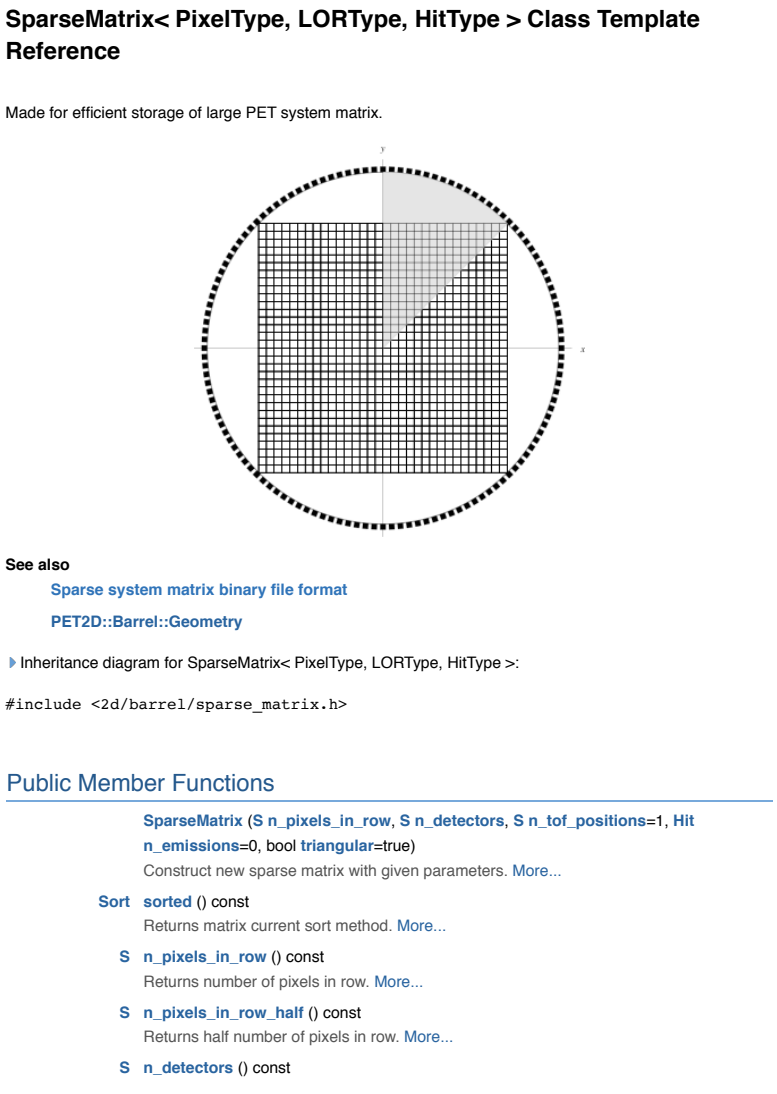
Figure A.1.: Example CMake output for successful PET tools build configuration

Once CMake succeeds setting up build process – all prerequisites are properly found and resolved, `Makefile` file will be written by default in UNIX environment. In order to perform actual build process `make` command have to be executed in project root.

Aside default CMake customization options, PET tools define their own compilation options. For best performance these options should not be altered.

A.4. Producing and reading *Doxygen* documentation

Source code files, C++ classes, namespaces and built commands are extensively documented using *Doxygen* [78] tool and inline comments. The documentation can be built by running `doxygen` command in the root of the PET tools project tree, and then read by opening `doc/html/index.html` file in the web browser.



SparseMatrix< PixelType, LORType, HitType > Class Template Reference

Made for efficient storage of large PET system matrix.

See also

Sparse system matrix binary file format

PET2D::Barrel::Geometry

Inheritance diagram for SparseMatrix< PixelType, LORType, HitType >:

`#include <2d/barrel/sparse_matrix.h>`

Public Member Functions

SparseMatrix (S n_pixels_in_row, S n_detectors, S n_tof_positions=1, Hit n_emissions=0, bool triangular=true)
Construct new sparse matrix with given parameters. More...

Sort sorted () const
Returns matrix current sort method. More...

S n_pixels_in_row () const
Returns number of pixels in row. More...

S n_pixels_in_row_half () const
Returns half number of pixels in row. More...

S n_detectors () const

Figure A.2.: Example Doxygen documentation page for `SparseMatrix` class

A.5. Running tests

J-PET tools project uses unit tests via *Catch* framework [75] in order to provide high quality valid code. In order to run tests `make test` has to be built, then `./test` command has to be run to perform tests.

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
test is a Catch v1.3.1 host application.
Run with -? for options

========================================================
All tests passed (61372 assertions in 106 test cases)
```

Figure A.3.: Output of successful `test` execution

A.6. PET tools commands

Brief description of PET tools commands compiled during the build process is provided below for the reader's comfort. Detailed documentation for these commands can be found in Doxygen generated documentation.

All commands provide in-place help when supplied with `-?` or `-help` command line option. Simulation or reconstruction commands provide additional `-G` or `-gpu` option to run in GPU accelerated mode.

```
usage: ./3d_hybrid_matrix [options] ... matrix_file ...
build: OpenMP/CUDA WARP
note: All length options below should be expressed in meters.
options:
  -c, --config          load config file (file [=])
  -n, --n-pixels        number of pixels in one dimension (int [=256])
  -p, --s-pixel         pixel size (float [=auto])
  -s, --shape           detector (scintillator) shape (square, circle, ...
  -w, --w-detector      detector width (float [=auto])
  -h, --h-detector      detector height (float [=auto])
      --d-detector      inscribe detector shape into circle of ...
```

Figure A.4.: Partial `3d_hybrid_matrix -help` output

A.6.1. List of commands

2d_barrel_geometry *(optional)* generates precise geometry description file, used to calculate analytic P (3.5) representation and iterate through TOR non-zero P pixels.

This command require *Boost* library to be present in the system.

2d_barrel_lm_reconstruction performs 2D barrel list-mode reconstruction. This command is essentially list-mode specific re-implementation of bin-mode reconstruction described in Chapter 5.

2d_barrel_matrix performs 2D barrel system matrix simulation described in Chapter 4. This file is used by **2d_barrel_reconstruction**.

2d_barrel_phantom performs 2D barrel phantom simulation described in Chapter 4 producing an input response file for **2d_barrel_reconstruction**.

2d_barrel_reconstruction performs 2D barrel bin-mode reconstruction described in Chapter 5 from given bin-mode response file and system matrix file.

2d_strip_phantom performs 2D strip phantom simulation from given textual phantom description generating list-mode response file that can be used to perform reconstruction using **2d_strip_reconstruction**.

2d_strip_reconstruction performs 2D strip reconstruction as described in Chapter 6 using given response file.

3d_hybrid_matrix performs 3D slice system matrix simulation as described in Chapter 7 for given z position.

3d_hybrid_phantom performs 3D phantom simulation described in Chapter 7 for given *JSON* phantom description generating list-mode response file.

3d_hybrid_reconstruction performs full 3D reconstruction for J-PET scanner as described in Chapter 7 from list-mode response file using given geometry file and optionally system matrix file slice or multiple slices.

3d_tool_crop 3D raw image crop tool

3d_tool_psf 3D PSF FWHM calculation tool

A.6.2. List of commands not intended for general use

2d_barrel_format_converter converts general 3D position plus time into TOR index plus TOF position delta. Used to convert external simulation data.

2d_strip_kernel_monte_carlo simulates 2D strip P, equivalent of **2d_barrel_matrix**. Meant to compare 2D strip P analytic approximation quality.

3d_hybrid_sensitivity performs 3D scanner sensitivity simulation and outputs the 3D sensitivity map. (CPU only)

A.7. Source code structure

cmake contains extra CMake modules.

doc present when `doxygen` is run, contains documentation.

lib contains third-party libraries.

math contains *Mathematica* modules.

phantoms contains example phantom description files.

scripts contains helper (shell) scripts.

src contains C++ source code.

- **common** common classes for 2D and 3D.
- **2d** 2D generic simulation and reconstruction classes and commands.
  - **geometry** 2D generic geometry classes.
  - **barrel** 2D barrel simulation and reconstruction.
    - **cuda** CUDA (GPU) specific code.
  - **strip** 2D strip simulation and reconstruction.
    - **cuda** CUDA (GPU) specific code.
- **3d** 3D hybrid classes and commands.
  - **geometry** 3D generic geometry classes.
  - **hybrid** 3D hybrid simulation and reconstruction.
    - **cuda** CUDA (GPU) specific code.
- **util** utility classes and helpers.

Appendix B.

Result visualization and analysis tools

Some important results and scripts that were performed using PET tools can be found in a separate repository at

`https://sorbus.if.uj.edu.pl/pet/data` [1]

B.1. PSF FWHM and NRMSE calculation tools

J-PET tool project provides two important analysis tools – **3d_tool_psf** for calculating PSF FWHM described in Section 3.6.2 and **3d_tool_nrmse** for calculating NRMSE described in Section 3.6.3.

While these measures can be calculated by other tools, such as *Mathematica*, the built-in tools were optimized to calculate the measures for many image reconstruction iterations and output textual data that can be later used for plots such as Figure 7.5 presented in Section 7.5.3.

B.2. Using *Mathematica* for visualization and analysis

Mathematica [79] is not only computer-algebra system (CAS), but also great interactive visualization and analysis environment with its own language. Many analysis and plots presented in this work were done using Mathematica 10, and are available as *notebooks* – files with `.nb` extension in data repository.

[1] Will render as *404 Page Not Found* for not-logged user.

B.3. System matrix visualization

J-PET tools contains **math** subdirectory containing many basic notebooks and some Mathematica modules. Most important is **2d_barrel_matrix.m** defining

ReadPETMatrix function reading sparse matrix file from given file into optimized Mathematica representation.

ImagePETMatrix plotting result of *ReadPETMatrix*, either as sensitivity map or TOR plot with optional second argument containing list of TOR indices.

B.4. Reconstruction visualization and quality analysis

Mathematica notebook file `data/201510_3d_sensitivity/analyze.nb` can serve as an example of how to make an analysis of PET simulation output system matrix file. As shown in Figure B.1 below, Mathematica was used to visualize interactively the sensitivity of J-PET "big" barrel scanner in $x - y$ plane depending on position along z axis of the scanner.

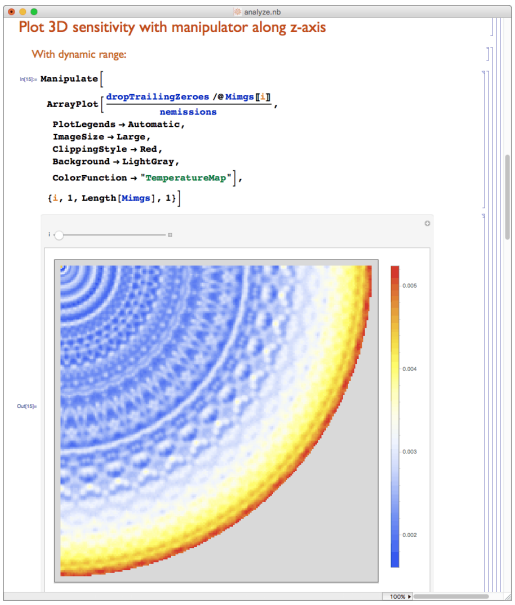


Figure B.1.: `data/201510_3d_sensitivity/analyze.nb` notebook

B.5. Other third party tools for visualization

2D reconstruction and simulation tools output `.png` greyscale files together with regular output. These files can be displayed using any image viewer. Generated PNG files are 8-bit, thus are normalized to $0 - 255$ range, where data minimum value is mapped to 0 and maximum is mapped to 255.

3D reconstruction and simulation tools output RAW binary files together with `.nrrd` (*Nearly Raw Raster Data*) textual description files. These files can be open by several analysis and visualization programs supporting NRRD format described in detail by [80] or any applications capable of reading floating point binary data. The major advantage of NRRD and RAW binary file tandem is that it stores unnormalized and un-quantized exact results. So in case of the image reconstruction output, voxel values represent unaltered number of emissions estimates.

B.6. *ParaView* for 3D imagery visualization and analysis

ParaView [81] is an advanced open-source visualization application. It supports many input formats including RAW volumetric data. It can also read NRRD files, that contain necessary information to load accompanying binary volumetric data.

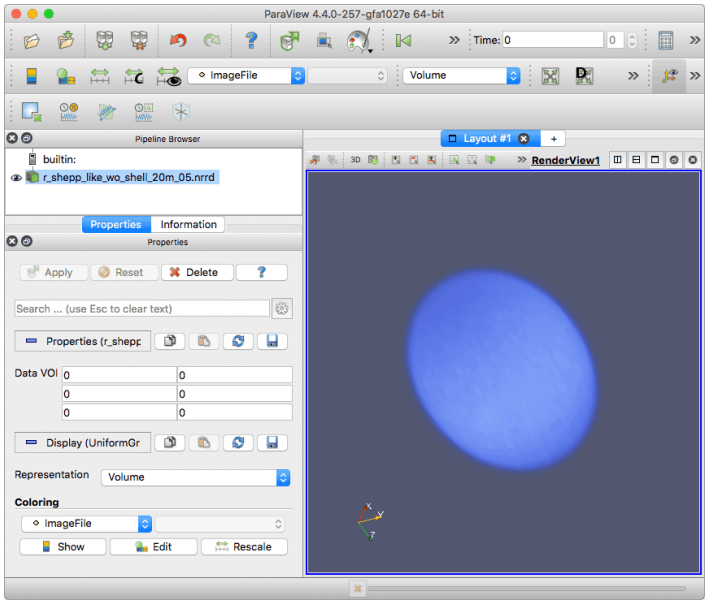


Figure B.2.: ParaView 3D reconstruction result volumetric display

B.7. *MRIcro* simple and 3D visualization tool

MRIcro [82] is recommended to anyone looking for some lightweight application for 3D visualization. It comes in two flavors – *MRIcro for OS X* which is OS X specific version and *MRIcroGL* which is OpenGL multi-platform implementation.

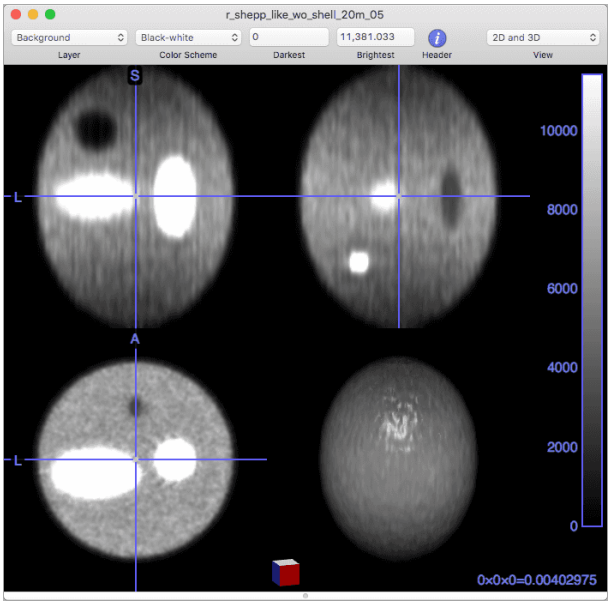


Figure B.3.: MRIcro for OS X visualizing 3D reconstruction result from `.nrrd` file

MRIcro supports natively NIfTI format, and offers automatic converter from DICOM used widely in Europe for computed tomography. It can also read NRRD files generated by J-PET tools.

Glossary

PET *positron emission tomography* - imaging technique for PET tomography device or PET scanner detecting gamma radiation coming from annihilation of positrons (anti-electrons).

J-PET *Jagiellonian PET* - PET scanner using polymer plastic scintillators, built at Jagiellonian University in Krakow, subject of this work.

scintillator - element absorbing radiation and emitting visible light.

photomultiplier - element detecting light (arriving photons) as voltage changes on electrical connectors.

scintillator detector or *PET detector* - system of scintillator and one or more photomultipliers.

time slot or *time window* - time period relative to reference clock.

coincidence - appearance of signal on given pair of detectors in a single time slot.

response - readout of coincidence measurement.

event - unobserved directly, but indirectly through PET device, emission of two quanta gamma.

TOR *tube of response*, alternatively **LOR**, *line of response* - scanner response space element, described by pair of detectors and space spanned between them. Given response belongs to specific TOR if it is a measurement of detected coincidence on the detector pair spanning this TOR.

TOF *time of flight* - additional time information, usually a time difference of signal detection on photomultipliers pair. Can be used to estimate more precisely emission position along TOR.

ML *maximum likelihood* - measure used to evaluate agreement between the candidate statistical model parameters, e.g. emission density image, and data-incomplete observation, e.g. PET scan, according to specific statistical model [54].

EM *expectation maximization*, **MLEM**, *maximum likelihood expectation maximization* - specific iterative method (algorithm) of estimating maximum likelihood statistical model parameters [51–53].

LM *list mode* - representation for PET scan data, carrying detailed information for each response, unlike simpler representations used for PET imaging based on event counting, e.g. using binning (histograms).

phantom or *imaging phantom* - reference object used for scanning, imaging or calibrating tomography devices, used as substitute for live or cadaver subjects.

pixel - element of 2D discrete image space.

voxel - element of 3D discrete image space.

CPU *central processing unit* - main processing unit (processor) of computer or computing device, responsible for executing operating system and general purpose operations.

GPU *graphics processing units* - specialized processing unit (processor) used for generation and processing of computer graphics, usually emplaced on dedicated add-on graphics card.

GPGPU *general-purpose computing on graphics processing units* - general-purpose computation carried on GPU. GPUs were not designed for GPGPU in the first place, but were adapted via specialized programming interface.

SIMD *single instruction multiple data* - computational model and computer architecture where single machine instruction can operate on multiple operands (data points) simultaneously, so in a single cycle computer can perform multiple operations, such as additions or multiplications. However, it is not possible to perform two different operations at once, in opposition to parallel computing.

parallel algorithm - algorithm executed on multiple independent computing units, usually made of several independent computing paths.

vector algorithm - algorithm performing computations using SIMD vector instructions.

FLOP *floating point operation* - operation or computer instruction operating on floating point number.

FLOPS *floating point operations per second* - measure of processor performance expressed as a maximum number of floating point operations the processor is capable of executing within a second.

memory bandwidth - measure of memory throughput performance expressed as a maximum number of bytes that can be transferred from or to memory within a second.

OpenMP - programming interface standard for performing parallel computation on CPU.

CUDA - programming interface for performing GPGPU on NVIDIA GPU devices.

SIMT *single instruction multiple threads* - computational model specific to NVIDIA GPU devices, similar to SIMD, but from the programmer's perspective using scalar instructions and so called *CUDA threads*, that are implicitly translated into SIMD vector elements and vector instructions of GPU.

Xeon Phi - family of Intel computational accelerators based on well known *x86* architecture, using special extended 512-bit vector registers and instructions, capable of carrying 16 single precision floating point numbers.

Bibliography

[1] Harrison H Barrett, Timothy White, and Lucas C Parra. "List-mode likelihood". In: *Journal of the American Statistical Association* 14.11 (1997), pp. 2914–2923.

[2] Paweł Moskal, Neha Gupta-Sharma, Michał Silarski, ..., Adam Strzelecki, et al. "Hit time and hit position reconstruction in the J-PET detector based on a library of averaged model signals". In: *Acta Phys. Polon. A* 127 (Feb. 2015), pp. 1495–1499.

[3] GL Brownell, CA Burnham, B Hoop Jr, and DE Bohning. "Quantitative dynamic studies using short-lived radioisotopes and positron detection". In: *Dynamic Studies with Radioisotopes in Medicine. Proceedings of the Symposium on Dynamics Studies with Radioisotopes in Clinical Medicine and Research*. 1971.

[4] ZH Cho, L Eriksson, and JK Chan. "A circular ring transverse axial positron camera". In: *International Journal of Nuclear Medicine and Biology* 3.3 (1976), pp. 165–166.

[5] T Ido, C-N Wan, V Casella, JS Fowler, AP Wolf, M Reivich, and DE Kuhl. "Labeled 2-deoxy-D-glucose analogs. 18F-labeled 2-deoxy-2-fluoro-D-glucose, 2-deoxy-2-fluoro-D-mannose and 14C-2-deoxy-2-fluoro-D-glucose". In: *Journal of Labelled Compounds and Radiopharmaceuticals* 14.2 (1978), pp. 175–183.

[6] Alejandro Sánchez-Crespo, Pedro Andreo, and Stig A Larsson. "Positron flight in human tissues and its influence on PET image spatial resolution". In: *European journal of nuclear medicine and molecular imaging* 31.1 (2004), pp. 44–51.

[7] Konrad Szymański, Paweł Moskal, ..., Adam Strzelecki, et al. "Simulations of γ quanta scattering in a single module of the J-PET detector". In: *Bio-Algorithms and Med-Systems* 10.2 (2014), pp. 71–77.

[8] Paweł Kowalski, Paweł Moskal, Wojciech Wiślicki, Lech Raczyński, ..., Adam Strzelecki, et al. "Multiple scattering and accidental coincidences in the J-PET detector simulated using GATE package". In: *Acta Phys. Polon. A* 127 (Feb. 2015), pp. 1505–1512.

[9] Paweł Moskal. "Strip device and the method for the determination of the place and response time of the gamma quanta and the application of the device for the Positron Emission Thomography". Pat. No. US20120112079, PL388555, JP2012533734, EP2454612. 2009. URL: https://www.google.com/patents/US20120112079.

[10] Paweł Moskal, Piotr Salabura, Michał Silarski, Jerzy Smyrski, Jarosław Zdebik, and Marcin Zieliński. "Novel detector systems for the Positron Emission Tomography". In: *Bio-Algorithms and Med-Systems* 7.2 (2011), p. 73.

[11] Paweł Moskal et al. "Strip-PET: a novel detector concept for the TOF-PET scanner". In: *Nuclear Medicine Review* 15.Suppl. C (2012), pp. C68–C69.

[12] Paweł Moskal et al. "TOF-PET detector concept based on organic scintillators". In: *Nuclear Medicine Review* 15.Suppl. C (2012), pp. C81–C84.

[13] Paweł Moskal et al. "Polymer scintillator detectors for TOF-PET with large longitudinal field of view". In: *Nuclear Medicine Review* 15 (2012).

[14] Paweł Moskal, Oleksandr Rundel, ..., Adam Strzelecki, et al. "Time resolution of the plastic scintillator strips with matrix photomultiplier readout for J-PET tomograph". In: *Physics in medicine and biology* 61.5 (2016), p. 2025.

[15] Nikodem Krawczyk. "Study of possibility of simultaneous usage of J-PET and CT tomographs". In Polish. BA thesis. Jagiellonian University in Krakow, 2015.

[16] Grzegorz Korcyl, Paweł Moskal, ..., Adam Strzelecki, et al. "Trigger-less and reconfigurable data acquisition system for positron emission tomography". In: *Bio-Algorithms and Med-Systems* 10.1 (2014), pp. 37–40.

[17] Grzegorz Korcyl. "A Novel Data Acquisition System Based on Fast Optical Links and Universal Readout Boards". PhD thesis. AGH University of Science and Technology, 2015.

[18] Marek Pałka, ..., Adam Strzelecki, et al. "A novel method based solely on FPGA units enabling measurement of time and charge of analog signals in Positron Emission Tomography". In: *Bio-Algorithms and Med-Systems* 10.1 (2014), pp. 41–45.

[19] Paweł Moskal, Natalia Zoń, ..., Adam Strzelecki, et al. "A novel method for the line-of-response and time-of-flight reconstruction in TOF-PET detectors based on a library of synchronized model signals". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 775 (2015), pp. 54–62.

[20] L.A. Shepp and Y. Vardi. "Maximum Likelihood Reconstruction for Emission Tomography". In: *Medical Imaging, IEEE Transactions* 1.2 (Oct. 1982), pp. 113–122. ISSN: 0278-0062.

[21] L.A. Shepp and Y. Vardi. "A Statistical Model for Positron Emission Tomography". In: *Journal of the American Statistical Association* 80.389 (Mar. 1985).

[22] Y Picard, V Selivanov, M Verreault, and R Lecomte. "Optimizing communications for parallel ML-EM image reconstruction on large clusters of processors". In: *Nuclear Science Symposium, 1998. Conference Record. 1998 IEEE*. Vol. 3. IEEE. 1998, pp. 1574–1580.

[23] IK Hong, ST Chung, HK Kim, YB Kim, YD Son, and ZH Cho. "Fast forward projection and backward projection algorithm using SIMD". In: *Nuclear Science Symposium Conference Record, 2006. IEEE*. Vol. 6. IEEE. 2006, pp. 3361–3368.

[24] Guillem Pratx, Garry Chinn, Peter D Olcott, and Craig S Levin. "Fast, accurate and shift-varying line projections for iterative reconstruction using the GPU". In: *Medical Imaging, IEEE Transactions on* 28.3 (2009), pp. 435–445.

[25] JL Herraiz, S Espaa, S García, R Cabido, AS Montemayor, Manuel Desco, Juan José Vaquero, and José Manuel Udías. "GPU acceleration of a fully 3D iterative reconstruction software for PET using CUDA". In: *Nuclear Science Symposium Conference Record (NSS/MIC), 2009 IEEE*. IEEE. 2009, pp. 4064–4067.

[26] JL Herraiz, Samuel España, Jacobo Cal-González, Juan José Vaquero, Manuel Desco, and José Manuel Udías. "Fully 3D GPU PET reconstruction". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 648 (2011), S169–S171.

[27] Jing-yu Cui, Guillem Pratx, Sven Prevrhal, and Craig S Levin. "Fully 3D list-mode time-of-flight PET image reconstruction on GPUs using CUDA". In: *Medical physics* 38.12 (2011), pp. 6775–6786.

[28] Guillem Pratx and Lei Xing. "GPU computing in medical physics: A review". In: *Medical physics* 38.5 (2011), pp. 2685–2697.

[29] Guillem Pratx, Suleman Surti, and Craig Levin. "Fast list-mode reconstruction for time-of-flight PET using graphics hardware". In: *Nuclear Science, IEEE Transactions on* 58.1 (2011), pp. 105–109.

[30] Guillem Pratx, Jing-Yu Cui, Sven Prevrhal, and Craig S Levin. "3-D tomographic image reconstruction from randomly ordered lines with CUDA". In: Morgan Kaufmann, NVIDIA, 2011, pp. 679–691.

[31] Jingyu Cui, Guillem Pratx, Sven Prevrhal, Bin Zhang, Lingxiong Shao, and Craig S Levin. "Measurement-based spatially-varying point spread function for list-mode PET reconstruction on GPU". In: *Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC), 2011 IEEE*. IEEE. 2011, pp. 2593–2596.

[32] S Ha, S Matej, M Ispiryan, and K Mueller. "GPU-accelerated forward and back-projections with spatially varying kernels for 3D DIRECT TOF PET reconstruction". In: *IEEE transactions on nuclear science* 60.1 (2013), p. 166.

[33] Suleman Surti, Austin Kuhn, Matthew E Werner, Amy E Perkins, Jeffrey Kolthammer, and Joel S Karp. "Performance of Philips Gemini TF PET/CT scanner with special consideration for its time-of-flight imaging capabilities". In: *Journal of Nuclear Medicine* 48.3 (2007), pp. 471–480.

[34] Joel S Karp, Suleman Surti, Margaret E Daube-Witherspoon, and Gerd Muehllehner. "Benefit of time-of-flight in PET: experimental and clinical results". In: *Journal of Nuclear Medicine* 49.3 (2008), pp. 462–470.

[35] JS Karp, S Surti, ME Daube-Witherspoon, CR Divgi, and AE Perkins. "Clinical benefits of Time-of-Flight in PET imaging". In: *MEDICAMUNDI* 52.1 (2008), p. 07.

[36] BW Jakoby, Y Bercier, M Conti, ME Casey, B Bendriem, and DW Townsend. "Physical and clinical performance of the mCT time-of-flight PET/CT scanner". In: *Physics in medicine and biology* 56.8 (2011), p. 2375.

[37] Sriram Krishnamoorthy, Benjamin LeGeyt, Matthew E Werner, Manohar Kaul, FM Newcomer, Joel S Karp, and Suleman Surti. "Design and performance of a high spatial resolution, time-of-flight PET detector". In: *Nuclear Science, IEEE Transactions on* 61.3 (2014), pp. 1092–1098.

[38] Nicolas A Karakatsanis, Martin A Lodge, Arman Rahmim, and Habib Zaidi. "Introducing time-of-flight and resolution recovery image reconstruction to clinical whole-body PET parametric imaging". In: *IEEE Nucl. Sci. Symp. & Medical Imaging Conf.(NSS/MIC) 2014*. 2014.

[39] BC LeGeyt, WJ Ashmanskas, J Dormer, and S Krishnamoorthy. "ROCSTAR: Data Acquisition Electronics for TOF PET". In: (2014).

[40] *TOP 500 Lists November 2015*. 2015. URL: http://www.top500.org/lists/2015/11/.

[41] Samuel Williams, Andrew Waterman, and David Patterson. "Roofline: an insightful visual performance model for multicore architectures". In: *Communications of the ACM* 52.4 (2009), pp. 65–76.

[42] Tianyi David Han and Tarek S Abdelrahman. "Reducing branch divergence in GPU programs". In: *Proceedings of the Fourth Workshop on General Purpose Processing on Graphics Processing Units*. ACM. 2011, p. 3.

[43] Michael J Flynn. "Some computer organizations and their effectiveness". In: *Computers, IEEE Transactions on* 100.9 (1972), pp. 948–960.

[44] Dean M Tullsen, Susan J Eggers, and Henry M Levy. "Simultaneous multithreading: Maximizing on-chip parallelism". In: *ACM SIGARCH Computer Architecture News*. Vol. 23. 2. ACM. 1995, pp. 392–403.

[45] Susan J Eggers, Joel S Emer, Henry M Leby, Jack L Lo, Rebecca L Stamm, and Dean M Tullsen. "Simultaneous multithreading: A platform for next-generation processors". In: *Micro, IEEE* 17.5 (1997), pp. 12–19.

[46] Piotr Bialas and Adam Strzelecki. "Benchmarking the cost of thread divergence in CUDA". In: *Parallel Processing and Applied Mathematics*. 11th International Conference, PPAM 2015, Krakow, Poland, September 6-9, 2015. Revised Selected Papers. Springer Berlin Heidelberg, 2016.

[47] Cedric Bastoul. "Code generation in the polyhedral model is easier than you think". In: *Proceedings of the 13th International Conference on Parallel Architectures and Compilation Techniques*. IEEE Computer Society. 2004, pp. 7–16.

[48] Cédric Bastoul. "Generating loops for scanning polyhedra: Cloog user's guide". In: *Polyhedron* 2 (2004), p. 10.

[49] Piotr Białas, Jakub Kowal, and Adam Strzelecki. "GPU-Accelerated and CPU SIMD optimized Monte-Carlo simulation of ϕ^4 Model". English. In: *Facing the Multicore-Challenge III*. Ed. by Rainer Keller, David Kramer, and Jan-Philipp Weiss. Vol. 7686. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, pp. 131–132. ISBN: 978-3-642-35892-0.

[50] Cliff X Wang, Wesley E Snyder, Griff Bilbro, and Pete Santago. "Performance evaluation of filtered backprojection reconstruction and iterative reconstruction methods for PET images". In: *Computers in biology and medicine* 28.1 (1998), pp. 13–25.

[51] Arthur P Dempster, Nan M Laird, and Donald B Rubin. "Maximum likelihood from incomplete data via the EM algorithm". In: *Journal of the royal statistical society. Series B (methodological)* (1977), pp. 1–38.

[52] Sean Borman. "The expectation maximization algorithm-a short tutorial". In: *submitted for publication* (2004), pp. 1–9.

[53] Geoffrey McLachlan and Thriyambakam Krishnan. *The EM algorithm and extensions*. John Wiley & Sons, 1996.

[54] Rolf Sundberg. "Maximum likelihood theory for incomplete data from an exponential family". In: *Scandinavian Journal of Statistics* (1974), pp. 49–58.

[55] L. Parra and H.H. Barrett. "List-mode likelihood: EM algorithm and image quality estimation demonstrated on 2-D PET". In: *Medical Imaging, IEEE Transactions on* 17.2 (Apr. 1998), pp. 228–235. ISSN: 0278-0062.

[56] Guillem Pratx and Craig Levin. "Online detector response calculations for high-resolution PET image reconstruction". In: *Physics in medicine and biology* 56.13 (2011), p. 4023.

[57] National Electrical Manufacturers Association et al. *NEMA NU 2: Performance measurements of positron emission tomographs*. National Electrical Manufacturers Association, 2012. URL: https://www.nema.org/Standards/Pages/Performance-Measurements-of-Positron-Emission-Tomographs.aspx.

[58] S Jan, G Santin, D Strul, Steven Staelens, K Assie, D Autret, S Avner, R Barbier, M Bardies, PM Bloomfield, et al. "GATE: a simulation toolkit for PET and SPECT". In: *Physics in medicine and biology* 49.19 (2004), p. 4543.

[59] Paweł Kowalski, Wojciech Wiślicki, Lech Raczyński, ..., Adam Strzelecki, et al. "Scatter fraction of the J-PET tomography scanner". In: *Acta Phys. Polon. B* 47 (2016), p. 549.

[60] Piotr Białas, Jakub Kowal, Adam Strzelecki, et al. "GPU based Monte-Carlo simulations of PET system matrix". In: *KUKDM*. 2014.

[61] Linda Kaufman. "Implementing and accelerating the EM algorithm for positron emission tomography". In: *Medical Imaging, IEEE Transactions on* 6.1 (1987), pp. 37–51.

[62] Volker Gaede and Oliver Günther. "Multidimensional access methods". In: *ACM Computing Surveys (CSUR)* 30.2 (1998), pp. 170–231.

[63] Michael Doggett. "Texture caches". In: *IEEE Micro* 3 (2012), pp. 136–141.

[64] Piotr Białas, Jakub Kowal, Adam Strzelecki, et al. "System Response Kernel Calculation for List-mode Reconstruction in Strip PET Detector". In: *Acta Phys. Polon. B* Proceed. Suppl. 6 (Oct. 2013), pp. 1027–1036.

[65] Piotr Białas, Jakub Kowal, Adam Strzelecki, et al. "List-mode reconstruction in 2D strip PET". In: *Bio-Algorithms and Med-Systems* 10.1 (2014), pp. 9–12.

[66] Piotr Białas, Jakub Kowal, Adam Strzelecki, et al. "GPU accelerated image reconstruction in a two-strip J-PET tomograph". In: *Acta Phys. Polon. A* 127 (Feb. 2015), pp. 1500–1504.

[67] Mark Harris et al. "Optimizing parallel reduction in CUDA". In: *NVIDIA Developer Technology* 2.4 (2007).

[68] Joshua Schaefferkoetter, Jinsong Ouyang, Yothin Rakvongthai, Carmela Nappi, and Georges El Fakhri. "Effect of time-of-flight and point spread function modeling on detectability of myocardial defects in PET". In: *Medical physics* 41.6 (2014), p. 062502.

[69] Ewelina Kubicz, Szymon Niedźwiecki, Paweł Moskal, ..., Adam Strzelecki, et al. "Characterization of the spatial resolution of the J-PET detector". In: *Warsaw Medical Physics Meeting*. 2015.

[70] Ronald J Jaszczak, Kim L Greer, Carey E Floyd Jr, C Craig Harris, and R Edward Coleman. "Improved SPECT quantification using compensation for scattered photons." In: *Journal of nuclear medicine: official publication, Society of Nuclear Medicine* 25.8 (1984), pp. 893–900.

[71] Data Spectrum Corporation. *Flanged Jaszczak ECT Phantoms*. 2007. URL: http://www.spect.com/pub/Flanged_Jaszczak_Phantoms.pdf.

[72] Khronos OpenCL Working Group and Aaftab Munshi. *The OpenCL C++ Specification*. 2015. URL: https://www.khronos.org/registry/cl/specs/opencl-2.1-openclc++.pdf.

[73] Khronos Group and Boaz Ouriel. *The SPIR Specification*. 2014. URL: https://www.khronos.org/registry/spir/specs/spir_spec-2.0.pdf.

[74] Hideyuki Tanaka. *cmdline: A simple command line parser for C++*. 2011. URL: https://github.com/tanakh/cmdline.

[75] Phil Nash. *Catch: A modern, C++-native, header-only, framework for unit-tests, TDD and BDD*. 2015. URL: https://github.com/philsquared/Catch.

[76] Beman Dawes, David Abrahams, and Rene Rivera. *Boost: free peer-reviewed portable C++ source libraries*. 2015. URL: http://www.boost.org.

[77] Ken Martin and Bill Hoffman. *Mastering CMake*. Kitware, 2010. URL: http://www.cmake.org.

[78] Dimitri van Heesch. *Doxygen: Source code documentation generator tool*. 2008. URL: http://www.doxygen.org.

[79] Wolfram Research. *Mathematica: Computer algebra system (CAS)*. 2015. URL: http://www.wolfram.com/mathematica/.

[80] Teem developers. *Nearly Raw Raster Data*. 2009. URL: http://teem.sourceforge.net/nrrd/.

[81] Kitware. *ParaView: An open-source, multi-platform data analysis and visualization application*. 2015. URL: http://www.paraview.org.

[82] Chris Rorden. *MRIcro*. 2015. URL: http://www.mricro.com.

List of Algorithms

List of figures

List of tables