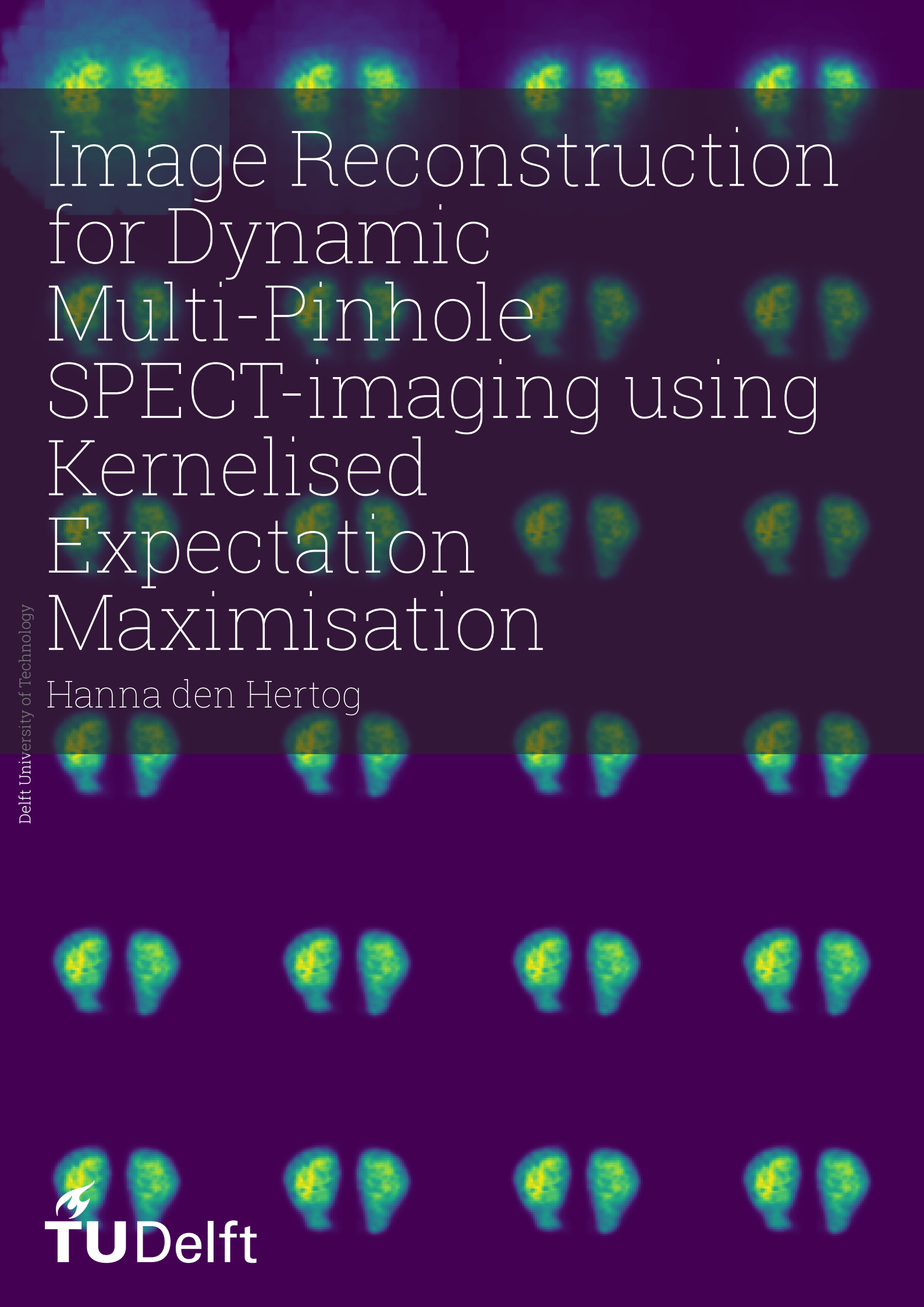


Image Reconstruction for Dynamic Multi-Pinhole SPECT-imaging using Kernelised Expectation Maximisation

Hanna den Hertog

Delft University of Technology

TU Delft

Image Reconstruction for Dynamic Multi-Pinhole SPECT-imaging using Kernelised Expectation Maximisation

by

Hanna den Hertog

Student Name	Student Number
H. den Hertog	5146461

Supervisors:	Dr. M.C. Goorden
	Dr. H.N. Kekkonen
Committee members:	Dr. Y. van Gennip
	Dr.ir. R. de Kruijff
Project Duration:	December 2022 – May 2023

TU Delft

Abstract

Background Dynamic SPECT scanning provides a non-invasive way to image the time-dependent distribution of radio-labelled tracers inside living tissue. Beside human medicine, dynamic SPECT also finds its applications in pre-clinical research on small animals. In pre-clinical research, multi-pinhole collimators are used to enable high-resolution sub-millimeter imaging. Conventional iterative reconstruction methods, such as Maximum Likelihood Expectation Maximisation (MLEM) perform poorly in reconstructing the noisy and low-count scans in dynamic SPECT. This limits the temporal resolution that can be achieved.

Method The reconstruction of noisy, low-count time-frames can be aided by incorporating information from earlier and later time-frames. Wang and Qi (2015) published the paper 'PET Image Reconstruction Using Kernel Method', with a proposed method entitled Kernelised Expectation Maximisation (KEM) for dynamic PET, a method that uses principles from Machine Learning, such as Support Vector Machines and the 'kernel trick' to incorporate prior information in the reconstruction algorithm. This method is highly adjustable due to a number of input parameters of the method. In this paper, KEM is implemented for dynamic multi-pinhole SPECT. The effects of the KEM parameters are explored in computer simulations. Two different dynamic phantoms are used, one of the striata in a mouse brain which were adapted from a paper by Vastenhouw et al. (2007) and one of the hepatobiliary system adapted from Vaissier et al. (2012). The results of KEM are benchmarked against conventional MLEM with a Gaussian post-filter.

Results In high-count simulations, the MLEM reconstructions had a lower mean-squared-error than the KEM image, while the signal-to-noise ratio of KEM was better than MLEM. The images produced after 200 iterations were indistinguishable, however. In the low-count regime, KEM was shown to be more resistant to noise than MLEM. Varying the input parameters of KEM gave rise to differences in performance, such as (over-)smoothing effects and a different level of noise-suppresion in the reconstructed image.

Conclusions In the simulations used in this paper, KEM was shown to outperform conventional MLEM with a Gaussian post-filter from low-count projections. The optimal input parameters of the KEM algorithm, however, need to be found ad hoc by searching the parameter space. Further research should look into finding a set of rules or guidelines for finding the optimal parameters.

Contents

Nomenclature

Abbreviations

Abbreviation	Definition
SPECT	Single-Photon Emission Computed Tomography
U-SPECT	Ultra-high resolution SPECT
KEM	Kernelised Expectation Maximisation
MLEM	Maximum Likelihood Expectation Maximisation
RBF	Radial Basis Function
kNN	k-Nearest Neighbours
PET	Positron Emission Tomography
PMT	Photo-multiplier tube
SVM	Support Vector Machine
DAT	Dopamine Transporter

Symbols

Symbol	Definition	Unit	Space/Domain
f	Activity vector	[Bq]	$\mathbb{R}^J$
g	Projection vector	[counts]	$\mathbb{R}^N$
i	Pixel index	[-]	$0, 1, \ldots, N$
j	Voxel index	[-]	$\mathbb{N}$
P	System matrix	[-]	$\mathbb{R}^{N \times J}$
K	Kernel matrix	[-]	$\mathbb{R}^{J \times J}$
κ	Kernel function	[-]	
ϕ_j	Feature vector of voxel j	[-]	$\mathbb{R}^B$

1 Introduction

The scientific field of medical imaging has made major advancements in the last centuries. Since the invention of the X-ray in 1895, it has grown to be a multi-billion dollar industry, with institutions and companies worldwide working on improving existing techniques and developing new ones[26]. Due to the non-invasive nature of these techniques, they are widely used for diagnosing diseases in various organs of the body, localising tumorous tissues and for conducting pre-clinical research on small animals.

Nuclear imaging is a sub-field of medical imaging which studies processes within organs and tissues though radioactive tracers that are administered to the subject in (preferably) small dosages. By performing multiple consecutive scans, one can even gain insight in the dynamics and interactions of the tracer as a function of time[21]. Examples of nuclear imaging include Positron Emission Tomography (PET) and Single Photon Emission Computed Tomography (SPECT). PET tracers are proton-rich isotopes which decay via positron emission. After emission, the positron will quickly encounter an electron along its trajectory and annihilate under the emission of a photon pair [12]. The photons travel in opposite directions and are detected on the gamma-cameras surrounding the subject. Since the photons' trajectories were antiparallel, it is possible to trace a line between the two detection points and find the line along which the annihilation took place. When many photon-pairs are detected, this provides valuable information on the three-dimensional distribution of the PET-tracer. SPECT, as the name suggests, uses tracers that emit just a single photon upon decay. The photon is also detected on a gamma-camera, but since there is just one photon per decay, it is not possible to retrace its trajectory like in PET. To resolve this issue, pinhole collimators can be added between the subject and the detectors. A collimator is a thick slab of radiation-hard material with narrow channels penetrating from one side to the other [18]. Collimators 'filter' the incoming radiation, since rays that are not travelling straight through the channel are attenuated. By shaping the channel like a small pinhole, the images are magnified on the gamma-camera. When a photon is detected on the gamma-camera, it can be traced back, though the aperture of one of the pinholes in the collimator, to a single line in the scanner volume. This enables the tracer distribution to be recovered through a process called image reconstruction.

Medical image reconstruction is an intricate process. It belongs to the domain of inverse mathematical problems, where one computes from a set of measurements the unknown factors that produced these measurements[2]. In the case of nuclear imaging, the goal is to determine the volumetric radioactive activity in the subject from the collected photon-counts on the detector pixels. This proves to be challenging for all nuclear imaging techniques, but additional challenges are posed for dynamic SPECT especially. This is caused by the low-count and high noise projection data. The low counts are in part caused by attenuation of photons inside the collimator. Moreover, a higher temporal resolution in dynamic SPECT causes the number of counts per time-frame to decrease. The collected detector data is also contaminated with noise, due to the statistical nature of radioactive decay of the tracer. This noise causes the problem of image reconstruction for SPECT to be ill-conditioned, implying that a small perturbation in the observed counts may cause large deviations in the reconstructed image[2].

Various techniques can be employed to estimate the activity distribution. These can be partitioned into two categories; analytical and iterative methods. The choice for one or the other may be based on (a combination of) the available computational power, the amount of collected data and the desired image quality. Analytical methods are computationally less demanding, hence their popularity in the earlier days of nuclear imaging [3]. One of the major downsides of these techniques, however, is that the approximations and interpolations employed in analytical methods tend to give rise to artefacts in the reconstruction. An artefact is an object that is not present in the real image, but is caused by the reconstruction. In medical applications, such artefacts may have severe effects, for example when such a non-existent object is misdiagnosed as a tumour[24]. Furthermore, analytical reconstruction is usually not achievable for scans with complex geometries, such as those performed with multi-pinhole collimators.

Iterative algorithms make use of the physical and statistical models that underlie the SPECT scanner. This reduces the sensitivity to noise in the projection and decreases the chances of artefacts appearing in the reconstructed image. As the name suggests, iterative methods use a repeating algorithm to update the estimated activity distribution before it arrives at the final reconstructed image. This requires significant computational power, since each iteration involves a forward and backward projection of the current estimate[3]. Since computers have evolved remarkably since the invention of SPECT, iterative reconstruction methods have gained popularity over the last decades and are expected to spread even further in the upcoming years. Nonetheless, iterative methods may still require long computations, especially since the desired resolution is ever-increasing and convergence of the image tends to require many iterations.

Maximum Likelihood Expectation Maximisation (MLEM) is a fundamental method, that takes into account the Poisson statistics that underlie radioactive decay and models the scanner's geometry using a system matrix. This matrix encapsulates the physical laws that dictate how the tracer distribution is projected onto the detectors. Specifically, this matrix allows one to compute the expected number of detected counts for each detector pixel, given an activity distribution. MLEM employs expectation maximisation to find the tracer distribution that caused the recorded scanner data with the greatest likelihood, in an iterative manner. The MLEM algorithm is relatively easy to implement and intuitive to understand, but it is sensitive to noise and the images tend to converge slowly[31]. Moreover, it performs poorly at reconstructing singular low-count timeframes for dynamic SPECT.

The reconstruction of these singular time-frames can be aided by incorporating information from earlier and/or later time-frames in their reconstruction. Wang and Qi (2015) [35] provided such an algorithm for dynamic PET image reconstruction in their paper 'PET Image Reconstruction Using Kernel Method'. For their novel method, they were inspired by some well-established principles from the field of machine learning. Using these ideas, they were able to construct a second matrix, the 'kernel matrix', which contains information from the projections in all time-frames of the scan. Wang and Qi applied their proposed method to PET patient datasets and found promising results that outperformed a number of other methods, including MLEM. Therefore, KEM has the potential to enables a higher temporal resolution for dynamic PET or allow for more low-dose SPECT scanning, which decreases a patient's exposure to harmful radiation[23].

In this paper, KEM is implemented for dynamic multi-pinhole SPECT in order to discover whether the method yields better reconstructions than conventional MLEM. Before pitching into the image reconstruction methods, some background on dynamic SPECT scanning is provided in Chapter 2. After that, the two iterative algorithms will be deduced and introduced in Chapter 3. Moreover, this chapter includes the phantoms that are used in the simulations in this paper. The results of the simulations are presented in Chapter 5. Conclusions on the performance of KEM in comparison to MLEM for dynamic SPECT are drawn in the last chapter.

2
Background

In this chapter, some background of dynamic SPECT scanning and image reconstruction is provided.

2.1. SPECT

Single Photon Emission Computed Tomography (SPECT) is a non-invasive method used for diagnosis of diseases and medical research. Unlike many other nuclear imaging techniques, the radiation is not emitted by the scanning apparatus itself, but by radioisotopes that are inside the patient. Depending on the design of the research and the half-life of the radio-isotope, doctors inject the patient with radio-labelled drugs or ligands a fixed amount of time ahead of the scan. During the SPECT-scan, the emitted photons are collected on gamma-detectors that surround the patient. The collected data is commonly referred to as the projection. Ultimately, the goal of a SPECT scan is to obtain a three-dimensional image of the ligand-concentration in the patient. This enables scientists to analyse an abundance of biological processes inside the body as a whole or in specific organs.

The most popular isotope used for SPECT is metastable Technetium-99 (^{99m}Tc). It decays to Technetium-99 under the emission of a gamma-photon with an energy of $140keV$. This photon energy is just high enough to be detected by common gamma-cameras, thus limiting the negative side effects of the radiation on the subject. Thanks to its short half-life of 6.0058 hours, ^{99m}Tc allows fast data collection due to its high activity. The cumulative patient radiation exposure still remains low, since over 95% of the isotopes decay within a 24-hour time-frame [25].

$$^{99m}_{43}Tc \rightarrow {}^{99}_{43}Tc + \gamma_{\text{140keV}} \tag{2.1}$$

An example of a scan that is often ordered by medical professionals is a bone scan using ^{99m}Tc-labelled diphosphonates (^{99m}Tc-DPs). This specific radioactive tracer is absorbed by newly formed bone tissue and is therefore a useful marker for increased bone metabolism[22]. The presence of sites with high tracer concentration can be an indicator for cancer or other bone diseases. In many cases, the results of the SPECT-scan eliminate the further need to take a biopsy.

2.1.1. Dynamic SPECT

As mentioned, SPECT scans are useful for imaging a variety of biological processes inside the body. These processes can be imaged statically, where the images show only the spatial location and concentration of the tracer during the entirety of the scanning-time, or dynamically, which provides spatial as well as temporal information[21]. The choice for one of either techniques depends on the clinical applications, but it is important to bear in mind that there is a trade-off between temporal resolution and image quality. This is caused by the fact that reducing the scanning time per image frame also reduces the number of decays within the time-frame.

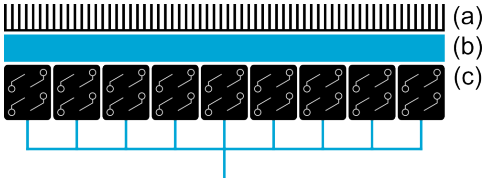


Figure 2.1: Schematic of a gamma-camera consisting of a parallel-hole collimator (a), a scintillating crystal (b) and an array of photo-multiplier tubes (c).

2.2. SPECT scanners

2.2.1. Gamma-camera

The SPECT scanner studied in this paper is the three-headed Ultra-high resolution Single Photon Emission Computed Tomography (U-SPECT/CT) system jointly developed by the commercial company MILabs B.V. and the research group at the TU Delft. It is used by institutions worldwide for pre-clinical studies on rodents[17]. The photons are detected on three large-area, stationary gamma-detectors by means of a process known as scintillation detection. In Figure 2.1, a schematic of a single gamma-camera is shown. The collimator (a) is a slab of radiation-hard material with parallel holes in it. This collimator ensures that only the gamma-rays hitting the camera at a 90° angle are projected onto the crystal underneath it. This, in turn, allows the rays that strike the crystal to be traced back to their originating position in the phantom during reconstruction.

Due to the presence of the collimator, only a minor selection of photons reaches the scintillating crystal (b). Upon impact with the crystal, the gamma-photon is absorbed by an electron inside the crystal, which causes it to move from the conduction band to the valence band. If this happens, the electron is free to move through the lattice. It leaves a net positive charge ('hole') behind, which is attracted to the moving electron through Coulomb forces. The electron-hole pair moves through the crystal until it is trapped by a defect in the lattice. In the case of NaI(Tl), the electron-hole pairs are captured by the Thallium atoms where the electron decays and subsequently thousands of lower-energy photon, in the visible range, are emitted[10].

After the high-energy gamma-photons are converted to light flashes by the scintillator, they must be recorded in a way that also keeps track of the position on the gamma-camera where the gamma-ray hit the camera. This is done by an array of photo-multiplier tubes (PMTs) (c). When a light flash strikes the top of a PMT (the photo-cathode), it causes the cathode to release an electron. The electron is then accelerated towards the first dynode due to an electric field. It then enters a cascade of emission, acceleration and absorption, where, at each dynode, the number of emitted electrons increases exponentially[16]. Finally, the amplified stream of electrons strikes the anode, which results in an easily detectable current pulse. The pulses on the detector plate are counted during the scanning-time for each detector pixel separately, and this data is used to reconstruct the tracer distribution inside the scanner.

The detector as displayed in figure 2.1 has a finite resolution, due to the limitations in the density of channels in the collimator and the photon-yield of the scintillator. A typical gamma-camera nowadays is able to achieve a spatial resolution of 3.5 mm for rays of 140 keV (corresponding to the rays emitted by ^{99m}Tc)[18].

2.2.2. Pinhole collimator

In order to retrace the location where the recorded photon was emitted, a pinhole collimator is placed between the object of interest and the gamma-camera. The channels in this collimator are shaped like pinholes and essentially behave as small apertures, which magnify and invert the image that is projected onto the detectors (see Figure 2.2). This results in a superior spatial resolution, without the need to further increase the resolution of the gamma-camera itself, which can be very costly[5]. The magnification factor is proportional to the distance between the distance between the object and the centre of the pinhole, thus making it especially applicable in imaging of small animals, where only a small field of view is required.

The full layout of the SPECT scanner is depicted in Figure 2.4. As shown, the stationary gamma-detectors are placed in an equilateral triangular set-up to collect data from multiple angles. The pinhole collimator is cylindrical, such that all pinholes focus on a central scan volume. In order to scan structures

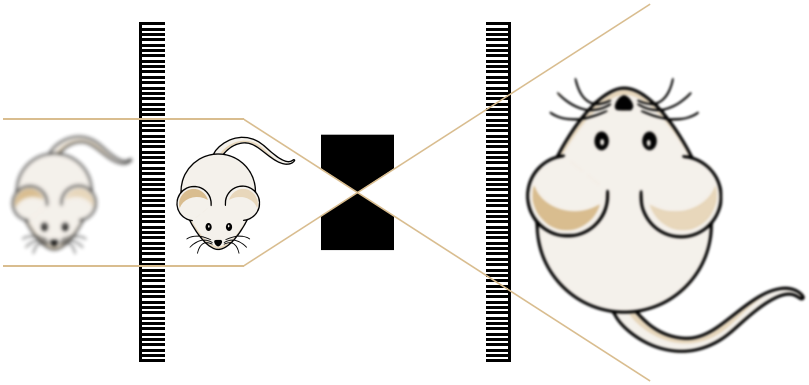


Figure 2.2: Schematic of the projection of a mouse by means of a parallel-hole collimator only (left-hand side) and by means of a combination of a parallel-hole collimator and a pinhole collimator (right-hand side).

Figure 2.3: A cylindrical multi-pinhole collimator with 75 pinholes. Image taken from Van der Have [27].

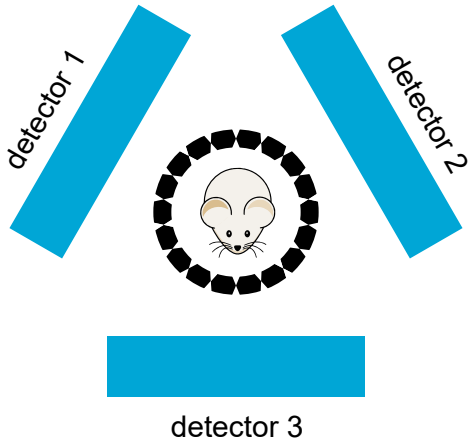


Figure 2.4: Schematic of a three-headed SPECT-scanner with a phantom placed inside of a collimator.

that do not fit within the central field of view of the converging pinholes, the object is sequentially stepped through the scanner to collect photons from all parts of the structure. Following this method, a sub-millimeter resolution can be achieved[5].

2.3. Mathematical basis

The naming conventions used in this paper are summarised in the Nomenclature on page iii for easy reference.

2.3.1. Variable definitions

Let $f(x, y, z)$ denote the unknown number of photons emitted at each point (x, y, z) inside the scanner. This number corresponds one-to-one with the number of radioactive decays at that position. Moreover, we assume that this quantity is directly proportional to the tracer activity in (x, y, z).

Next, g represents the expected projection of f on the detector plates. In the case of a three-headed scanner, g comprises three two-dimensional images, each corresponding to another detector-plate. The g value for each pixel on a gamma-camera is given by the expected number of detected gamma-photons in that pixel within a given time-frame.

To enable the computations that are involved with the reconstruction, the projection on the gamma-camera is discretised into I square, two-dimensional pixels, each labelled by an index. In the remainder of this paper, the pixel index will be labelled by the letter i. Similarly, the scanner volume is discretised into J three-dimensional, cubic voxels, each labelled by a different j-index. Thus, we can conveniently represent the tracer distribution as the column vector

$f = (f_1, f_2, ..., f_J) \in \mathbb{R}^J$

and, likewise, the counts can be packed into the vector

$g = (g_1, g_2, ..., g_N) \in \mathbb{N}^I$

2.3.2. Forward model

It goes without saying that the measured projection depends on the tracer distribution and on the geometry of the scanner system. More precisely, the number of counts in detector pixel i (g_i) is a weighted sum over the activities in all voxels inside the field of view of the SPECT scanner (see Eq. 2.2).

$$g_i = p_{i,1}f_1 + p_{i,2}f_2 + ... + p_{i,m}f_m = \sum_{j=1}^{J} p_{i,j}f_j \tag{2.2}$$

The weights $p_{i,j}$ represent the probability that a photon emitted in voxel j is detected in pixel i. The summation notation suggests we may summarise Eq. 2.2 for all pixels as a matrix product,

$$g = Pf, \tag{2.3}$$

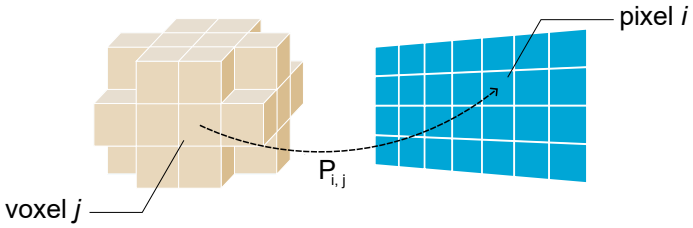


Figure 2.5: The matrix element $P_{i,j}$ denotes the probability that a gamma-ray emitted from voxel j in the activity volume (left-hand side) is detected in pixel i on the detector plate (right-hand side).

where P is an $I \times J$ matrix and the (i, j)th element of P is equal to $p_{i,j}$. This matrix is referred to as the system matrix. The elements and size of the system matrix depend on the characteristics of the scanner, such as its geometry and the type of collimator that is used. The construction of the system matrix can be performed in numerous ways, both experimentally and using simulations. In the former case, one would sequentially step a point source (radioactive bead) along the voxels in the scanner's field of view and record the counts on the detector for each position[6]. Not surprisingly, this process takes a lot of time, especially when the scanner volume is large or when the voxel size is reduced. Oftentimes, fewer measurements are taken and interpolation is used to compute the matrix elements for all J voxels. Alternatively, advanced software toolkits such as GATE are available to simulate the physics behind complex pinhole geometries and absorption in collimators numerically. It uses Monte Carlo simulations to compute the matrix elements[28].

2.3.3. Image reconstruction

The system matrix allows us to calculate the expected projection using Eq. 2.3, given the tracer distribution. During image reconstruction, we want to do the opposite, namely find the unknown tracer distribution f from the recorded projection g. This makes image reconstruction for SPECT an inverse mathematical problem[2]. Theoretically, this can be solved directly by multiplying both sides of Eq. 2.3 with the left-inverse of P^{-1}. In practice, however, this is not feasible, due to a number of reasons. Firstly, the computational complexity of calculating an inverse matrix scales rapidly with its dimensions. Moreover, it is important to note that real-life SPECT reconstruction is an ill-posed problem by the definition of Hadamard[15].

Definition 1 (Well-posedness (Hadamard))**.** The problem of determining a solution $g = Pf$ in a metric space $\mathcal{G}$ (with distance $\rho_{\mathcal{G}}(\cdot, \cdot)$) from initial data f in a metric space $\mathcal{F}$ (with distance $\rho_{\mathcal{F}}(\cdot, \cdot)$) is well-posed if all three criteria are satisfied:

- **Existence** For any $g \in \mathcal{G}$ there exists a solution $f \in \mathcal{F}$;
- **Uniqueness** For any $g \in \mathcal{G}$ there exists exactly one solution $f \in \mathcal{F}$;
- **Stability** The problem is stable with respect to the spaces $(\mathcal{G}, \mathcal{F})$: For any $\epsilon > 0$ there exists a $\delta(\epsilon) > 0$ such that, for any $f_1, f_2 \in \mathcal{F}$, the inequality $\rho_{\mathcal{F}}(f_1, f_2) < \delta(\epsilon)$ implies $\rho_{\mathcal{G}}(g_1, g_2) < \epsilon$, where $g_1 = R(f_1)$, $g_2 = R(f_2)$.

Problems that are not well-posed, are called ill-posed.

In real-world SPECT-scans, the measurement data that is collected is not a perfect forward projection as described in equation 2.3. The projections are corrupted by random scatters and missing counts due to attenuation photons within the object of interest. These effects undermine the existence, uniqueness and stability of a solution.

Apart from noise and attenuation, another complicating factor for SPECT reconstruction is the low counting statistic. Due to the pinhole collimator in the system, almost all radiation is absorbed before it could even reach the detector. This property is described by the sensitivity of the scanner, which refers to the ratio of emitted photons that reach the detector plates. This ratio may be expressed as the number of counts per second per Becquerel or as a percentage. In a previous paper on multi-pinhole SPECT, the photon-sensitivity has been shown to be in the order of 250 cps/MBq, or equivalently, 0.025%[7]. The number of recorded counts can be increased by simply increasing the acquisition time, but this is undesirable when using live animals and is inefficient when scanner-time is precious. Injecting more

tracer to increase the activity is also not preferable, as this increases the radiation dose absorbed by the patient.

3

Reconstruction Methods

A variety of reconstruction techniques for nuclear imaging exist. Throughout the years since the invention of SPECT scans, different techniques gained and lost popularity due to developments in computing power, size of the projection data and the desired image quality, amongst other factors. There are broadly two categories of techniques; analytical and iterative algorithms.

3.1. Analytical methods

For a long time, analytical methods, such as Filtered Backprojection, used to be the industry-standard for SPECT reconstruction. It uses mathematical techniques to calculate the tracer distribution from the acquired projection data directly and in a single step, which makes analytical reconstruction a computationally efficient method[29].

Unfortunately, analytical methods rely on a number of underlying assumptions. The data is assumed to be free of noise, attenuation and scatter. Needless to say, these conditions are violated in real-life SPECT scans. Another assumption, namely that projections are collected from an unlimited number of detection angles, is also not met. As a consequence, the uniqueness of the analytical solution is not guaranteed. Small deviations from the noise-free projection may also cause unphysical negative values for the activity in some voxels and may give rise to artefacts in the reconstruction. This can have severe effects when such artefacts are misinterpreted in clinical diagnoses[24]. These factors make analytic methods inadequate for pinhole SPECT[31].

3.2. Iterative methods

Iterative methods have been around since the very invention of SPECT, however, their application in clinical practise was delayed due to lacking computational capacity[29]. They are more demanding because, as the name suggests, multiple repeating steps are needed to arrive to an image of sufficient quality. The advantage of iterative methods is their reduced sensitivity to noise compared to analytic methods. Moreover, iterative methods include the underlying laws of physics and the collimator geometries in the reconstruction algorithm. This makes iterative reconstruction preferable for multi-pinhole SPECT. This paper will explore a number of these methods, which all branch from Maximum Likelihood Expectation Maximisation.

3.2.1. Maximum Likelihood Expectation Maximisation

The most elemental iterative method considered in this paper is Maximum Likelihood Expectation Maximisation (MLEM). It was first proposed by Shepp and Vardi in 1982, and it incorporates the statistical nature of radioactive decays. This makes MLEM more robust than analytical methods when it comes to noise[5]. Instead of trying to find a solution in one step, MLEM iteratively updates the reconstruction to find the tracer distribution that produced the projection data with the highest likelihood.

Similarly to many other counting experiments, the observed number of counts in projection pixel i (g_i) is modelled as a draw from a Poisson distribution. The discrete random variable g_i has probability mass function Eq. 3.1, where g_i is the observed number of counts in the projection pixel and $\langle g_i \rangle$ is the

expectation of g_i.

$$\mathbb{P}(g_i) = \frac{\langle g_i \rangle^{g_i} e^{-\langle g_i \rangle}}{g_i!} \tag{3.1}$$

The expected number of counts in the projection depends on the unknown tracer distribution f via the system matrix that was introduced in section 2.3.2.

$$\langle g_i \rangle = \mathbb{E}\left[g_i | f\right] = \sum_{j=1}^{J} p_{ij} f_j \tag{3.2}$$

To find the probability that some trace distribution f produced the observed projection in i, we define the partial likelihood function $L(g_i)$.

$$L(g_i|f) = \mathbb{P}(g_i|f) \tag{3.3}$$

To compute the likelihood of the whole image, we use that the g_i are independent random variables. This results in the ensemble likelihood function $L(f)$, which is simply the product of the partial likelihood functions.

$$L(g|f) = \prod_{i=1}^{I} L(g_i|f) = \prod_{i=1}^{I} \frac{\langle g_i \rangle^{g_i} e^{-\langle g_i \rangle}}{g_i!} \tag{3.4}$$

Maximum Likelihood Expectation-Maximisation, fittingly, is aimed at finding a tracer distribution f^{MLEM} that maximises the expectation of the likelihood above. In order to obtain physically valid solutions only, the solution space is limited to $F = \{f \in \mathbb{R}^J | f_j \geq 0, \forall j = 1, ..., J\}$, i.e. the space where the activity in all voxels is non-negative.

$$\hat{f}^{\mathsf{MLEM}} = \arg\max_{f \in F} L(g|f) \tag{3.5}$$

As per usual, this can be done by differentiating the likelihood with respect to f and equating to zero. A common trick in statistical analysis is to minimise the negative natural logarithm of the likelihood function, rather than the likelihood function directly. This facilitates easier differentiation, while resulting in the same optimiser[11].

$$\begin{aligned} l(g|f) &= -\ln(L(g|f)) \\ &= \sum_{i=1}^{I} \left(-g_i \ln(\langle g_i \rangle) + \langle g_i \rangle + \ln(g_i!)\right) \\ &= \sum_{i=1}^{I} \left(-g_i \ln\left(\sum_{j=1}^{J} p_{ij} f_j\right) + \left(\sum_{j=1}^{J} p_{ij} f_j\right) + \ln(g_i!)\right) \end{aligned} \tag{3.6}$$

The last term in Eq. 3.6 is independent of f and therefore does not affect the values of vector f at which $l(g|f)$ attains its extreme values. Thus, this term is omitted from the equation in the remainder of this paper. Differentiating Eq. 3.6 pixel-wise yields Eq. 3.7.

$$\frac{\partial l(g|f)}{\partial f_j} = \sum_{i=1}^{I} p_{ij} - \sum_{i=1}^{I} \frac{p_{ij} g_i}{\sum_{j=1}^{J} f_j p_{ij}} = 0 \tag{3.7}$$

This lead Shepp and Vardi to the update equation that makes up every iteration of MLEM.

$$\hat{f}_j^{\,new} = \frac{\hat{f}_j^{\,old}}{\sum_{i=1}^{I} p_{ij}} \left(\sum_{j=1}^{J} p_{ij} \frac{g_i}{\sum_{i=1}^{I} p_{ij} \hat{f}_j^{\,old}} \right) \tag{3.8}$$

Eq. 3.8 can be vectorised for the entire tracer distribution. Here, $\mathbb{1}_N$ is a vector of length N with all entries equal to one, n resembles the iteration number and the superscript T denotes the matrix transpose. The division $\frac{g}{P\hat{f}^n}$ is to be performed component-wise.

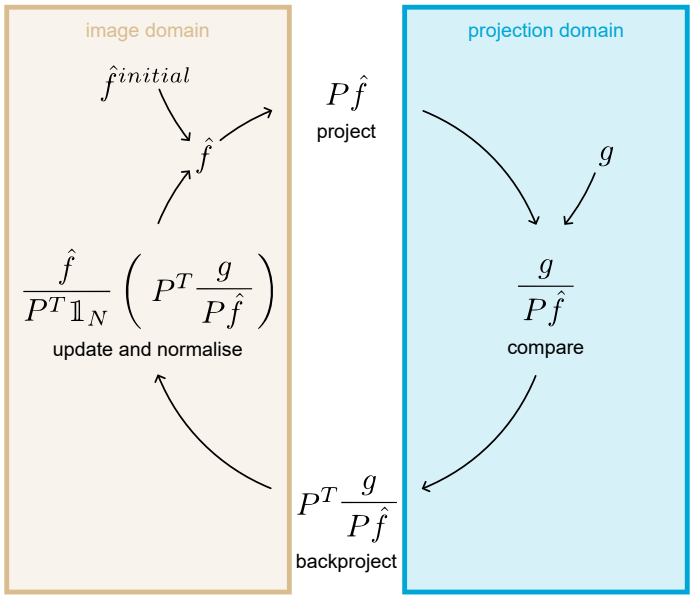


Figure 3.1: The MLEM iterations displayed in separate steps. The inputs are $\hat{f}^0$ and g, the initial guess for the activity and the recorded projection, respectively.

$$\hat{f}^{n+1} = \frac{\hat{f}^n}{P^T\mathbb{1}_I}\left(P^T\frac{g}{P\hat{f}^n}\right) \tag{3.9}$$

Intuitively, the update equation can be dissected into four successive steps. These are displayed graphically in Fig. 3.1. Firstly, the current estimate $\hat{f}$ is projected using the system matrix. Next, the ratio between this projection and the measured data g from the scan is computed. This ratio is a measure of the error in the current estimate. The error ratio is projected back to the image domain, after which the estimate is normalised and updated before entering a new iteration. The normalisation ensures that the number of counts in the forward projection of the estimates are the same as the counts in the measured projection[36].

It is important to note that the updates in the MLEM algorithm are multiplicative. This means that if a certain voxel has zero activity in estimate n, i.e. $\hat{f}_j^n$ = 0, it will remain zero in all following estimates. Therefore, it is of great importance to choose an appropriate $\hat{f}^0$, where only voxels which are known a priori to have no tracer are set to zero. Oftentimes, $\hat{f}^0$ is set to be uniform inside the field of view of the pinhole collimator and zero outside.

Furthermore, the images produced by MLEM are guaranteed to be non-negative, unlike with analytical reconstruction. To see this, observe that p_{ij} and gg_i are greater than or equal to zero for all i and j. If $\hat{f}^0$ has only non-negative values, then all subsequent estimates will be non-negative.

Convergence of the MLEM algorithm

The MLEM algorithm is guaranteed to converge to a maximum likelihood solution. A short proof of this statement was written by Iusem (1992) [19]. This section follows this proof and elaborates some steps that were skipped over in the proof by Iusem. In order to prove the converge to a maximum likelihood solution, a few definitions and assumptions are used.

The following assumptions are made regarding the observed data and the system matrix.

Assumption 1. Without loss of generality, assume $\sum_{i=1}^{I} g_i = 1$, that is, assume that the sum of counts is normalised.

Assumption 2. Assume $\sum_{i=1}^{I} p_{ij} = 1$, that is, assume the probability of detection of an event is equal to one.

Assumption 3. Assume $\sum_{j=1}^{J} p_{ij} \neq 0$, that is, assume that there are no all-zero rows in P i.e. there are no 'blind' pixels. This assumption can be justified by observing that 'blind' pixels can safely be omitted from the model.

Moreover, some sets are defined as follows.

Definition 2 ($\mathbb{R}_+$)**.** Let $\mathbb{R}_+$ denote the set of all non-negative real numbers.

Definition 3 ($\mathbb{R}_{++}$)**.** Let $\mathbb{R}_{++}$ denote the set of all positive real numbers (this does not include 0).

Definition 4 ($\bar{\Delta}$)**.** Let $\bar{\Delta}$ denote the set $\{f \in \mathbb{R}_+^J : \sum_{j=1}^{J} f_j = 1\}$.

Definition 5 (Δ)**.** Let Δ denote the set $\{f \in \mathbb{R}_{++}^J : \sum_{j=1}^{J} f_j = 1\}$.

Definition 6 (Γ)**.** Let Γ denote the set $\{f \in \mathbb{R}^J : Af > 0\}$. Here, the comparison '$>$' is understood to be element-wise, i.e. $x > y$ if $x_j > y_j$ for all j.

Recall that the MLEM update equation is given by $f_j^{k+1} = \frac{f_j^k}{\sum_{i=1}^{I} p_{ij}} \sum_{i=1}^{I} \frac{g_i p_{ij}}{\sum_{j=1}^{J} p_{ij} f_j^k}$. Using assumption 2, this simplifies to $f_j^{k+1} = f_j^k \sum_{i=1}^{I} \frac{g_i p_{ij}}{\sum_{j=1}^{J} p_{ij} f_j^k}$. To further shorten notation, we define the function U.

Definition 7 (U)**.** Let $U : \bar{\Delta} \cap \Gamma \to \bar{\Delta} \cap \Gamma$ be given by $U(f)_j = f_j \sum_{i=1}^{I} \frac{g_i p_{ij}}{\sum_{j=1}^{J} p_{ij} f_j}$.

Using this notation, the update equation in can be conveniently written as $f_j^{k+1} = U(f^k)_j$. Note that the update equation is multiplicative. If the activity in some voxel is set to zero in f^0, it will remain zero in all subsequent iterations. Hence, the following assumption on the domain of f^0 is necessary to guarantee convergence.

Assumption 4. The initial activity f^0 is in Δ.

This assumption leads to two useful properties of the sequence of estimates $\{f_k\}$, which are written up in the next two lemmas.

Lemma 1. f^k is in Δ for all k.

Proof. This can be shown through mathematical induction. By assumption 4, f^0 is in Δ. Now, suppose f^k is in Δ. Then $\sum_{j=1}^{J} f_j^k = 1$. We need to show that f^{k+1} is in Δ, i.e. that $\sum_{j=1}^{J} f_j^{k+1} = 1$. To see this, use the update equation:

$$\sum_{j=1}^{J} f_j^{k+1} = \sum_{j=1}^{J} \left[f_j^k \sum_{i=1}^{I} \frac{g_i p_{ij}}{\sum_{j=1}^{J} p_{ij} f_j^k} \right] = \sum_{i=1}^{I} \sum_{j=1}^{J} \left[f_j^k \frac{g_i p_{ij}}{\sum_{j=1}^{J} p_{ij} f_j^k} \right] = \sum_{i=1}^{I} g_i \frac{1}{\sum_{j=1}^{J} p_{ij} f_j^k} \sum_{j=1}^{J} \left[f_j^k p_{ij} \right] = \sum_{i=1}^{I} g_i = 1. \tag{3.10}$$

The last equality is a consequence of assumption 1. This proves the induction step. □

Lemma 2. The sequence $\{f^k\}$ is bounded.

Proof. From Remark 1, all f^k are in Δ. Then $\sum_{j=1}^{J} f_j^k = 1$ and the f_j are in $\mathbb{R}_{++}$. This implies that the f_j are bounded below by zero and above by 1. Therefore, $\|f^k\|_2 \leq 1 \cdot J$ for all k, so $\{f^k\}$ is bounded. □

Recall that the negative log-likelihood function $l : \bar{\Delta} \cap \Gamma \to \mathbb{R}$ was derived in Eq. 3.6. For this proof, the domain of l is extended to $\bar{\Delta}$ as follows:

$$l(f) = \begin{cases} \sum_{i=1}^{I} \left(-g_i \ln \left(\sum_{j=1}^{J} p_{ij} f_j \right) + \sum_{j=1}^{J} p_{ij} f_j \right), & \text{if} f \in \bar{\Delta} \cap \Gamma \\ +\infty, & \text{if} f \in \bar{\Delta} \setminus \Gamma \end{cases} \tag{3.11}$$

Thus, the negative log-likelihood of activity data that results in zero or negative observed counts in a projection pixel, is set to approach infinity.

Remark 1. The set Δ is a convex set.

Theorem 1. The negative likelihood function l is convex, and it attains its minimum at points in Δ.

Proof. The twice differentiable negative log-likelihood function of several variables is convex on a convex set if and only if its Hessian matrix of second order partial derivatives is negative semi-definite on the interior of the convex set. Recall that Δ is a convex set (Remark 1).

The Hessian matrix of l is given by

$$H_l = \begin{bmatrix} \frac{\partial^2 l}{\partial f_1^2} & \cdots & \frac{\partial^2 l}{\partial f_1 \partial f_J} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 l}{\partial f_J \partial f_1} & \cdots & \frac{\partial^2 l}{\partial f_J^2} \end{bmatrix} \tag{3.12}$$

$$P = \begin{bmatrix} p_{11} & \cdots & p_{1J} \\ \vdots & \ddots & \vdots \\ p_{N1} & \cdots & p_{NJ} \end{bmatrix} \tag{3.13}$$

$$K = \begin{bmatrix} \kappa(\phi_1, \phi_1) & \cdots & \kappa(\phi_1, \phi_J) \\ \vdots & \ddots & \vdots \\ \kappa(\phi_J, \phi_1) & \cdots & \kappa(\phi_J, \phi_J) \end{bmatrix} \tag{3.14}$$

The second order partial derivatives are computed as follows:

$$\frac{\partial^2 l}{\partial f_a \partial f_b} = \frac{\partial}{\partial f_a} \left(\sum_{i=1}^{I} p_{ib} - \frac{g_i p_{im}}{\sum_{j=1}^{J} p_{ij} f_j} \right) = -\sum_{i=1}^{I} \frac{g_i p_{ia} p_{ib}}{\left(\sum_{j=1}^{J} p_{ij} f_j \right)^2} = \frac{\partial^2 l}{\partial f_b \partial f_a} \tag{3.15}$$

Clearly, H_l is a symmetric matrix. To check that H_l is negative semi-definite, we need to show that $u^T H_l u \leq 0$ for all $u \in \mathbb{R}^J$. To this end, let $u \in \mathbb{R}^J$ be arbitrary. The product $u^T H_l$ results in a J dimensional row-vector:

$$\left[\sum_{j=1}^{J} u_j \frac{\partial^2 l}{\partial f_1 \partial f_j}, \quad \sum_{j=1}^{J} u_j \frac{\partial^2 l}{\partial f_2 \partial f_j}, \quad \ldots, \quad \sum_{j=1}^{J} u_j \frac{\partial^2 l}{\partial f_J \partial f_j} \right] \tag{3.16}$$

Multiplying by u gives a scalar

$$u^T H_l u = \sum_{j'} u_{j'} \left[\sum_{j=1}^{J} u_j \frac{\partial^2 l}{\partial f_{j'} \partial f_j} \right] \tag{3.17}$$

Substituting the partial derivatives from Eq. 3.15 and rearranging the (now triple) summation yields

$$u^T H_l u = -\sum_{i=1}^{I} g_i \left[\sum_{j'=1}^{J} \frac{u_{j'} p_{ij'}}{\sum_{j"=1}^{J} p_{ij"} f_{j"}} \sum_{j=1}^{J} \frac{u_j p_{ij}}{\sum_{j"=1}^{J} p_{ij"} f_{j"}} \right] \tag{3.18}$$

The two summations between square brackets are mutually independent, and identical up to the naming of the index. Thus, we can combine the factors into a square term, which gives the desired result; $u^T H_l u \leq 0$ for all $u \in \mathbb{R}^J$.

$$u^T H_l u = -\sum_{i=1}^{I} \left(\frac{\sqrt{g_i}}{\sum_{j=1}^{J} p_{ij} f_j} \sum_{j=1}^{J} p_{ij} u_j \right)^2 \leq 0 \tag{3.19}$$

□

We can close the gap between the likelihood equation and the update equation by examining the minimisers of the likelihood function and the fixed points of the update equations.

Definition 8 (G)**.** Let G denote the set of minimisers of l.

Definition 9 (F)**.** let F denote the set of fixed points of U. Fixed points of U are the points $f_n \in \bar{\Delta} \cap \Gamma$ where $f = U(f)$.

Theorem 2 ($G \subseteq F$)**.** The set G is contained in the set F.

Proof. For differentiable convex functions $l : \mathbb{R}^J \to \mathbb{R}$, a necessary and sufficient condition for a vector f in its domain to be a minimiser is that $\nabla l(f) = 0$.

$$\frac{\partial l}{\partial f_j} = \sum_{i=1}^{I} p_{ib} - \frac{g_i p_{im}}{\sum_{j=1}^{J} p_{ij} f_j} = 0 \tag{3.20}$$

Rearranging this expression and using assumption 2 gives:

$$\sum_{i=1}^{I} \frac{p_{ij} g_i}{\sum_{j=1}^{J} p_{ij} f_j} - 1 = 0 \tag{3.21}$$

Multiplying both sides by f_j,

$$f_j \sum_{i=1}^{I} \frac{p_{ij} g_i}{\sum_{j=1}^{J} p_{ij} f_j} - f_j = 0 \tag{3.22}$$

$$U(f)_j - f_j = 0 \tag{3.23}$$

Thus, f is a fixed point of U. □

The bivariate function d denotes the Generalised Kullback-Leibner divergence (GKLD) in Δ. The GKLD computes the expectation of the logarithmic difference between the probability of the observed data in the first distribution (x) with the second distribution (y)[20].

Definition 10 (Generalized Kullback-Leibler divergence)**.** Define $\delta : \mathbb{R}_+ \times \mathbb{R}_{++} \to \mathbb{R}$ as

$$\delta(a, b) = \begin{cases} a \ln \frac{a}{b}, & \text{if } a > 0 \\ 0, & \text{if } a = 0. \end{cases} \tag{3.24}$$

Define $d : \bar{\Delta} \times \Delta \to \mathbb{R}$ as

$$d(x, y) = \sum_{j=1}^{J} \delta(x_j, y_j). \tag{3.25}$$

The functions δ and d have some well-known properties, four of which are of importance in this proof. Proofs of these, and more properties of the GKLD, can be found in [1].

1. $\delta(a, b)$ is continuous and jointly convex in a and b
2. $d(x, y) \geq 0$ for all x in $\bar{\Delta}$ and y in Δ
3. $d(x, y) = 0$ if and only if $x = y$
4. For x in $\bar{\Delta}$, $\{y^k\} \subset \Delta$, $\lim_{k\to\infty} d(x, y^k) = 0$ if and only if $\lim_{k\to\infty} y^k = x$

Using property 1 in definition 10, the joint convexity of δ, the inequality in theorem 3 can be derived.

Theorem 3. For all $z \in \bar{\Delta} \cap \Gamma$, $x \in \Delta$ we have $d(U(z), U(x)) \leq d(U(z), x) - d(U(z), z) + l(z) - l(x)$.

Proof. See [19]. □

The last function that is used to prove our final theorem, that the MLEM estimates converge to a maximum likelihood solution, is ϕ_z.

Definition 11. For some fixed z in $\bar{\Delta}$ define $\phi_z : \bar{\Delta} \cap \Gamma \to \mathbb{R}$ as

$$\phi_z(x) = \sum_{j=1}^{J} z_j \ln \left(\sum_{i=1}^{I} \frac{g_i p_{ij}}{\sum_{j=1}^{J} p_{ij} x_j} \right) \tag{3.26}$$

Theorem 4. $\phi_z(x)$ has the following properties:

1. $\phi_z(x)$ is continuous
2. $\phi_z(x) = d(z,x) - d(z,U(x))$ for x in Δ
3. $\phi_z(x) \geq 0$ for all x in $\bar{\Delta} \cap \Gamma$
4. if $\phi_z(x) = 0$ then x is in G

Proof. 1. Trivial

2. To show this, start from the right-hand side:

$$d(z,x) - d(z,U(x)) = \sum_{j=1}^{J} \delta(z,x) - \sum_{j=1}^{J} \delta(z,U(x)) = \sum_{j=1}^{J} z_j \ln \frac{z_j}{x_j} - \sum_{j=1}^{J} z_j \ln \frac{z_j}{U(x)_j} \tag{3.27}$$

Combining the two summations and splitting the logarithms yields

$$d(z,x) - d(z,U(x)) = \sum_{j=1}^{J} [z_j \ln(z_j) - z_j \ln(x_j) - z_j \ln(z_j) - z_j \ln(U(x)_j)] \tag{3.28}$$

Filling in the definition of $U(x)_j$, cancelling like-terms and recombining the logarithms gives

$$d(z,x) - d(z,U(x)) = \sum_{j=1}^{J} z_j \ln \left(\frac{x_j \frac{\sum_{i=1}^{I} g_i p_{ij}}{\sum_{j=1}^{J} p_{ij} x_j}}{x_j} \right) = \phi_z(x) \tag{3.29}$$

3. This can be derived from Theorem 3 and taking z in G and x in Δ. From Theorem 2, we know that z is a fixed point. Thus $U(z) = z$. Plugging this into 3 gives:

$$d(z,U(x)) \leq d(z,x) - d(z,z) + l(z) - l(x) \tag{3.30}$$

Rearranging and applying the third GKLD property yields

$$d(z,x) - d(z,U(x)) = \phi_z(x) \geq l(x) - l(z) \geq 0 \tag{3.31}$$

Where the last inequality follows from the fact that z is a minimiser of l, implying that $l(z) \leq l(x)$ for x in Δ.

4. This also follows from Eq. 3.31,

$$\phi_z(x) = 0 \geq l(x) - l(z) \geq 0 \tag{3.32}$$

Therefore, we conclude that $l(x) = l(z)$. Since z minimises l, this implies that x also minimises l, hence x in also in G

□

Finally, we are able to prove the convergence of the MLEM algorithm to a maximum likelihood solution.

Theorem 5. The sequence $\{f^k\}$, defined by

- $f_j^{k+1} = f_j^k \sum_{i=1}^{I} \frac{g_i p_{ij}}{\sum_{j=1}^{J} p_{ij}} f_j = U(f^k)_j$
- $f^0 \in \Delta$

converges to a point f^* in G.

Proof. To prove the convergence, fix z in G. Combining properties 2 and 3 from Theorem 4 and plugging in f^k for x, we find that the sequence $\{d(z, f^k)\}$ is decreasing and bounded from below by zero.

$$0 \leq \phi_z(f^k) = d(z, f^k) - d(z, U(f^k)) = d(z, f^k) - d(z, f^{k+1}) \tag{3.33}$$

Since $\{d(z, f^k)\}$ is decreasing and bounded below by zero, this sequence converges. Consequently, the sequence $\{\phi_z(f^k)\}$ goes to zero. To see this, use part 2. of Proposition 2,

$$\lim_{k\to\infty} \phi_z(f^k) = \lim_{k\to\infty} \big(d(z, f^k) - d(z, U(f^k))\big) = \lim_{k\to\infty} \big(d(z, f^k) - d(z, f^{k+1})\big) = 0 \tag{3.34}$$

Recall from Remark 2 that the infinite sequence $\{f^k\}$ is bounded. By the Bolzano–Weierstrass theorem, $\{f^k\}$ has a convergent subsequence. Let $\{f^{k_m}\}$ denote a subsequence with limit point f^*, such that $\lim_{k\to\infty} f^{k_m} = f^*$.

From Eq. 3.33, we find that

$$d(z, f^*) \leq d(z, f^0) \tag{3.35}$$

Since z is a minimiser of l, f^0 is in $\bar{\Delta}$ and by property 2 of the function d, we know that the right-hand side of Eq. 3.35 is finite. Consequently, the left-hand side must also be finite. Since $d(z, f)$ approaches infinity for f in $\bar{\Delta} - \Gamma$, the limit point f^* must be in $\bar{\Delta} \cap \Gamma$

Combining the converging subsequence $\{f^{k_m}\}$ with the result in Eq. 3.34 gives

$$0 = \lim_{k\to\infty} \phi_z(f^{k_m}) = \phi_z(f^*) \tag{3.36}$$

By part 4. of Proposition 4, this implies x^* is in G. Thus, we may apply part 3. of Proposition 4,

$$0 \leq \phi_{f^*}(f^k) = d(f^*, f^k) - d(f^*, U(f^k)) = d(f^*, f^k) - d(f^*, f^{k+1}) \tag{3.37}$$

The above implies that $\{d(f^*, f^k)\}$ is decreasing and bounded below by zero. Therefore, $\{d(f^*, f^k)\}$ converges. Earlier, we noted that $\{f^k\}$ has subsequence $\{f^{k_m}\}$ which converges to f^*. Thus $\{d(f^*, f^{k_m})\}$ goes to zero by property 4 of GKLD. Since the limit of a convergent sequence equals the limit of any of its subsequences, we must have that the sequence $\{d(f^*, f^k)\}$ as a whole goes to zero. Again, using property 4 of GKLD, but now the other way, proves that $\lim_{k\to\infty} f^k = f^*$, where f^* was shown to be in G. □

Bare MLEM is notorious for its slow convergence, especially in low activity ('cold') regions. Many accelerating modifications to MLEM have been proposed during the last decades, but convergence of the image usually is often not preserved for these variations.

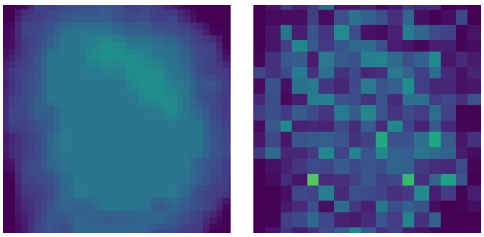


Figure 3.2: A smooth ground truth activity (left) is reconstructed using MLEM (right) and shows the checkerboard effect.

Interestingly, though, convergence to a maximum likelihood solution does not actually imply that the estimates are closer to the ground truth after every iteration. Initially, the images converge to the ground truth, but at later iterations, the image start to degrade due to the non-idealities, such as noise, in the projections. The images at high iteration numbers are dominated by high spatial frequencies and are thereby too noisy to be used for diagnosis. This is referred to as the chessboard or checkerboard effect, because the images show sharp edges with higher pixel values alternating with lower pixel values[9]. Early-stopping rules and low-pass post-filtering are used in order to obtain a smooth and accurate reconstruction[36]. The most commonly used filter is the Gaussian filter. The amount of blurring caused by the filter can be adjusted by tweaking the standard deviation σ of the applied filter. The best value of σ, one that reduces noise yet does not remove too much detail, needs to be determined ad hoc[4].

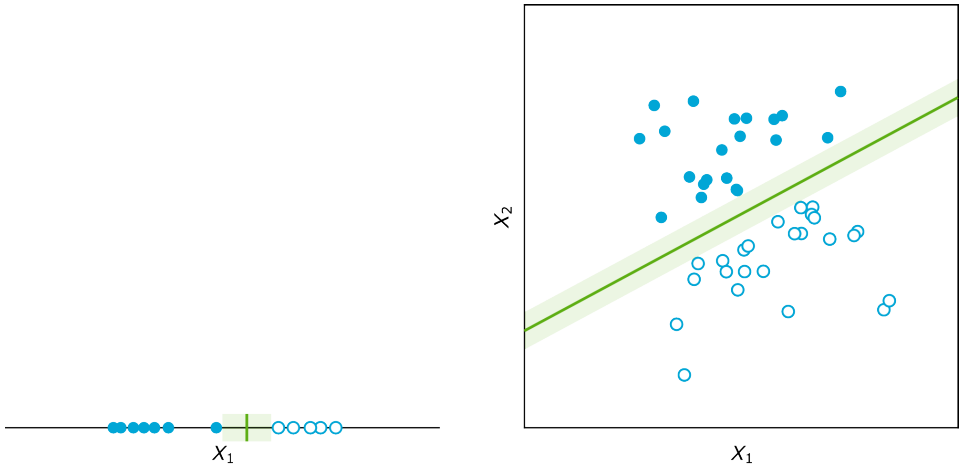


Figure 3.3: Examples of linearly separable datasets. The hollow and solid dots represent the two classes of data to be separated, the green line is the decision hyperplane and the green region is the margin. (I): dataset in $\mathbb{R}^1$. (II): dataset in $\mathbb{R}^2$.

3.2.2. Kernelised Expectation-Maximisation

MLEM image reconstruction for short time-frames in dynamic SPECT scanning is troublesome, due to low-counts and relatively high noise. Wang and Qi addressed this challenge for PET reconstruction in their paper "PET Image Reconstruction Using Kernel Method" (2015). Their proposed Kernelised Expectation-Maximisation (KEM) incorporates prior information about the activity in each voxel in the projection model to aid the reconstruction of a low-count time-frames. For the implementation of KEM, Wang and Qi took inspiration from several machine learning techniques, which are explained in the next sections.

Support Vector Machines

The Support Vector Machine (SVM) is a popular classification algorithm. It classifies new data points based on the position of the data point with respect to a so-called decision hyperplane. A hyperplane is an $(n-1)$-dimensional subspace that divides an n-dimensional space into two half-spaces. Thus, a hyperplane in one-dimension is a single point, in two-dimensions it is a line, in three dimensions it is a plane, etc. Defining the decision hyperplane is essentially an optimisation problem. The SVM will try to find a hyperplane which separates the data points of opposite labels and maximises the margin between the two subsets[8]. The margin is defined as the distance between the decision hyperplane and the data point closest to it. This data point is called the support vector, hence the name SVM.

In mathematical terms, let $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$ denote a dataset. $x_i \in \mathbb{R}^n$ is the feature vector of a data point and $y_i \in \{+1, -1\}$ is the target, i.e. the label that the SVM tries to fit. The optimal decision hyperplane can be written as in Eq. 3.38, where $w \in \mathbb{R}^n$ is the normal vector to the hyperplane.

$$w^T \cdot x = 0 \tag{3.38}$$

To decide on the label y^* of a feature x^*, the decision rule in Eq. 3.39 is in place.

$$y^* = \begin{cases} +1, & \textit{if } w^T \cdot x^* + b > 0 \\ -1, & \textit{if } w^T \cdot x^* + b < 0 \end{cases} \tag{3.39}$$

An example of a one dimensional dataset is depicted in Fig. 3.3.I. This dataset contains points in $\mathbb{R}^1$ which are either labelled by a hollow $(+1)$ or by a solid marker (-1). The green line marks the optimal decision hyperplane. Points on the left side of the decision-point are solid, the other points are hollow. The same principle extends to $\mathbb{R}^2$ (Fig. 3.3.II.) and higher dimensions alike.

'Lifting trick'[1]

The datasets shown in 3.3 are not very representative of real-life datasets, since they are linearly separable, whereas most datasets are not. A rudimentary example of a linearly inseparable dataset in plotted in Fig. 3.4.I. Here, the points near the centre of the data cluster are of the hollow class, while the points further away are solid. Thus, we cannot define a point that separates the data points into the two distinct classes, let alone optimise its margin.

To solve the problem of linear inseparability, the original features in $\mathbb{R}^n$ are 'lifted' to a higher-dimensional space $\mathbb{R}^N$ by a feature map Φ, such that the data points from the two different classes become linearly separable in the $\mathbb{R}^N$. The SVM will now be able to find the decision-hyperplane in this higher-dimensional space. The expression for this hyperplane is given in Eq. 3.40, where $v \in \mathbb{R}^N$ is a normal vector to the decision hyperplane in the higher dimensional space[14].

$$v^T \cdot \phi(x) = 0 \tag{3.40}$$

To decide on the label y^* of a feature $\phi(x^*)$, the modified decision rule in Eq. 3.41 is in place.

$$y^* = \begin{cases} +1, & \textit{if } v^T \cdot \phi(x^*) > 0 \\ -1, & \textit{if } v^T \cdot \phi(x^*) < 0 \end{cases} \tag{3.41}$$

The linear decision hyperplane can be mapped back to the original data space, resulting in a non-linear decision boundary.

The feature map used in Fig. 3.4 is Eq. 3.42. Unfortunately, however, in general it is not obvious what feature maps will enable the SVM to separate the data. Moreover, the data may require an extremely complex and high (or even infinite) dimensional mapping to facilitate support vector classification. Consequently, optimising the hyperplane will require heavy computations on polynomial combinations of feature vectors in $\mathbb{R}^N$. The computational costs of this may become rampant.

$$\phi\left(\begin{bmatrix} X_1 \\ X_2 \end{bmatrix}\right) = \begin{bmatrix} X_1 \\ X_2 \\ X_1^2 + X_2^2 \end{bmatrix} = \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} \tag{3.42}$$

'Kernel trick'

The kernel trick avoids the need to perform computations in the high-dimensional feature space explicitly. Instead, only a simple kernel function $\kappa : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is used. An example of a kernel function is the radial basis function (RBF) in Eq. 3.43.

$$\kappa(\phi_1, \phi_2) = \exp\left(-\gamma \|\phi_1 - \phi_2\|^2\right) = \exp\left(-\gamma\left(\phi_1^2 + \phi_2^2 - 2\phi_1\phi_2\right)\right) \tag{3.43}$$

Due to the norm that appears in the exponent, we can quickly deduce that feature vectors with a small distance between them result in a larger value of κ. Thus, in a sense, the kernel function is a measure of similarly between features. The kernel parameter γ scales the values of κ; decreasing γ means that the kernel function of feature-pairs returns larger values. For mathematical simplicity, assume $\gamma = \frac{1}{2}$.

To see how this simple formula implicitly performs computations in high-dimensional feature space, we use a Taylor expansion of the factor $\exp(\phi_1\phi_2)$. Recall that the general Taylor series expansion of $\exp(x)$ at $x = 0$ is given by Eq. 3.44.

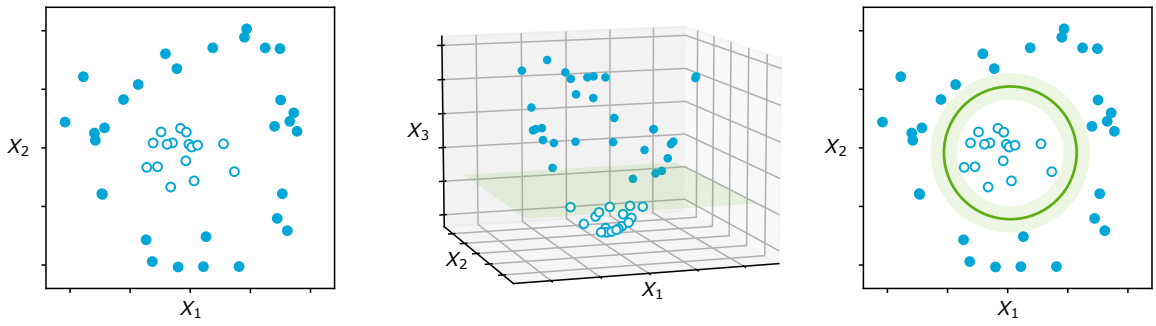
$$\exp(x) = 1 + \frac{1}{1!}x^1 + \frac{2}{2!}x^2 + \frac{3}{3!}x^3 + ... \tag{3.44}$$

Plugging in $\phi_1\phi_2$ in the place of x yields

$$\exp(\phi_1\phi_2) = 1 + \frac{1}{1!}(\phi_1\phi_2)^1 + \frac{2}{2!}(\phi_1\phi_2)^2 + \frac{3}{3!}(\phi_1\phi_2)^3 + ... \tag{3.45}$$

This expression can be written as the inner product between two rather fiddly vectors:

[1] The term 'lifting trick' was taken from the blogposts and dissertation by Dr. Gregory Gundersen



(I) (II) (III)

Figure 3.4: Demonstration of the 'lifting trick'. The hollow and solid dots represent the two classes of data to be separated, the green line is the decision hyperplane and the green region is the margin. (I): A linearly inseparable dataset in the input space. (II): The same dataset as in (I), mapped to $\mathbb{R}^2$ (the feature space), where the dataset is linearly separable by a hyperplane. The feature map is described by $X_3 = X_1^2 + X_2^2$. (III): The same dataset and hyperplane mapped back to the input space, where the decision hyperplane is non-linear.

$$\exp(\phi_1\phi_2) = \begin{bmatrix} 1 \\ \sqrt{\frac{1}{1!}}\phi_1 \\ \sqrt{\frac{1}{2!}}\phi_1^2 \\ \sqrt{\frac{1}{3!}}\phi_1^3 \\ \vdots \end{bmatrix} \cdot \begin{bmatrix} 1 \\ \sqrt{\frac{1}{1!}}\phi_2 \\ \sqrt{\frac{1}{2!}}\phi_2^2 \\ \sqrt{\frac{1}{3!}}\phi_2^3 \\ \vdots \end{bmatrix} \tag{3.46}$$

Thus, we can actually rewrite the last factor of the kernel function in Eq. 3.43 as a dot product of infinite dimensional vectors. To write the entire kernel function as a dot product, we must include the square root of the first factor $\exp\left(-\frac{1}{2}\left(\phi_1^2 + \phi_2^2\right)\right)$ into the vectors in Eq. 3.46. For readability, define $c = \sqrt{\exp\left(-\frac{1}{2}\left(\phi_1^2 + \phi_2^2\right)\right)}$. The full dot-product representation for the kernel function then becomes

$$\kappa(\phi_1, \phi_2) = \begin{bmatrix} c \\ c\sqrt{\frac{1}{1!}}\phi_1 \\ c\sqrt{\frac{1}{2!}}\phi_1^2 \\ c\sqrt{\frac{1}{3!}}\phi_1^3 \\ \vdots \end{bmatrix} \cdot \begin{bmatrix} c \\ c\sqrt{\frac{1}{1!}}\phi_2 \\ c\sqrt{\frac{1}{2!}}\phi_2^2 \\ c\sqrt{\frac{1}{3!}}\phi_2^3 \\ \vdots \end{bmatrix} \tag{3.47}$$

From this form, it becomes clear that the kernel function implicitly calculates the dot product between features in the high-dimensional space. Thus, the great advantage of the kernel trick is that the mapping to the high-dimensional space in not performed explicitly, hence does not require the enormous computational power that the lifting trick does. Moreover, we do not even need to know the mapping that is needed to transform the dataset to the higher-dimensional space, since we use infinite dimensional dot-products[30].

Feature extraction

The KEM method defines a set of feature vectors $\{\phi_j\}_{j=1}^{J}$, where every vector is related to a single voxel. The feature vectors contain prior information about the activity in that specific voxel. This prior information could come from projections from earlier or later time-frames in case of dynamic SPECT, or from for example the data of a simultaneous MRI-scan in case of multi-modal imaging.

Kernel function

The label corresponding to feature vector ϕ_j is the unknown activity f_j. KEM assumes that every f_j can be described as some function Γ of its corresponding feature vector ϕ_j (Eq. 3.48).

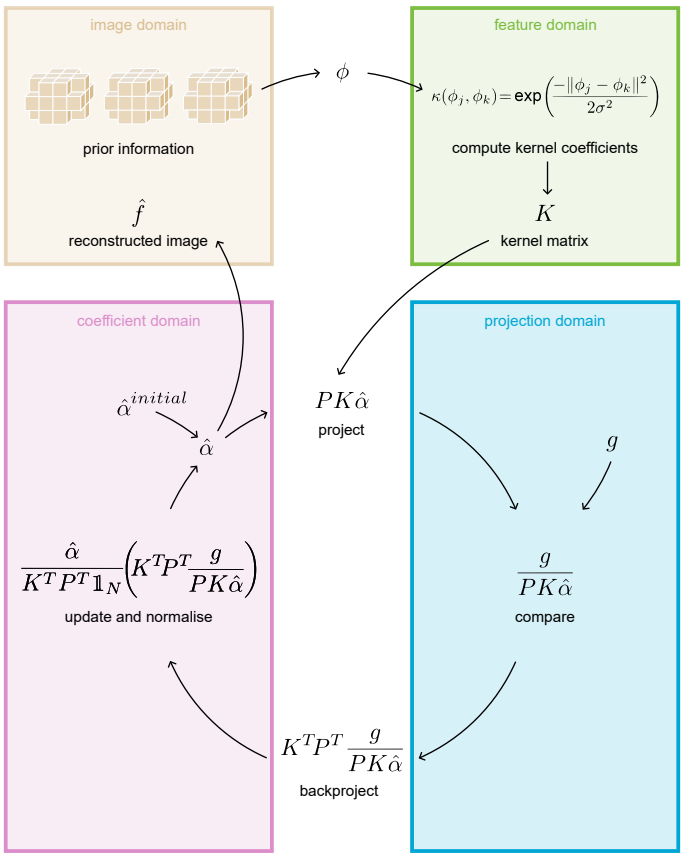


Figure 3.5: The KEM algorithm displayed in separate steps. The inputs are time-binned MLEM reconstructions (used to construct the kernel matrix), $\hat{\alpha}^0$ (the initial guess for the coefficient image) and g (the recorded projection).

$$f_j = \Gamma(\phi_j) \tag{3.48}$$

Generally, a highly complex and non-linear function Γ is needed to describe the activities accurately. Thus, a simple linear model in the feature space, such as $\Gamma(\phi_j) = w^T\phi_j$ is usually not adequate. In the same spirit as for Support Vector Machines, the 'lifting trick' can be used on the features to find a linear expression for Γ. The vector-valued function $\Phi : \mathbb{R}^B \rightarrow \mathbb{R}^N$ maps the feature vectors from their original low-dimensional space to a high or even infinitely dimensional feature space $\mathbb{R}^N$. The feature mapping is chosen such that Γ becomes a linear function of the feature vectors in the higher-dimensional feature space.

$$f_j = \Gamma(\phi_j) = v^T\Phi(\phi_j) \tag{3.49}$$

In Eq. 3.49, $v \in \mathbb{R}^N$ is a vector in the higher-dimensional feature space, which can be written as a weighted sum of all j higher-dimensional features $\Phi(\phi_j)$.

$$v = \sum_{l=1}^{N} \alpha_l \Phi(\phi_l) \tag{3.50}$$

Substituting 3.50 into 3.49 results in a formula that shows that the activity in voxel j can be written as a weighted sum of high-dimensional dot products.

$$f_j = \Gamma(\phi_j) = \sum_{l=1}^{N} \alpha_l \Phi(\phi_l)^T\Phi(\phi_j) \tag{3.51}$$

The feature mapping Φ is unknown, however, and additionally it tends to be very high dimensional. This makes the computations very computationally demanding. In the same way as shown earlier, this problem is averted by applying the 'kernel trick'. The inner products of the unknown high-dimensional features are then replaced by a kernel function of the original features.

$$f_j = \sum_{l=1}^{N} \alpha_l \kappa(\phi_l, \phi_j) \tag{3.52}$$

This summation can be summarised for the whole set of voxels in the vector-matrix product 3.53.

$$f = K\alpha \tag{3.53}$$

In this equation, $K \in \mathbb{R}^{J\times J}$ is the kernel matrix, where element (l, j) is equal to $\kappa(\phi_l, \phi_j)$, and $\alpha \in \mathbb{R}^J$ is the kernel coefficient vector. Using this expression in the forward model that was introduced in Eq. 2.3 yields the kernel-based forward model.

$$g = PK\alpha \tag{3.54}$$

Update equation for KEM

In a similar fashion to MLEM, the KEM algorithm is aimed at finding a kernel coefficient vector $\hat{\alpha}^{\text{KEM}}$ that maximises the expectation of the Poisson-likelihood of the observed data. In order to obtain physically valid solutions only, the solution space is limited to $A = \{\alpha \in \mathbb{R}^J | \alpha_j \geq 0, \forall j = 1, ..., J\}$.

$$\hat{\alpha}^{\text{KEM}} = \underset{\alpha\in A}{\arg\max} L(g|K\alpha) \tag{3.55}$$

Once again, an iterative update formula is used to estimate $\hat{\alpha}^{\text{KEM}}$:

$$\hat{\alpha}^{n+1} = \frac{\hat{\alpha}^n}{K^TP^T\mathbb{1}_I}\left(K^TP^T\frac{g}{KP\hat{\alpha}^n}\right) \tag{3.56}$$

In order to find the KEM-estimate of the activity $\hat{f}^{\text{KEM}}$ from the α^{KEM}, the following equation is used.

$$\hat{f}^{\text{KEM}} = K\hat{\alpha}^{\text{KEM}} \tag{3.57}$$

It is interesting to note that the KEM algorithm reduces to the MLEM algorithm if the K-matrix is replaced by the J-dimensional identity matrix.

4

Practical Implementation

The implementation of the reconstruction methods was done in Python 3. An annotated excerpt of the source code that was written for this paper is included in the appendix.

4.1. Simulated Scanner

The three-headed U-SPECT scanner with a multi-pinhole collimator as described in Section 2.2 was used in the simulations. Its system matrix, which is used for real experimental purposes, was provided. In this matrix, the scanner volume is discretised into 105 by 105 by 79 cubic voxels of size 0.75 mm. The three detector plates each measure 464 by 383 pixels, where a pixel is 1.1 by 1.1 mm. This implies that there are over $4.5 \cdot 10^{11}$ possible voxel-pixel pairs. Storing all the elements of the system matrix in the memory of a computer and performing matrix-multiplications with it impractical and impossible at worst in many cases due to memory issues. The full ('dense') representation is redundant, since the majority of emitted photons are blocked by the pinhole collimator. This is shown in Fig. 4.2, which displays the simulated projection of the phantom in Fig. 4.1. This leads to many zero-elements in the matrix, making it a natural choice to save P in a sparse format. Oftentimes, also a threshold is used to further decrease the matrix size, by eliminating voxel-pixel combinations which have a detection probability below the threshold between them[5]. The probabilities in this matrix take into account penetration of the pinholes, but scatters are not included in the model.

4.2. Simulated Phantoms

Two dynamic phantoms were simulated to test the performance of the reconstruction algorithms. The difference between the two phantoms lies in the type of dynamic behaviour. The striatum phantom keeps the same relative shape throughout all time-frames, but decreases in total intensity as time progresses. In the hepatobiliary phantom, the tracer moves from one organ to the next, which causes changes in shape and total intensity of the activity inside the phantom.

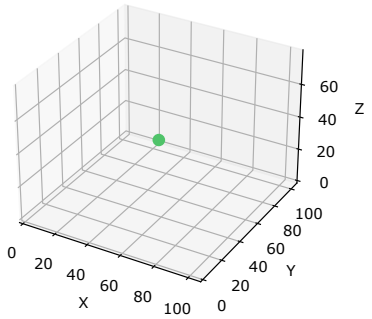


Figure 4.1: Example of a small spherical tracer distribution inside the U-SPECT scanner.

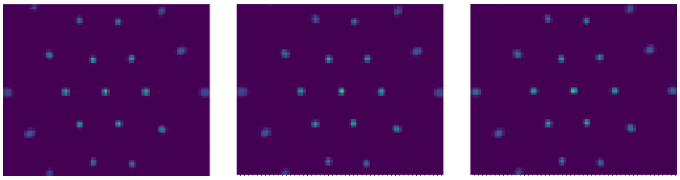


Figure 4.2: The projection of the spherical tracer distribution shown in Fig. 4.1.

4.2.1. Striatum phantom

This phantom is based on the paper 'Movies of dopamine transporter occupancy with ultra-high resolution focusing pinhole SPECT' by Vastenhouw et al.[33]. In this paper, U-SPECT was used to study the dynamics of dopamine transporter (DAT), a neurotransmitter which is suspected to play a key role in the degeneration in Parkinson's disease. The binding of DAT takes place in the striatum, a cluster of neurons in the forebrain, which is responsible for the reward system and the motor system[13].

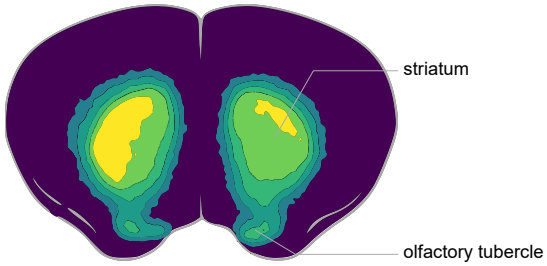


Figure 4.3: A labelled two-dimensional slice of the striatum phantom, including the olfactory tubercles. Adapted from Vastenhouw et al[33].

The method used was as follows: a small dosage of radiolabelled DAT was administered to laboratory mice. The researchers waited four hours to allow the tracer to spread evenly before injecting the mice with a dosage of cocaine. In order to observe the effects of the cocaine on the DAT binding, 45 second-long SPECT acquisitions were performed from 17 minutes before the cocaine-injection up to 28 minutes after the injection. MLEM was used to reconstruct each of these images, to create the movie that is featured in the title of their paper. The olfactory tubercle, a small sub-structure of the striatum, was of particular interest for the study.

For this paper, 3D series of dynamic phantoms with a similar anatomy were generated to compare the performance of the MLEM and KEM algorithm for SPECT image reconstruction. To this end, the autoradiographic image was taken from the paper by Vastenhouw et al. and traced using automatic tracing software. This created an image of the brain slice, including the olfactory tubercle, in six discrete intensity-levels. The activity-level was adjustable to enable both high-count and low-count simulations and to simulate the time-dependent decrease in DAT binding. A two-dimensional slice of the striatum phantom is shown in Fig. 4.3.

The dynamic series of phantoms used in this paper were made on the basis of the results displayed in Figure 1 [p. 985] of the aforementioned research. Here, it was noted that:

- The total activity in the left and right striatum fluctuated between 90% and 100% of its maximum value during the 17 minutes leading up to the cocaine injection until 3 minutes after the administration
- The total activity in the left and right striatum decreased approximately linearly from 100% at the 20-minute mark to 50% at the last frame at 45 minutes.

One hundred consecutive 45-second frames were simulated, a small selection with time-stamps and a graph with their respective total activities is shown in Fig. 4.4.

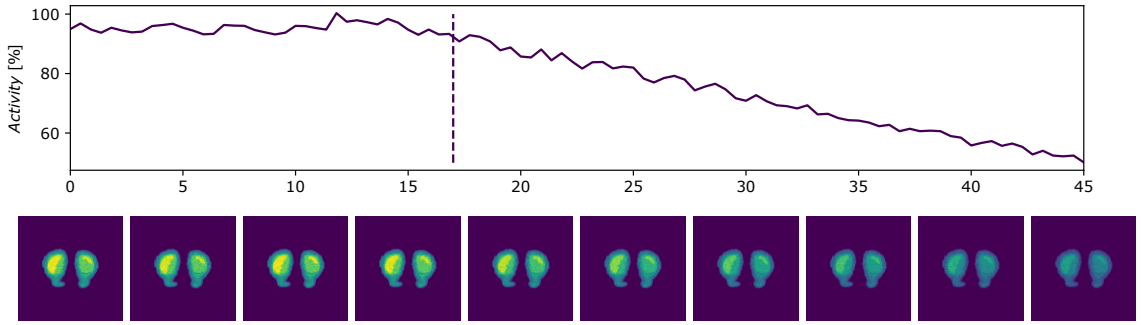


Figure 4.4: Top: Total activity in the left and right striatum, expressed as the percentage of the maximum activity over all frames. The dashed vertical line denotes the moment when the cocaine was administered. Bottom: Ten temporal frames of the simulated phantom, where the activity in each phantom corresponds to the time indicated straight above it.

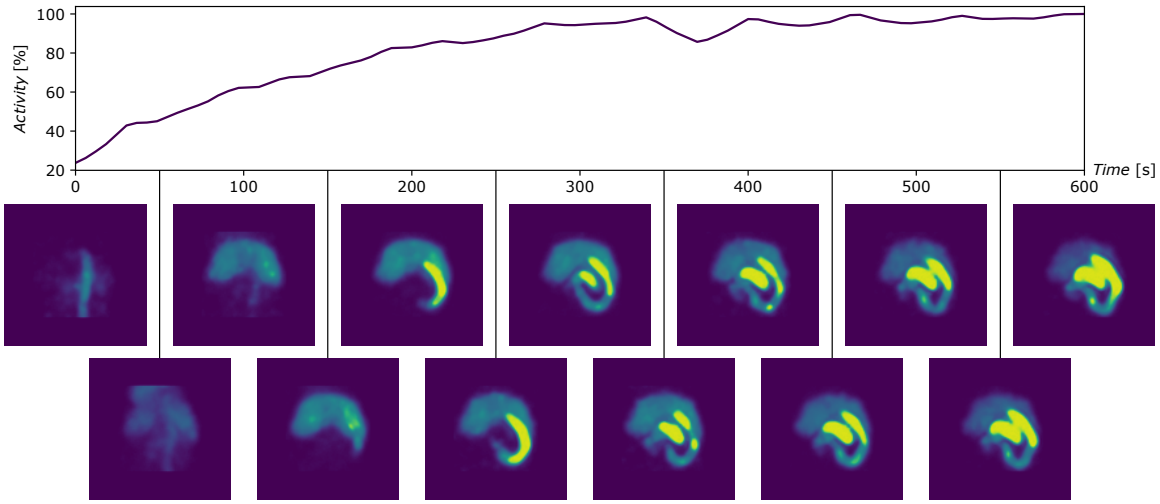


Figure 4.5: Top: Total activity in the phantom. Bottom: Thirteen temporal frames of the simulated phantom, where the activity in each phantom corresponds to the time indicated straight above it.

4.2.2. Hepatobiliary phantom

The second phantom was adapted from the paper 'Fast Spiral SPECT with Stationary γ-Cameras and Focusing Pinholes' by Vaissier et al.[32]. This research used in vivo dynamic scans of the thorax and abdomen of a mouse. For this, the radioligand ^{99m}Tc-mebrofenin was injected in the tail vein. From there, it circulated to the liver and subsequently secreted to the bowel. The goal of this scan was to create a series of images that show the hepatobiliary tracts that connect the liver, gallbladder and bile. As opposed to the striatum phantom, this phantom is highly-dynamic in shape due to the excretion of the tracer.

The scanning started at the instant that the tracer was injected and finished 5 minutes later. The time-frames lasted 15 seconds each, resulting in a total of 20 frames. The reconstructions for a set of time-frames is shown in 4.5. These reconstructions were traced similarly to the striatum phantom to create phantoms that can be adjusted and used in the simulations in this paper.

4.3. Simulated Projections

The phantoms introduced in the previous section were used to create time-series of projections. This was done by taking each time-frame of the phantom and performing a forward projection using the system matrix. These are the expectations of the detected counts. The projections used for the recon-

structions in the simulations were obtained by taking a Poisson realisation of the forward projection. For the source code of this step, see the function `SimulateProjections` in line 236 in the Appendix.

4.4. Constructing the Kernel Matrix

To compile the kernel matrix, the prior information first needs to be packed into feature vectors. After that, the features are used to compute the coefficients in the kernel matrix using the kernel function.

Extracting feature vectors

To define the feature vectors for dynamic SPECT, Wang and Qi use MLEM reconstructions of summed dynamic data. To this end, they rebinned the full dataset, consisting of many timeframes, into T successive bins. For each bin of projections, the counts in each detector pixel are summed along the time-axis, resulting in T relatively high-count projections (left-hand side of Eq. 4.1).

$$\{g_t^{reb}\}_{t=1}^{T} \xrightarrow{\text{MLEM}} \{\hat{f}_t^{reb}\}_{t=1}^{T} \tag{4.1}$$

These composite images are then reconstructed using MLEM, which is expected to return acceptable reconstructions due to the increased number of counts (right-hand side of Eq. 4.1). The number of bins that will, in the end, result in the best KEM reconstruction is highly dependent on the dynamics of the tracer within the body. Selecting a high T will reduce the number of counts per rebinned MLEM reconstruction, which increases the noise present in the feature vectors. Choosing a low T, on the other hand, will not capture the temporal dynamic behaviour of the tracer, since the projections are summed over extended periods of time.

Each feature vector ϕ_j contains prior information about the activity in voxel j. The set of features $\{\phi_j\}_{j=1}^{J}$ consists of T-dimensional vectors whose values are the MLEM-reconstructed activities during each of the three time-bins for voxel j. The source code that is responsible for obtaining the feature vectors is included in the Appendix in the functions `RebinProjections` and `ReconstructRebinnedProjections`.

$$\phi_j = \begin{bmatrix} \hat{f}_{1,j}^{reb} \\ \hat{f}_{2,j}^{reb} \\ \vdots \\ \hat{f}_{T,j}^{reb} \end{bmatrix} \tag{4.2}$$

Kernel function

As mentioned, the (j,l)-th element of K is $\kappa(\phi_j, \phi_l)$. A variety of kernel functions κ exist, the most popular one being the Radial Basis Kernel. This kernel is a measure for similarity between two features. Using the formula for the RBK (Eq. 4.4) directly, however, would result in a very large, dense matrix. This would make it infeasible to perform the update equations.

$$\kappa(\phi_j, \phi_l) = \exp\left(\frac{-\|\phi_j - \phi_l\|^2}{2\sigma^2}\right) \tag{4.3}$$

For this reason, the k-nearest-neighbour (kNN) model is employed to make the K-matrix sparse and thereby reduce computation time and required memory. The k-nearest-neighbour model finds, for a feature ϕ_j, the k features that are closest in terms of Euclidean distance. These nearest neighbours are incorporated in the RBK for KEM, where the value zero is assigned to any feature that is not within the kNNs.

$$\kappa(\phi_j, \phi_l) = \begin{cases} \exp\left(\frac{-\|\phi_j - \phi_l\|^2}{2\sigma^2}\right), & \text{if } \phi_l \in k\text{NN of } \phi_j \\ 0, & \text{otherwise} \end{cases} \tag{4.4}$$

The choice of k in the construction of the kernel matrix is important. Choosing a value that is too small will diminish the advantages that KEM offers over MLEM for reconstruction of noisy, low-count projection. Making k too large, on the other hand, will oversmooth small structures in the image that actually have less than k similar pixels.

Finding the k-nearest-neighbours for some feature vector and its kernel with each neighbour is quite a heavy and time-consuming computation. Fortunately, this step is an independent process for each of the J features vectors. Thus, parallel computing could be used to accelerate this step. To do this, the

set of feature vectors was split up in smaller batches. The Multiprocessing package was used to assign each batch to a different core of the computer's processor. The results of each core are combined then to construct the kernel matrix.

4.5. Simulation parameters

In each simulation, some parameter is adjusted to study its effects on the MSE and SNR of the reconstruction. The resulting KEM-images are benchmarked against MLEM with a three-dimensional Gaussian post-filter of standard deviation of 1 voxel.

A series of eight simulations was performed, using the inputs displayed in Table 4.1. Simulations 3, 4, 7 and 8 serve to show the effects of choosing a different k or T, whereas simulations 1, 2, 6 and 7 show the method's robustness to noise, since the relative noise is much lower in high-count projections.

	Phantom	Description	Counts	k	T
1	Striatum	High count	10^{12}	48	3
2	Striatum	Low count	10^{7}	48	3
3	Striatum	High k	10^{7}	576	3
4	Striatum	Low k	10^{7}	4	3
5	Hepatobiliary	High count	10^{12}	48	3
6	Hepatobiliary	Low count	10^{7}	48	3
7	Hepatobiliary	Many bins	10^{7}	48	10
8	Hepatobiliary	Few bins	10^{7}	48	1

Table 4.1: The simulations presented in this chapter. The description denotes the parameter that is changed from the values used by Wang and Qi. The Counts reflect the total number of counts recorded in all time-frames.

4.6. Performance Metrics

In present-day literature in the field of nuclear imaging, a variety of metrics are employed to quantify the quality of an image. Since in this research only simulations are used, the ground truth image is known. Thus, it is possible to use the Mean-Squared Error (MSE) as a performance metric. The formula to compute the MSE between the ground truth image f and the reconstruction $\tilde{f}$ is displayed in Eq. 4.5. The MSE is computed at every iteration of the reconstruction algorithm to also monitor the convergence to or divergence from the ground truth image.

$$MSE_{dB}(\hat{f}, f) = 10 \log_{10} \left(\frac{\|f - \tilde{f}\|^2}{N^n} \right) (dB) \tag{4.5}$$

It is interesting to look at the signal-to-noise ratio (SNR), since MLEM is known to converge more slowly in regions with low activity. By monitoring the SNR, it is possible to study the convergence rate of a reconstruction algorithm in hot and cold parts of the phantom. The SNR is computed by selecting a large number of voxels in a cold region and finding the mean activity $\hat{N}$. The same is done to find the average activity $\hat{S}$ for a set of voxels in a hot region. The regions used for the computation are shown in Fig. 4.6 The SNR is computed with the formula in Eq. 4.6. The SNR is also computed at every iteration of the algorithm.

$$SNR_{dB}(\hat{f}) = 10 \log \left(\frac{\hat{S}}{\hat{N}} \right) (dB) \tag{4.6}$$

Naturally, the quality of a reconstruction can also be judged by simply inspecting the image produced. While the reconstructions are three-dimensional, it is customary to view two-dimensional slices rather than a volumetric render. In the next chapter, the central slice of MLEM and KEM from three different time-frames are displayed for every simulation alongside the ground truth activity of the time-frame. This gives a general idea of how well each region was recovered, however it is difficult to distinguish the differences between images quantitively.

For this reason, it is also useful to inspect the (reconstructed) activity along a single line through the scanner. These so-called profile lines give more insight into the differences and similarities between

Figure 4.6: The two regions of the 50^{th} frame of hepatobiliary phantom (left) and the striatum phantom (right). Yellow areas denote the hot regions and purple areas denote cold regions.

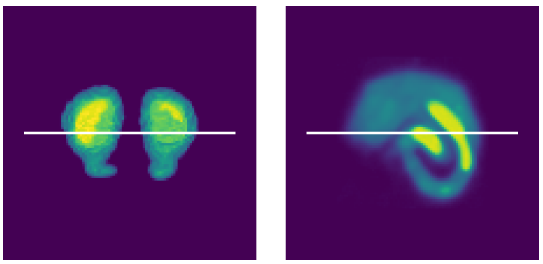


Figure 4.7: An example of the striatum phantom (left) and the hepatobiliary phantom (right) with the location of the profile line marked in white.

the reconstructions and the ground truths. The profile lines shown in the Results are drawn through the middle of the central slice, as depicted in Fig. 4.7.

In order to assess the performance of KEM and compare it to MLEM with a Gaussian post-filter, it is important to note that KEM is an adaptable method, in the sense that the algorithm uses a number of parameters as inputs. These parameters can be adjusted depending on the scan, by looking for example at the dynamics and shape of the phantom or the amount of counts recorded. The next section describes the simulations that were performed to assess KEM with a variety of inputs.

4.7. Computational Specifications

Due to the sheer size of the system matrix P and kernel matrix K, computations involving them are computationally demanding, even when performed using only sparse matrices. The sizes and densities of P and K are summarised in Table 4.2. The time it takes to calculate a forward or backward projection, operations which are both executed in every iteration of MLEM and KEM, increases rapidly with the number of pixels and voxels. Therefore, the high-performance computing cluster (HPC) of the TU Delft was used to execute the code that contains image reconstructing algorithms.

	Dimensions	Elements	Density
P	533136 by 870975	141647390	$3.1 \cdot 10^{-2}$ %
$K_{k=4}$	870975 by 870975	3483900	$4.6 \cdot 10^{-4}$ %
$K_{k=48}$	870975 by 870975	41806800	$5.5 \cdot 10^{-3}$ %
$K_{k=576}$	870975 by 870975	501681600	$6.6 \cdot 10^{-2}$ %

Table 4.2: The dimensions, number of non-zero elements and density of the matrices used in the simulations.

For sparse matrix computations, the SciPy Sparse module was used. This module offers a range of sparse data formats, such as Compressed Sparse Column, Compressed Sparse Row and List of Lists, each with their own advantages and disadvantages. Choosing the appropriate format can lead to significant speed-ups for matrix construction and matrix multiplications[34]. Nonetheless, constructing the kernel matrix can be a lengthy process, especially when k is large.

A simple method to reduce computation-time is by using that matrix multiplication is associative. While computing the KEM updates, the product $(PK)\hat{\alpha}$ takes much longer to execute than $P(K\hat{\alpha})$. This is due to the fact that it is more demanding to calculate the matrix-matrix product followed by a matrix-vector product than it is to perform two consecutive matrix-vector products. The iterations of both MLEM and KEM are also accelerated by computing the normalisation factors $K^T P^T \mathbb{1}_N$ (KEM) and $P^T \mathbb{1}_N$ (MLEM) only once instead of at every iteration, since it doesn't change unless P and K do.

5

Results

In this chapter, the results of the simulations described in 4.5 are presented. For each simulation, the MSE and SNR are plotted alongside the converged reconstructions and a profile line. The simulations are followed by a Discussion.

5.1. Striatum phantom with high counts

Simulation 1 used the default parameters of Table 4.1 on the striatum phantom in the high-count limit.

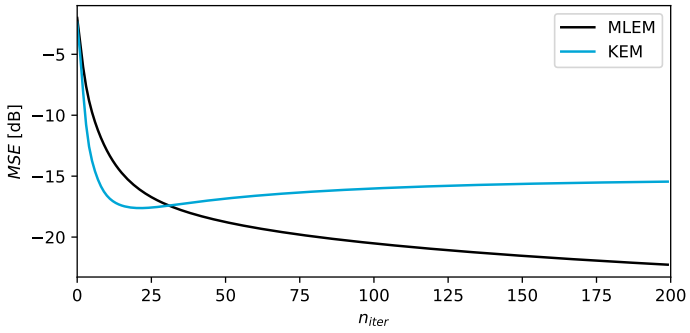


Figure 5.1: [Simulation 1] Mean Squared Error as a function of iteration number for KEM and MLEM reconstructions of the 50^{th} frame.The line indicating noise-free KEM coincides with the line for noisy KEM. The line indicating noise-free MLEM coincides with the line for noisy MLEM.

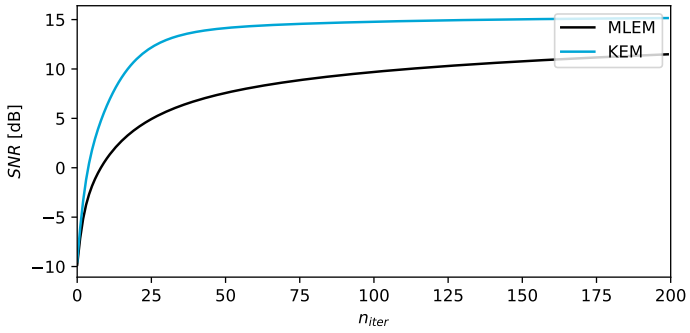


Figure 5.2: [Simulation 1] Signal-to-noise ratio as a function of iteration number for KEM and MLEM reconstructions of the 50^{th} frame.

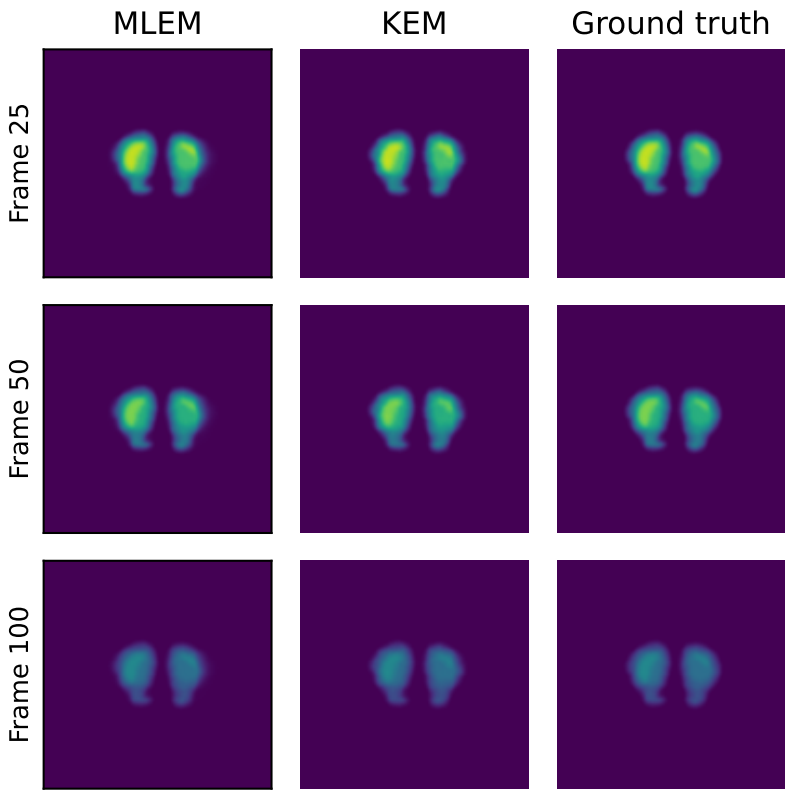


Figure 5.3: [Simulation 1] The MLEM and KEM reconstruction of frames 25, 50 and 100 after iteration 200, alongside the ground truth activities.

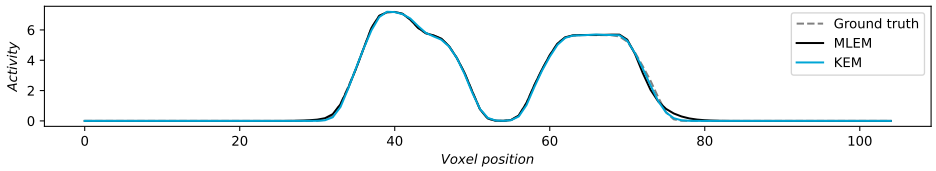


Figure 5.4: [Simulation 1] A profile line showing the (reconstructed) activity along the central horizontal line in frame 50 after iteration 200.

Simulation 1 shows KEM and MLEM reconstructions of the striatum phantom in the high-count limit. The images an dprofile line after 200 iterations (Fig. 5.3) look almost identical. The MSE of the MLEM reconstruction decreases gradually with the iterations, whereas the KEM image initially converges rapidly in terms of MSE and stabilises after roughly 20 iterations. The final MSE of MLEM is lower than that of KEM. The SNR, on the other hand, is higher for KEM.

5.2. Striatum phantom with low counts

Simulation 1 used the default parameters of Table 4.1 on the striatum phantom in the low-count limit.

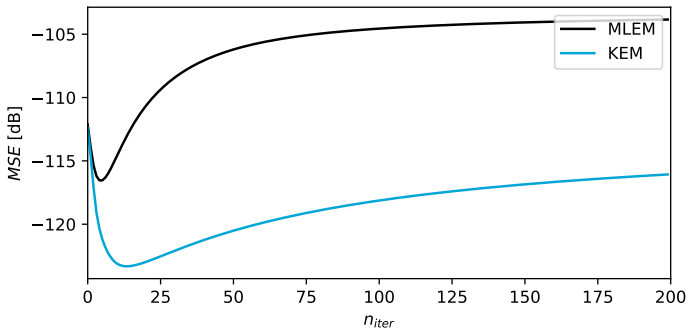


Figure 5.5: [Simulation 2] Mean Squared Error as a function of iteration number for KEM and MLEM reconstructions of the 50^{th} frame.

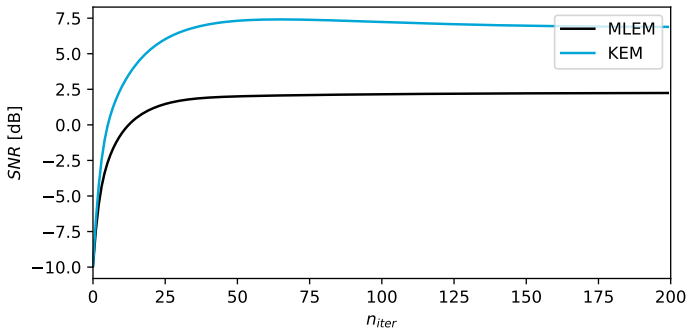


Figure 5.6: [Simulation 2] Signal-to-noise ratio as a function of iteration number for KEM and MLEM reconstructions of the 50^{th} frame.

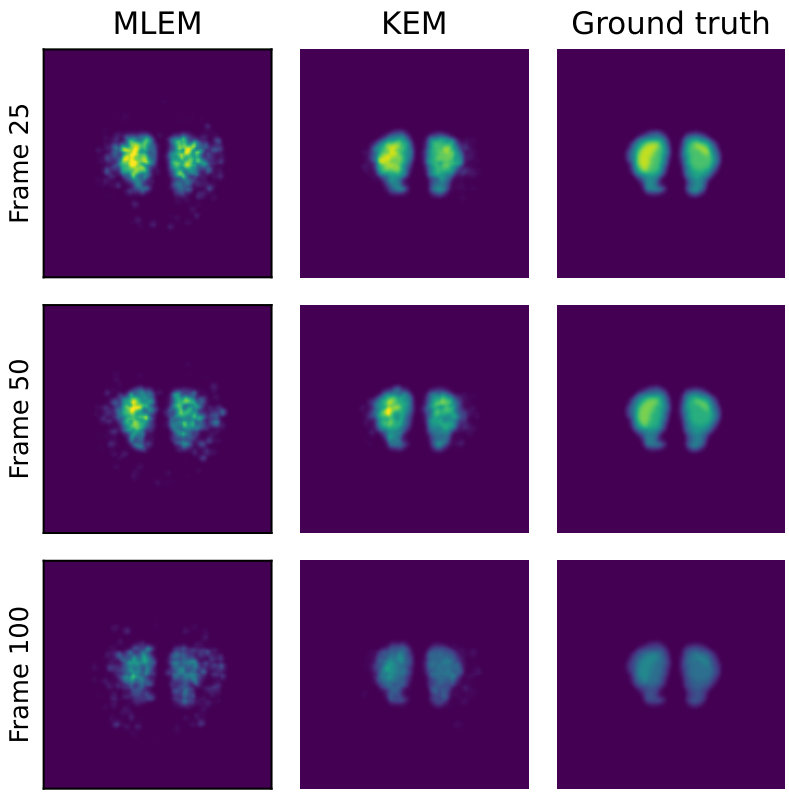


Figure 5.7: [Simulation 2] The MLEM and KEM reconstruction of frames 25, 50 and 100 after iteration 200, alongside the ground truth activities.

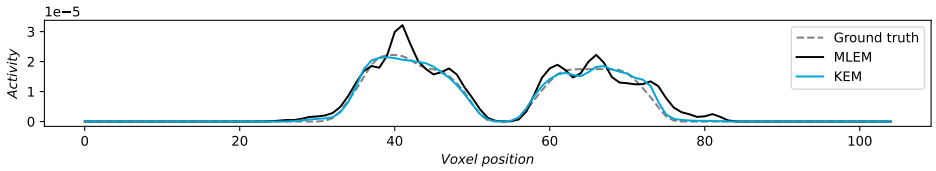


Figure 5.8: [Simulation 2] A profile line showing the (reconstructed) activity along the central horizontal line in frame 50 after iteration 200.

The final MLEM reconstructions are dominated by noise, both inside and outside the phantom. Its contour is visible, but the hot regions and the olfactory tubercle are not clearly distinguishable. The KEM image has very little noise outside the phantom and shows the tubercle and the hot spots in the left striatum. This is also reflected in the SNR, which is higher for KEM. The profile line of MLEM shows the checker-board effect, whereas the KEM line follows the ground truth more closely. The MSE of the KEM image degraded compared to the high count simulation, but is still lower than MLEM.

5.3. Striatum phantom with k=576

Simulation 3 used 576 nearest-neighbours for the K-matrix.

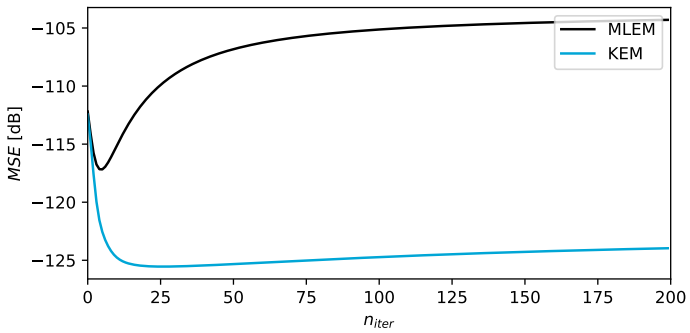


Figure 5.9: [Simulation 3] Mean Squared Error as a function of iteration number for KEM and MLEM reconstructions of the 50^{th} frame.

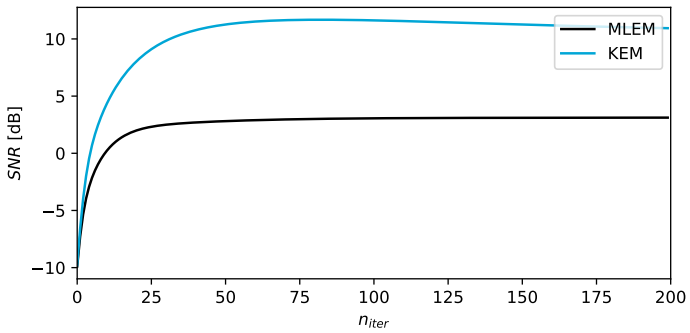


Figure 5.10: [Simulation 3] Signal-to-noise ratio as a function of iteration number for KEM and MLEM reconstructions of the 50^{th} frame.

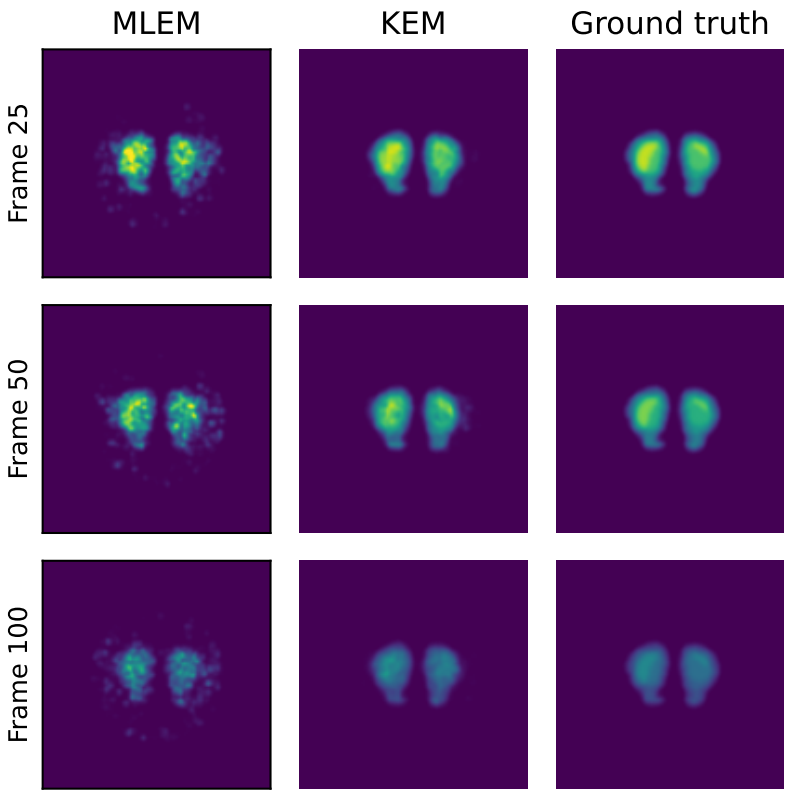


Figure 5.11: [Simulation 3] The MLEM and KEM reconstruction of frames 25, 50 and 100 after iteration 200, alongside the ground truth activities.

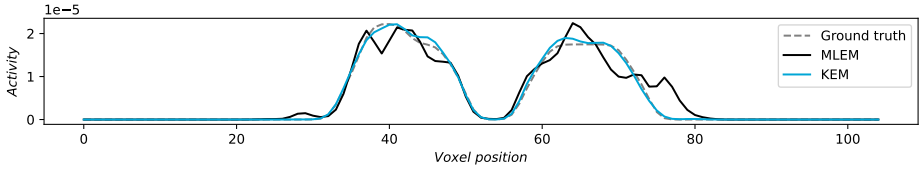


Figure 5.12: [Simulation 3] A profile line showing the (reconstructed) activity along the central horizontal line in frame 50 after iteration 200.

Increasing k resulted in a smoother image than Simulation 2. The background noise that was visible on the right side of the right striatum is no longer present in the KEM reconstruction. The SNR increased and the MSE does not degrade as much as in the previous simulation.

5.4. Striatum phantom with k=4

Simulation 4 used only 4 nearest-neighbours.

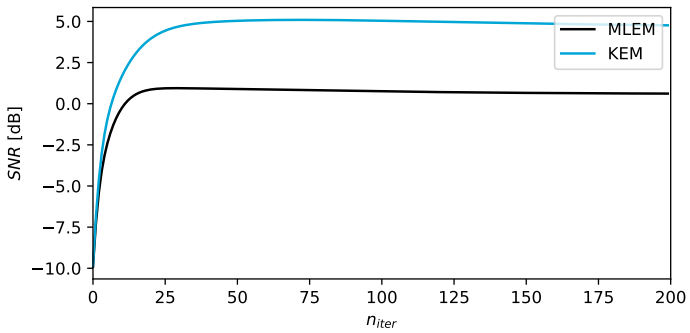


Figure 5.13: [Simulation 4] Signal-to-noise ratio as a function of iteration number for KEM and MLEM reconstructions of the 50^{th} frame.

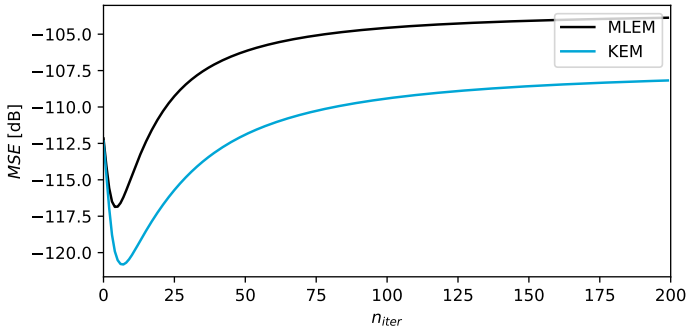


Figure 5.14: [Simulation 4] Mean Squared Error as a function of iteration number for KEM and MLEM reconstructions of the 50^{th} frame.

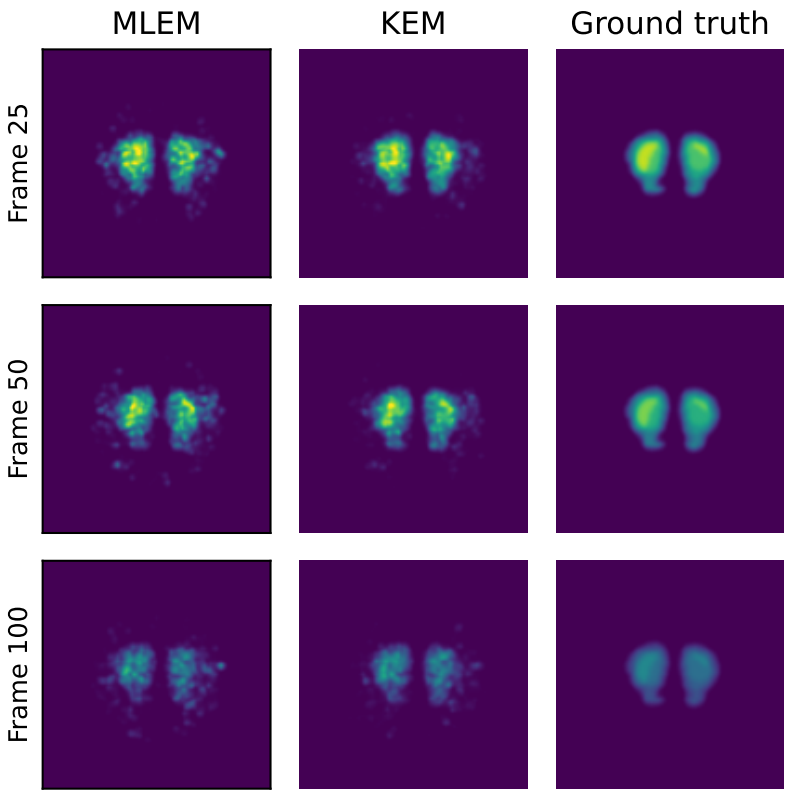


Figure 5.15: [Simulation 4] The MLEM and KEM reconstruction of frames 25, 50 and 100 after iteration 200, alongside the ground truth activities.

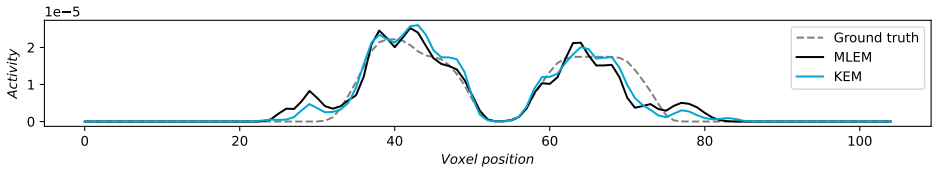


Figure 5.16: [Simulation 4] A profile line showing the (reconstructed) activity along the central horizontal line in frame 50 after iteration 200.

Where KEM was able to accurately reconstruct the hot regions in both striata in Simulation 2 and 3, the image produced by Simulation 4 is more speckled in the high-activity areas and also along the perimeter of the field of view. In the profile line, it is clear that KEM and MLEM both show the checkerboard effect. Thus, KEM lost some of its noise-reducing strength.

Despite the low number of nearest-neighbours, the K-matrix still caused KEM to result in images with a higher SNR and lower MSE.

5.5. Hepatobiliary phantom with high counts

Simulation 5 used the default parameters of Table 4.1 on the hepatobiliary phantom in the high-count limit.

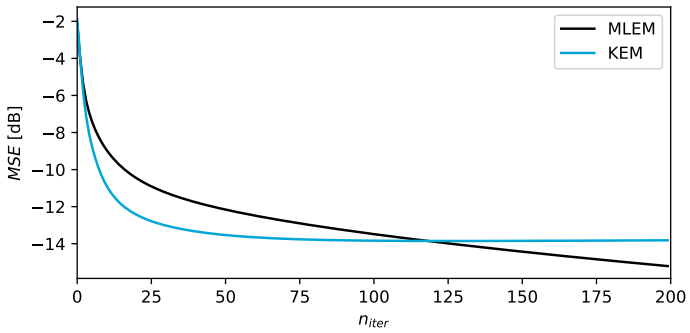


Figure 5.17: [Simulation 5] Mean Squared Error as a function of iteration number for KEM and MLEM reconstructions of the 50^{th} frame.

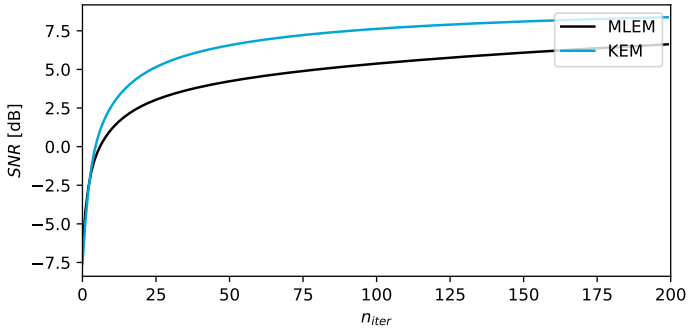


Figure 5.18: [Simulation 5] Signal-to-noise ratio as a function of iteration number for KEM and MLEM reconstructions of the 50^{th} frame.

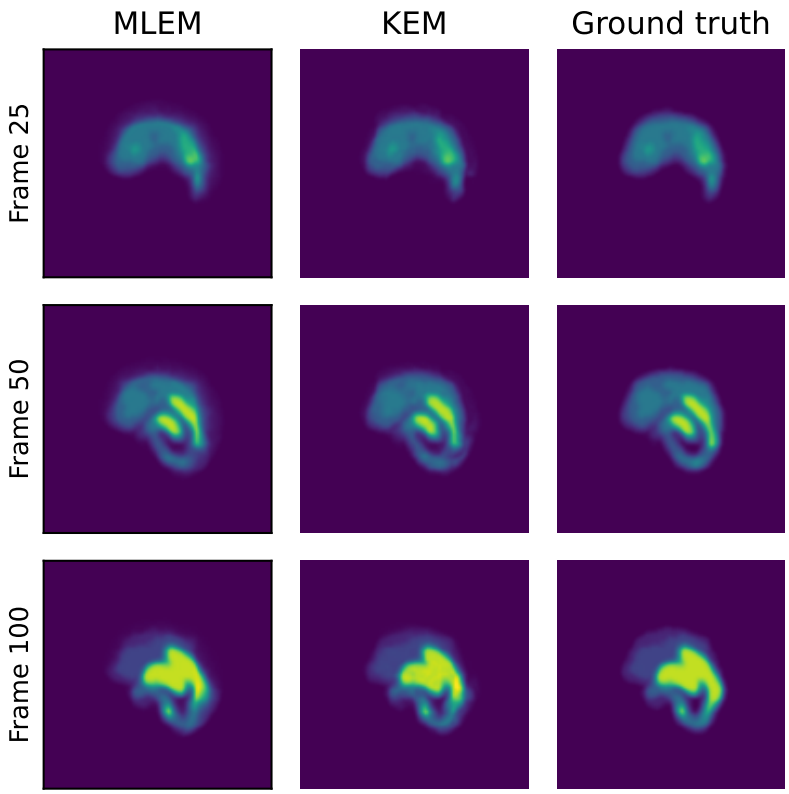


Figure 5.19: [Simulation 5] The MLEM and KEM reconstruction of frames 25, 50 and 100 after iteration 200, alongside the ground truth activities.

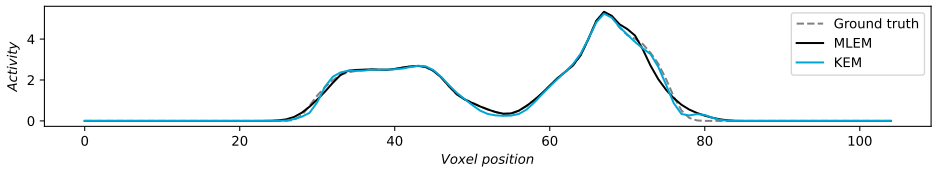


Figure 5.20: [Simulation 5] A profile line showing the (reconstructed) activity along the central horizontal line in frame 50 after iteration 200.

The images after 200 iterations (Fig. 5.19 once again look identical. The MSE of the MLEM reconstruction converges slowly and has not reached its limit after 200 iterations. The KEM image initially is constant after roughly 70 iterations. The fast MSE of MLEM is lower than that of KEM. The SNR, on the other hand, is higher for KEM.

5.6. Hepatobiliary phantom with low counts

Simulation 6 used the default parameters of Table 4.1 on the hepatobiliary phantom in the low-count limit.

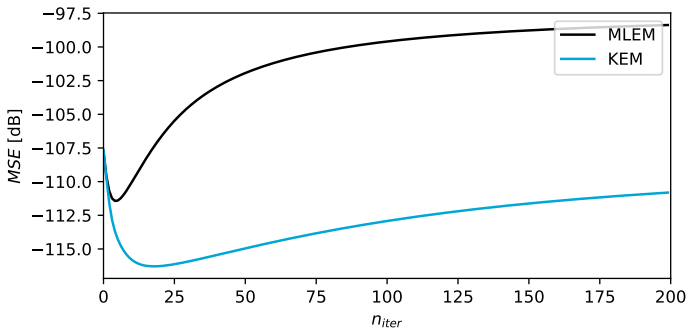


Figure 5.21: [Simulation 6] Mean Squared Error as a function of iteration number for KEM and MLEM reconstructions of the 50^{th} frame.

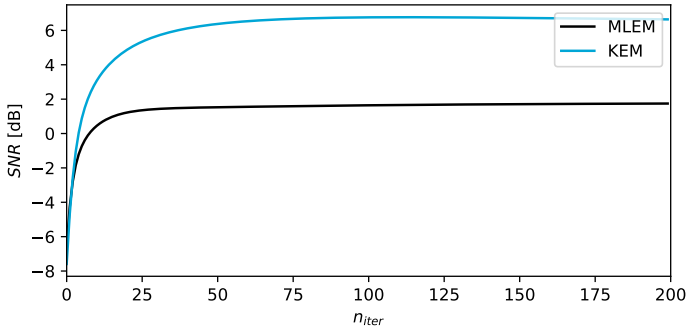


Figure 5.22: [Simulation 6] Signal-to-noise ratio as a function of iteration number for KEM and MLEM reconstructions of the 50^{th} frame.

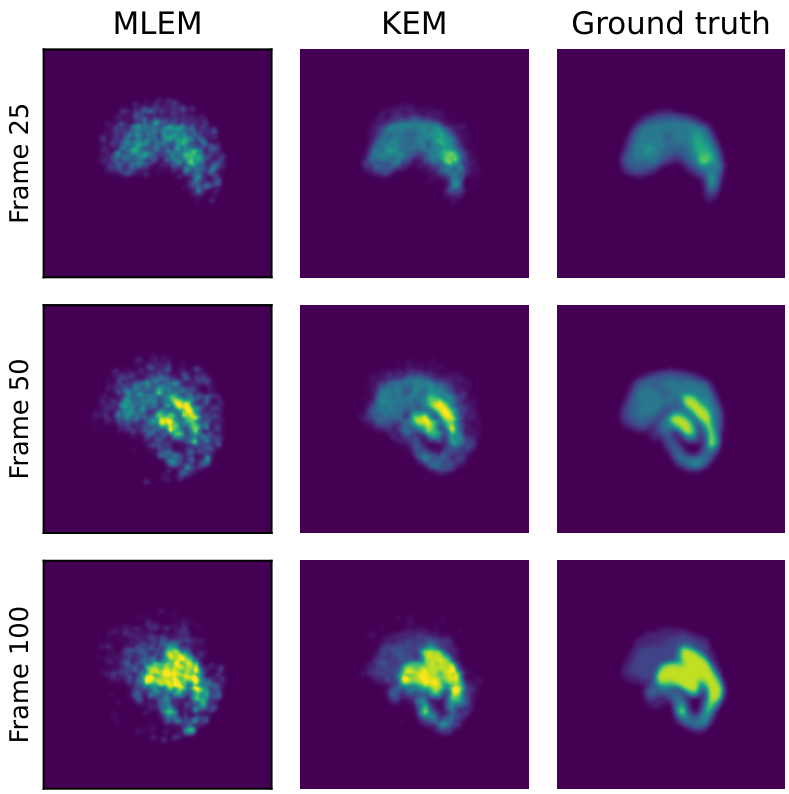


Figure 5.23: [Simulation 6] The MLEM and KEM reconstruction of frames 25, 50 and 100 after iteration 200, alongside the ground truth activities.

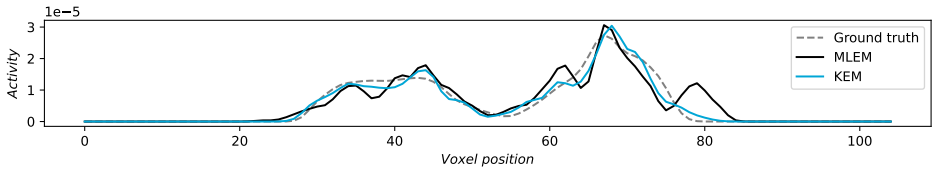


Figure 5.24: [Simulation 6] A profile line showing the (reconstructed) activity along the central horizontal line in frame 50 after iteration 200.

The MLEM image in the low-count limit looks speckled and shows a lot of noise along the perimeter of the field of view. The hot-regions are over-estimated and the structures in the low-activity regions are not visible in the final image. The KEM reconstruction is speckled, but the hot and cold regions are identifiable. The background noise is low, which is also reflected in the high SNR. KEM also outperforms MLEM in terms of MSE.

5.7. Hepatobiliary phantom with T=10

Simulation 7 used 10 time-bins for the K-matrix.

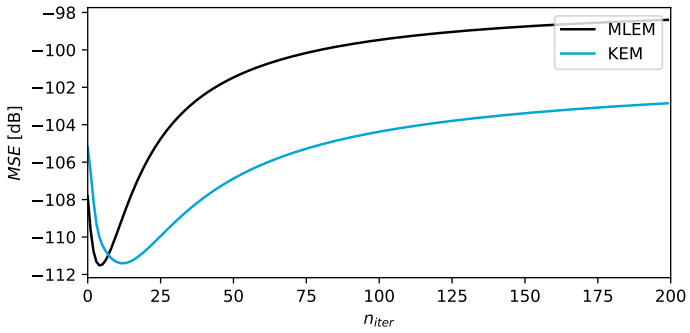


Figure 5.25: [Simulation 7] Mean Squared Error as a function of iteration number for KEM and MLEM reconstructions of the 50^{th} frame.

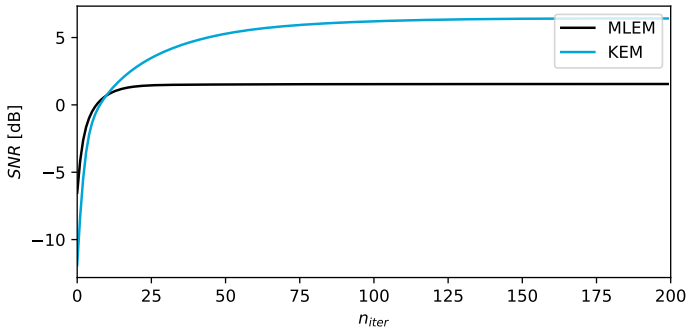


Figure 5.26: [Simulation 7] Signal-to-noise ratio as a function of iteration number for KEM and MLEM reconstructions of the 50^{th} frame.

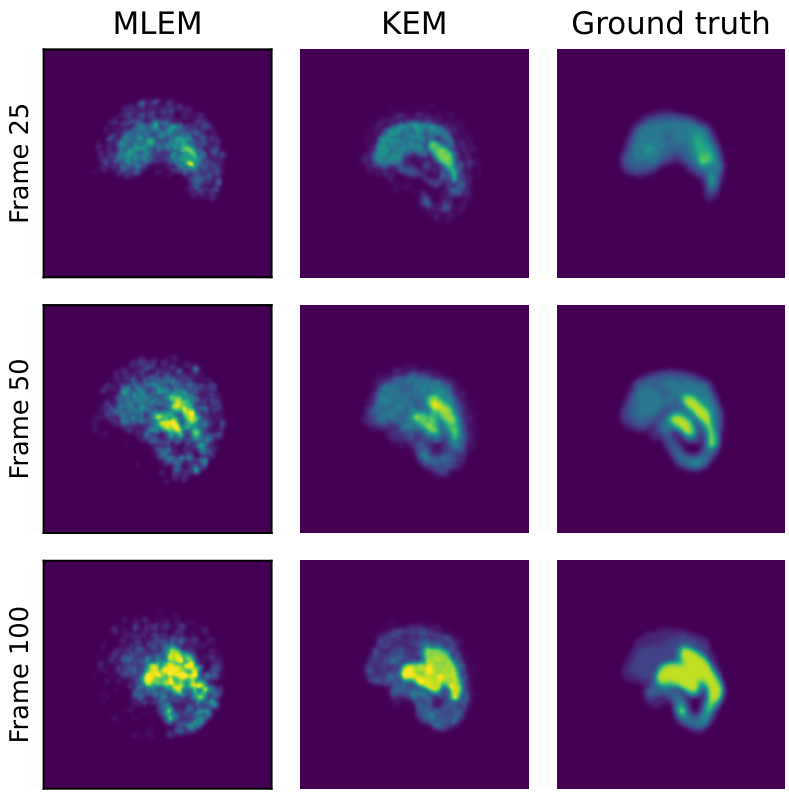


Figure 5.27: [Simulation 7] The MLEM and KEM reconstruction of frames 25, 50 and 100 after iteration 200, alongside the ground truth activities.

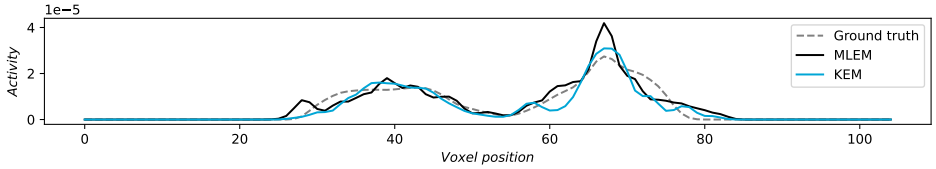


Figure 5.28: [Simulation 7] A profile line showing the (reconstructed) activity along the central horizontal line in frame 50 after iteration 200.

The KEM image degraded compared to the KEM image in simulation 6. The noise outside the phantom is still suppressed compared to MLEM and the silhouette of the phantom is reconstructed better.

5.8. Hepatobiliary phantom with T=1

Simulation 1 used 1 time-bin for the K-matrix.

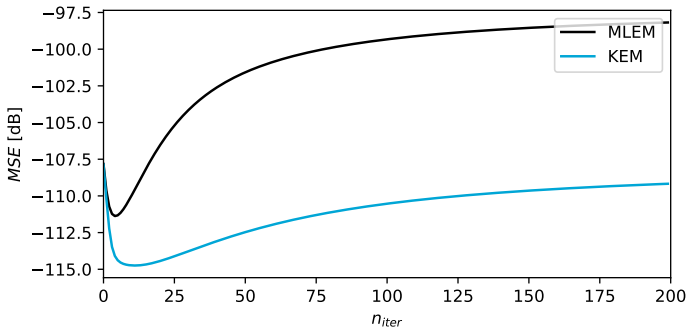


Figure 5.29: [Simulation 8] Mean Squared Error as a function of iteration number for KEM and MLEM reconstructions of the 50^{th} frame.

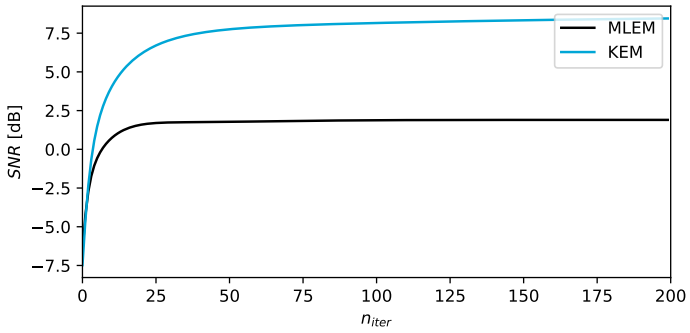


Figure 5.30: [Simulation 8] Signal-to-noise ratio as a function of iteration number for KEM and MLEM reconstructions of the 50^{th} frame.

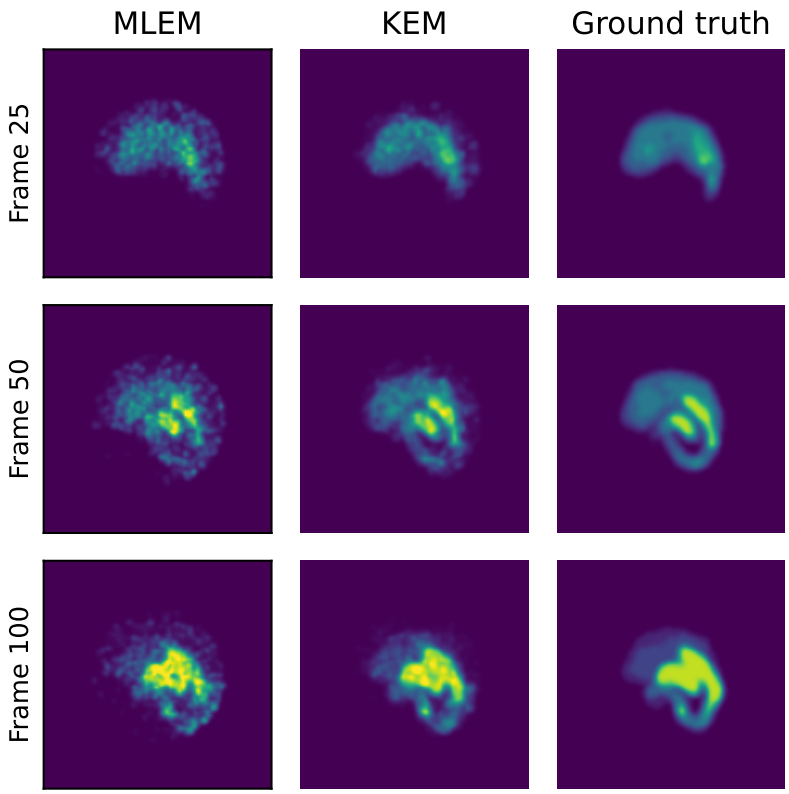


Figure 5.31: [Simulation 8] The MLEM and KEM reconstruction of frames 25, 50 and 100 after iteration 200, alongside the ground truth activities.

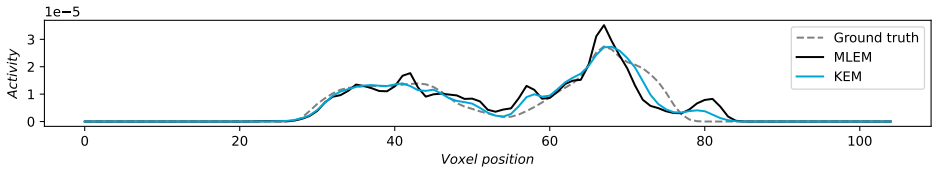


Figure 5.32: [Simulation 8] A profile line showing the (reconstructed) activity along the central horizontal line in frame 50 after iteration 200.

Simulation 8 shows that summing all projection into one bin causes small structures within the cold regions of the phantom to be over-smoothed. Nevertheless, KEM outperforms MLEM in both metrics.

5.9. Discussion

In the high-count simulations 1 and 5, the images from MLEM and KEM reconstructions look identical. KEM converged faster, but also starts to degrade, whereas MLEM has still not converged after 200 iterations. In the six low-count simulations, KEM outperforms MLEM in terms of MSE, SNR and by examining the produced images.

Decreasing k diminished the noise-suppresion of KEM, but has benefits when computations need to be fast or less intensive. The computing times are listed in Table 5.1. Assembling the sparse matrix takes roughly 70 times as long, and the full 200 iterations of the KEM reconstruction time is increased almost tenfold.

	k=4	k=48	k=576
Finding nearest-neighbours	214.4	217.6	273.7
Assembling matrix	31.6	141.4	2133.1
Saving matrix to disk	3.0	21.2	311.4
200 MLEM iterations	92.8	92.2	103.6
200 KEM iterations	112.1	162.1	1060.2

Table 5.1: Comparison of computation times in seconds for k=4, k=48 and k=576 for constructing the K-matrix and performing the KEM-update equation.

The value k=48 found optimal by Wang and Qi for their PET reconstruction does not yield the best results in these simulations. The over-smoothing effects they encountered in their study were also not observed in this research, though simulations with values of k over 576 were not performed.

The number of bins T did, however, affect the smoothing that occurred in the KEM images. Using many bins deteriorated the MLEM reconstructions of the separate time-bins, since each bin contains fewer counts. This caused the KEM reconstructions to become noisier. When only one bin is used, the MLEM reconstruction of the single bin contains little noise, but also lacks the temporal behaviour of the activity. In the final KEM reconstruction, the noise is suppressed nicely, but the specific shapes that are characteristic to the time-frame are smudged. Moreover, increasing the dimensionality of the feature vectors by increasing the number of bins causes the assembly of the K-matrix to be prolonged.

	T=1	T=3	T=10
Finding nearest-neighbours	186.9	217.6	553.6
Assembling matrix	137.2	141.4	198.8
Saving matrix to disk	25.2	21.2	27.5
200 MLEM iterations	91.7	92.2	96.3
200 KEM iterations	163.8	162.1	179.8

Table 5.2: Comparison of computation times in seconds for T=1, T=3 and T=10 for constructing the K-matrix and performing the KEM-update equation.

6

Conclusion

In this paper, the Kernelised Expectation Maximisation algorithm proposed for dynamic PET by Wang and Qi (2015) was implemented for dynamic multi-pinhole SPECT. A series of computer simulations of dynamic phantoms were performed to study the mean-square error to the ground truth and signal-to-noise ratio of KEM reconstructions. The results for KEM were compared to conventional Maximum Likelihood Expectation Maximisation reconstructions of the same frame with Gaussian post-filter.

When high-count projections were used, the significance of noise is smaller, hence the quality of the MLEM reconstruction improves. MLEM performed better in the high-count regime than KEM with regard to MSE, but the KEM images have a better SNR. The differences between the images after 200 iterations are marginal and are invisible to the naked eye.

The simulations from low-counts projections show that KEM can outperform MLEM with a post-filter both in terms of MSE and SNR. The quality of the KEM images depends on a number of input parameters of the algorithm. Future research could perform a more exhaustive search of the KEM-parameter space to find a set of decision rules for choosing the optimal parameter values. Moreover, experiments with real experimental data could show whether KEM indeed has the potential to either increase the temporal resolution or decrease the tracer dosage needed to obtain dynamic SPECT images.

References

[1] "3. Tomography". In: *Mathematical Methods in Image Reconstruction*. Society for Industrial and Applied Mathematics, Jan. 2001, pp. 41–62.

[2] Jingke Zhang et al. "A General Framework for Inverse Problem Solving using Self-Supervised Deep Learning". In: *Department of Biomedical Engineering, School of Medicine, Tsinghua University, Beijing, China* (2021).

[3] Marcel Beister, Daniel Kolditz, and Willi A. Kalender. "Iterative reconstruction methods in X-ray CT". In: *Physica Medica* 28.2 (Apr. 2012), pp. 94–108. DOI: 10.1016/j.ejmp.2012.01.003. URL: https://doi.org/10.1016/j.ejmp.2012.01.003.

[4] Nicolai Bissantz, Bernard A. Mair, and Axel Munk. "A multi-scale stopping criterion for MLEM reconstructions in PET". In: *2006 IEEE Nuclear Science Symposium Conference Record*. IEEE, 2006. DOI: 10.1109/nssmic.2006.353726. URL: https://doi.org/10.1109/nssmic.2006.353726.

[5] Woutjan Branderhorst et al. "Targeted multi-pinhole SPECT". In: *European journal of nuclear medicine and molecular imaging* 38 (Nov. 2010), pp. 552–61. DOI: 10.1007/s00259-010-1637-4.

[6] Chia-Lin Chen et al. "Integration of SimSET photon history generator in GATE for efficient Monte Carlo simulations of pinhole SPECT". In: *Medical Physics* 35.7Part1 (June 2008), pp. 3278–3284. DOI: 10.1118/1.2940159. URL: https://doi.org/10.1118/1.2940159.

[7] Yuan Chen et al. "Optimized sampling for high resolution multi-pinhole brain SPECT with stationary detectors". In: *Physics in Medicine & Biology* 65.1 (Jan. 2020), p. 015002. DOI: 10.1088/1361-6560/ab5bc6. URL: https://dx.doi.org/10.1088/1361-6560/ab5bc6.

[8] Corinna Cortes and Vladimir Vapnik. In: *Machine Learning* 20.3 (1995), pp. 273–297. DOI: 10.1023/a:1022627411411. URL: https://doi.org/10.1023/a:1022627411411.

[9] Jianan Cui et al. "Deep reconstruction model for dynamic PET images". In: *PLOS ONE* 12.9 (Sept. 2017). Ed. by Li Zeng, e0184667. DOI: 10.1371/journal.pone.0184667. URL: https://doi.org/10.1371/journal.pone.0184667.

[10] Steven J. Duclos. *Scintillator Phosphors for Medical Imaging*. Summer. The Electrochemical Society, 1998.

[11] Alexander Etz. "Introduction to the Concept of Likelihood and Its Applications". In: *Advances in Methods and Practices in Psychological Science* 1.1 (2018), pp. 60–69. DOI: 10.1177/2515245917744314. eprint: https://doi.org/10.1177/2515245917744314. URL: https://doi.org/10.1177/2515245917744314.

[12] Ronald Sumida Farhad Daghighian and Michael E. Phelps. "PET Imaging: An Overview and Instrumentation". In: *Journal of Nuclear Medicine Technology* 18.1 (1990), pp. 5–13.

[13] H. Gray. *Anatomy of the human body*. 20th ed. ISBN: 1-58734-102-6. Philadelphia: Lea & Febiger, 1918.

[14] Gregory Gundersen. *Implicit lifting and the kernel trick*. 2021. URL: https://gregorygundersen.com/.

[15] Jacques Hadamard. *Lectures on Cauchy's problem in linear partial differential equations*. English. New York: Dover Publications V, 316 p. (1952). 1952.

[16] Hamamatsu. *Photomultiplier Tubes | Basics and Applications*. 4th ed. 2017.

[17] Frans van der Have et al. "U-SPECT-II: An Ultra-High-Resolution Device for Molecular Small-Animal Imaging". In: *Journal of Nuclear Medicine* 50.4 (Mar. 2009), pp. 599–605. DOI: 10.2967/jnumed.108.056606. URL: https://doi.org/10.2967/jnumed.108.056606.

[18] Ken Herrmann Hojjat Ahmadzadehfar Hans-Jürgen Biersack. *Clinical Applications of SPECT-CT*. 2nd ed. ISBN 978-3-030-65850-2. Deptartment of Nuclear Medicine Universitätsklinikum Essen: Publisher, 2022.

[19] Alfredo N. Iusem. "A SHORT CONVERGENCE PROOF OF THE EM ALGORITHM FOR A SPECIFIC POISSON MODEL". In: *Brazilian Journal of Probability and Statistics* 6.1 (1992), pp. 57–67. ISSN: 01030752, 23176199. URL: http://www.jstor.org/stable/43601445 (visited on 04/28/2023).

[20] James M. Joyce. *International Encyclopedia of Statistical Science*. Ed. by Miodrag Lovric. Springer Berlin Heidelberg, 2011. DOI: 10.1007/978-3-642-04898-2. URL: https://doi.org/10.1007/978-3-642-04898-2.

[21] H.S. Khare et al. "Comparison of static and dynamic cardiac perfusion thallium-201 SPECT". In: *IEEE Transactions on Nuclear Science* 48.3 (June 2001), pp. 774–779. DOI: 10.1109/23.940162. URL: https://doi.org/10.1109/23.940162.

[22] Torsten Kuwert. "Skeletal SPECT/CT: a review". In: *Clinical and Translational Imaging* 2 (2015), pp. 505–517.

[23] Siqi Li et al. *Neural KEM: A Kernel Method with Deep Coefficient Prior for PET Image Reconstruction*. 2022. DOI: 10.48550/ARXIV.2201.01443. URL: https://arxiv.org/abs/2201.01443.

[24] Alireza Sadremomtaz Mahsa Noori Asl. "Analytical image reconstruction methods in emission tomography". In: *Journal of Biomedical Science and Engineering* 6.1 (2013). Department of Physics, Faculty of Sciences, University of Guilan, Rasht, Iran.

[25] Committee on the Mathematics and Physics of Emerging Dynamic Biomedical Imaging. "Single Photon Emission Computed Tomography". In: *Mathematics and Physics of Emerging Biomedical Imaging* 5 (1996). Available from: https://www.ncbi.nlm.nih.gov/books/NBK232492/.

[26] *Medical imaging market size, share and growth analysis (Report ID: 978-1-68038-139-9)*. URL: https://www.grandviewresearch.com/industry-analysis/medical-imaging-systems-market.

[27] M Rentmeester, F Have, and Freek Beekman. "Optimizing multi-pinhole SPECT geometries using an analytical model". In: *Physics in medicine and biology* 52 (June 2007), pp. 2567–81. DOI: 10.1088/0031-9155/52/9/016.

[28] D Strul S Jan G Santin. "GATE: a simulation toolkit for PET and SPECT". In: *Physics in Medicine & Biology* (2004). DOI: 10.1088/0031-9155/49/19/007.

[29] A. Seret and J. Forthomme. "Comparison of different types of commercial filtered backprojection and ordered-subset expectation maximization SPECT reconstruction software". In: *Journal of Nuclear Medicine Technology* 37.3 (2009), pp. 179–187. DOI: 10.2967/jnmt.108.061275.

[30] Shai Shalev-Shwartz and Shai Ben-David. "Kernel Methods". In: *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014, pp. 179–189. DOI: 10.1017/CBO9781107298019.017.

[31] Pieter E. B. Vaissier. *Image acquisition and reconstruction in multi-pinhole emission tomography*. 2014.

[32] Pieter E.B. Vaissier et al. "Fast Spiral SPECT with Stationary γ-Cameras and Focusing Pinholes". In: *Journal of Nuclear Medicine* 53.8 (June 2012), pp. 1292–1299. DOI: 10.2967/jnumed.111.101899. URL: https://doi.org/10.2967/jnumed.111.101899.

[33] B Vastenhouw et al. "Movies of dopamine transporter occupancy with ultra-high resolution focusing pinhole SPECT". In: *Molecular Psychiatry* 12.11 (July 2007), pp. 984–987. DOI: 10.1038/sj.mp.4002028. URL: https://doi.org/10.1038/sj.mp.4002028.

[34] Pauli Virtanen et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[35] Guobao Wang and Jinyi Qi. "PET Image Reconstruction Using Kernel Method". In: *IEEE Transactions on Medical Imaging* 34.1 (Jan. 2015), pp. 61–71. DOI: 10.1109/tmi.2014.2343916. URL: https://doi.org/10.1109/tmi.2014.2343916.

[36] Gengsheng L. Zeng. “The ML-EM algorithm is not optimal for poisson noise”. In: *2015 IEEE Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC)*. IEEE, Oct. 2015. DOI: 10.1109/nssmic.2015.7582178. URL: https://doi.org/10.1109/nssmic.2015.7582178.

A

Source Code

This excerpt of the source code is included to show the steps that are taken in the reconstruction. Some functions, such as those responsible for the performance metrics and plotting the results, are omitted.

```
#CONSTANTS
J = 105 * 105 * 79 #number of voxels
N = 464 * 383 * 3  #number of pixels
sigma = 1

#SIMULATION SETTINGS
phantom_name = 'striatum'
k = 48
T = 3
counts = 10**7

#IMPORTS
import numpy as np
from scipy.sparse import csr_matrix, coo_matrix, load_npz
from scipy.ndimage import gaussian_filter
from multiprocessing import Pool, cpu_count
import matplotlib.pyplot as plt
import os
import scipy
from itertools import repeat
from sklearn.metrics import mean_squared_error
from sklearn.neighbors import NearestNeighbors


def MLEMIteration(x_n, y):
    """
    Parameters
    ----------
    x_n : numpy-array of length J
        estimate of the activity after n iterations.
    y : numpy-array of length N
        observed projection.

    Returns
    -------
    numpy-array of length J
        estimate of the activity after n+1 iterations.
    """
    return x_n/PT1N*((P.T).dot(y/P.dot(x_n)))

def ReconstructMLEM(N_iter, y, x_0):
    """
    Parameters
    ----------
    N_iter : integer
        number of MLEM iterations to be performed.
    y : numpy-array of length N
```

```
        observed projection.
    x_0 : numpy-array of length J
        initial estimate of the activity.

    Returns
    -------
    x_n : numpy-array of length J
        estimate of the activity after N_iter iterations.
    """
    x_n = x_0
    for n in range(N_iter+1):
        x_n = MLEMIteration(x_n, y)
    return x_n

def KEMIteration(a_n, y):
    """
    Parameters
    ----------
    a_n : numpy-array of length J
        estimate of the coefficient image after N_iter iterations.
    y : numpy-array of length N
        observed projection.

    Returns
    -------
    numpy-array of length J
        estimate of the coefficient image after n+1 iterations.
    """
    return a_n/KTPT1N*(K.T).dot((P.T).dot((y)/(P.dot(K.dot(a_n)))))

def ReconstructKEM(N_iter, y, a_0):
    """
    Parameters
    ----------
    N_iter : integer
        number of KEM iterations to be performed.
    y : numpy-array of length N
        observed projection.
    a_0 : numpy-array of length J
        initial estimate of the coefficient image.

    Returns
    -------
    x_n : numpy-array of length J
        estimate of the activity after N_iter iterations.
    """
    a_n = a_0
    for i in range(N_iter+1):
        a_n = KEMIteration(a_n, y)
    x_n = K.dot(a_n)
    return x_n

def kNNPool(X1, X, k = k):
    """
    Parameters
    ----------
    X1 : numpy-array of size (B,10000)
        subset of the feature vectors.
    X : numpy-array of size (B,J)
        set of all feature vectors.
    k : integer
        number of nearest-neighbors to be found. The default is sigma.

    Returns
    -------
    distances : numpy-array of size (10000,k)
        euclidean distance between a feature vector in X1 and its k nearest-neighbors.
    indices : numpy-array of size (10000,k)
        indices of the k nearest-neighbors of the feature vectors in X1.
    """
    nbrs = NearestNeighbors(n_neighbors=k).fit(X)
```

```
    distances, indices = nbrs.kneighbors(X1)
    return distances, indices

def SimulateProjections(phantom_name, counts):
    """
    Parameters
    ----------
    phantom_name : string
        name of the phantom, 'striatum' or 'hepatobiliary'.
        the file phantom_name.npy contains 100 frames of the phantom. the phantoms in the
            file are normalised, in such a way that the projection of all 100 frames results
            in an expected total projection of 1 count.
    counts : int
        desired number of counts in all 100 projections.

    Returns
    -------
    projections : numpy-array of size (100,N)
        the simulated projections of the desired phantom with the desired number of counts.
    """
    phantoms = np.load(phantom_name+'.npy', allow_pickle = True)
    phantoms = phantoms * counts
    projections = np.empty(len(phantoms), dtype = 'object')

    for i in range(len(phantoms)):
        expected_projection_i = P.dot(phantoms[i])
        projections[i] = np.random.poisson(expected_projection_i)

    return projections

def RebinProjections(T, projections):
    """
    Parameters
    ----------
    T : integer
        number of time-bins.
    projections : numpy-array of size (100, N)
        simulated projections of the phantom.

    Returns
    -------
    rebinned_projections : numpy-array of size (B, N)
        rebinned projections of the phantom.
    """
    bin_indices = np.linspace(0,100,T+1, dtype = 'int')
    rebinned_projections = np.empty(T, dtype = 'object')

    for i in range(T):
        start_ind = bin_indices[i]
        end_ind = bin_indices[i+1]
        rebinned_projections[i] = np.sum(projections[start_ind:end_ind], axis = 0)
    return rebinned_projections

def GaussianFilter(image, sigma = sigma):
    """
    Parameters
    ----------
    image : numpy-array of size J
        activity data.
    sigma : float, optional
        desired standard deviation of the gaussian filter that is applied. The default is
            sigma.

    Returns
    -------
    numpy-array of size J
        filtered activity data.
    """
    image = image.reshape((ActDimZ, ActDimY, ActDimX))
    return np.ravel(gaussian_filter(image, sigma = sigma, mode = 'constant', cval = 0))

```

```
def ReconstructRebinnedProjections(rebinned_projections, N_iter):
    """
    Parameters
    ----------
    rebinned_projections : rebinned_projections : numpy-array of size (B, N)
        rebinned projections of the phantom.
    N_iter : integer
        number of MLEM iterations to be performed.

    Returns
    -------
    features : numpy-array of size (J, T)
        the set of feature vectors.
    """
    #parallel MLEM reconstruction of the rebinned projections
    with Pool(cpu_count()-1) as p:
        features = np.array(p.starmap(ReconstructMLEM, zip(rebinned_projections, repeat(
            N_iter))))

    #apply gaussian filter to reconstructions
    for i in range(len(features)):
        features[i] = GaussianFilter(features[i], sigma = 1)

    #normalise the features
    features = np.array(features)
    features = np.nan_to_num(features)
    std_features = np.std(features, axis=1)
    features = np.dot(features.T , np.diag(1/std_features))

    return features

def ParallelkNNSearch(features):
    """
    Parameters
    ----------
    features : numpy-array of size (J, T)
        the set of feature vectors.

    Returns
    -------
    kNN_ind : numpy-array of size (J, k)
        indices of the k nearest-neighbors per feature vectors.
    kNN_dist : numpy-array of size (J, k)
        euclidean distance to its k nearest-neighbors per feature vector.
    """
    features_batched = np.split(features, 10000)

    with Pool(cpu_count() - 1) as p:
        features_kNN = np.array(p.starmap(kNNPool, zip(features_batched, repeat(features))))

    kNN_dist = features_kNN[:,0,:,:].reshape((J,k))
    kNN_ind = features_kNN[:,1,:,:].reshape((J,k))
    kNN_ind = np.array(kNN_ind, dtype = int)

    return kNN_ind, kNN_dist

def $\kappa$(dist):
    return np.exp (-*np.linalg.norm(dist)**2)

$\kappa$ = np.vectorize($\kappa$)

def ConstructKMatrix(kNN_ind, kNN_dist):
    """
    Parameters
    ----------
    kNN_ind : numpy-array of size (J, k)
        indices of the k nearest-neighbors per feature vectors.
    kNN_dist : numpy-array of size (J, k)
        euclidean distance to its k nearest-neighbors per feature vector.

    Returns
```

```
257     -------
258     K : sparse.csr_matrix
259         compressed sparse row (csr) matrix representation of the kernel matrix.
260     """
261     kNN_dist_radial = np.array(list(map (, kNN_dist)))
262     row = np.repeat(np.arange(0,J), k)
263     col = kNN_ind.ravel()
264     data = kNN_dist_radial.ravel()
265     K=coo_matrix((data, (row, col)),shape=(J,J))
266     K = K.tocsr()
267     return K
268 
269 if __name__ == '__main__':
270     #LOAD SYSTEM MATRIX
271     P = load_npz('system_matrix.npz'))
272 
273     #PRECOMPUTATIONS FOR MLEM
274     PT1N = (P.T).dot(np.ones(N))
275 
276     #CONSTRUCT K-MATRIX
277     projections = SimulateProjections (phantom_name, counts)
278     rebinned_projections = RebinProjections(T, projections)
279     features = ReconstructRebinnedProjections (rebinned_projections , N_iter)
280     kNN_ind , kNN_dist = ParallelkNNSearch (features)
281     K = ConstructKMatrix(kNN_ind , kNN_dist)
282 
283     #PRECOMPUTATIONS FOR KEM
284     PT1N = (P.T).dot(np.ones(N))
285     KTPT1N = (K.T).dot(PT1N)
286 
287     #RECONSTRUCTIONS
288     frame_nr = 50
289 
290     x_MLEM = ReconstructMLEM(N_iter, projections[frame_nr], x_0)
291     x_KEM = ReconstructKEM(N_iter, projections[frame_nr], a_0)
```