On the Detection and Correction of Transient Motions Before Reconstruction in Positron Emission Tomography

A thesis submitted for the degree of
Doctor of Philosophy
Trinity Term 2011

Peter Nordberg
Magdalen College

Supervised By
Prof. Sir J. Michael Brady FRS, FREng, FMedSci.
and
Dr.. Jérôme Decleck, PhD.

ABSTRACT

Positron Emission Tomography is a very useful form of functional imaging. It allows physiology to be examined in vivo, and so provides a useful complement to anatomical imaging modalities like CT or MRI. Motion poses serious problems to PET, especially in quantitative studies. The first effect of motion is blurring, especially as changes due to motion are confounded with those from the physiology. The second effect is to do with the corrections for attenuation and scatter required for quantitative studies, nowadays based on a CT scan acquired before the PET acquisition. A change of position means that corrections introduce artefacts. Lastly, modern reconstruction algorithms are nonlinear, meaning the overall image is not simply a combination of images corresponding to the different positions.

This thesis concerns a particular class of motions that often cause PET scans to be unusable, namely twitches, transient motion between comparatively stable positions. These motions can be corrected by registration. One registration method for rigid body motions, e.g. of the brain, is presented here. This method does not require the data to be reconstructed prior to reconstruction, saving computation, the need to choose reconstruction parameters, and avoiding any artefacts that result from the motion.

Registration to correct the motion is not possible without knowledge of when the position changes. A method to localise when twitches occur during a scan is considered to be the major contribution of the thesis. The novel method presented here is a significant improvement on existing approaches as it automated and does not require any additional equipment or steps to the imaging procedure. It too is performed before reconstruction as this allows performance that exceeds real-time and allows the unusually strong statistical properties of raw PET data to be used.

Combined together, the work presented in this thesis allows twitches to be located and corrected.

DECLARATIONS

This thesis is submitted to the Department of Engineering Science, University of Oxford, in partial fulfilment of the requirements of the degree of Doctor of Philosophy. The thesis is entirely my own work, and except where otherwise stated, describes my own research.

Peter Nordberg
Magdalen College, Oxford

The methods discussed in "Chapter III: Registration in the Sinogram Domain" have been presented in [57], and similar experiments were presented in [58].

At the time of writing patents are being filed in the United States and Great Britain to cover the content of "Chapter IV: Twitch Detection – Basic Method" and "Chapter V: Twitch Detection – Implementation".

ACKNOWLEDGEMENTS

My supervisors, Mike Brady and Jérôme Declerck both deserve more thanks than I can give here. Their advice and guidance have made this thesis possible, and their tolerance and support have made my time as a graduate student not only more enjoyable but also a lot more interesting... perhaps more so than it should have been!

I would also like to thank Veerle Kersemans, who provided me with the preclinical data without which this thesis would be significantly less compelling. In addition to many hours of work setting up and performing the scans she has had to put up with my occasionally incessant emails.

This work has been paid for by Siemens Molecular Imaging and an Industrial Fellowship from the Royal Commission for the Exhibition of 1851. At the time of writing this the Twitch Detection method is moving forward to full patent filings in the United States and United Kingdom, and I hope that Siemens are as pleased with that as I am. The support of the Royal Commission has been very generous and invaluable: without it this sort of slow, speculative research could not have been justified commercially with the eventual result far from guaranteed at the outset.

An apology, perhaps, is owed to my parents and brother; over the past years I have become more distant than I would have intended to be. Approaching what must be the last milestone on the road to being an independent adult I'm increasingly aware of what a good preparation I've had in life, and how much and often I've changed -- and had the chances to do so again and again, for which I'm very grateful.

Finally, I would like to acknowledged the Oxford University Lightweight Rowing Club... after all, it is arguably responsible to the fourth year of my time as a graduate student after attempt on the Lightweight Boat Race in 2009. And, I guess I should also thank our Cantabrigian friends, without whom there would be little point in having a race.

A NOTE ON CORRECTIONS

My viva was a very valuable process. (It was other things too: interesting, exciting, and gruelling, to name a few.) Solving the problems uncovered has made far more significant improvements to the results than anticipated. It has also provided an opportunity to further refine some of the arguments and, indeed, methods used in the experimental work.

I have attempted to indicate the changes made through the manuscript, if only to communicate some of the narrative of how the work was done and to acknowledge the contribution made by the process and my examiners.

As will become apparent this work falls broadly into two parts. The first is an exploration into using a registration method pioneered in CT to PET data. There were more substantial differences in the implementation than intended, as became apparent during the viva. These have been highlighted in the revised thesis, although the experimental work has not been repeated. The second, and major, part of this work concerns twitch detection. Here the corrections involve addressing an erroneous set of validity constrains applied to some calculations. The result of changing them is that the method produces much better results, and the empirical work for this part of the thesis has been redone as a result.

I would like to sincerely thank my examiners, Dr. Sean Smart and Prof. Johan Nuyts; as said, the corrections have significantly improved the results, and in doing so this thesis.

Table of Contents

Introduction

In which what is going to be said in this thesis, and has already been said in the abstract, is said. Subsequent chapters will say what is going to be said, so that the conclusion can, for a change, say what has been said.

Positron Emission Tomography is an exciting form of medical imaging that allows the physiological function of a subject to be imaged in vivo. As such it is a powerful tool both in a clinical setting for diagnosis, treatment planning and evaluation; or in research tasks.

As Chapter I will discuss, PET technology has developed dramatically in recent years, producing ever better quality images and functional information. This has been driven by continual improvement across the board: new materials and technology used in the hardware, improvement in the design and capabilities of the scanners, the ever increasing computational power available for reconstruction & analysis software, development of advanced reconstruction algorithms, and the increasing understanding of how functional imaging complements other information available to clinicians.

However, as the latter half of Chapter I argues, the effect of motion is becoming a serious limitation to the quality of the information that PET studies can deliver – if only because every other aspect of the field has been improving. Of the wide variety of motions that occur this thesis concentrates on ***twitches***: abrupt changes between comparatively stable positions. This type of motion is particularly damaging to the quality of the scan and, if the time of the motion were known, the effects reduced. Twitches can be corrected through registration – the estimation of a transformation that aligns two images such that the motion is undone.

After a brief interlude to discuss the possible sources of experimental data in Chapter II, a registration method is presented. During the early, independent, development of the method a similar method was found in the CT literature. This method has been developed and adapted for PET, and is explored in Chapter III.

This method is of special interest as it operates on the projection domain data, before it is reconstructed to form an image. This is a running theme of this thesis: there is no

information present in an image that is not in the original data (unless it is injected during reconstruction, for instance by some 'prior'). Therefore, information and tasks normally considered to be 'image domain' should generally be possible to perform, less – or more – easily, on the original data... it even might be possible to do so better. It is perhaps surprising that the medical imaging community seems to have a conceptual boundary at the point of reconstruction: research is often either done before that point (e.g. on scanner hardware or reconstruction algorithms) or after (segmentation, registration, quantification etc.), but rarely at the same time.

Of course, the registration method presented, like all other – and there are many – would only be of use in twitch correction if the times of twitches were known. The initial intention was to continue the development of the pre-reconstruction registration theme, moving on to non-rigid motions, but it became apparent that the salient issue is finding the times of twitches. Present solutions are not satisfactory – they either require manual observation of the subject during the scan or external monitoring equipment (which can be unreliable and inconvenient to set up, reducing the number of subjects that can be scanned each shift and adding stress to the subjects).

Chapter IV presents what is, therefore, the main contribution of this thesis: a means of detecting when twitches (or any transient discontinuity in the data) occur that requires no additional equipment or setup. Chapter V discusses the implementation of the basic method, and the present implementation already offers performance that exceeds that required for real time use several times over. Indeed, the implementation is intrinsically suited for online use, processing the stream of data produced by the scanner.

Chapter VI presents experimental studies that explore and demonstrate the potential of the method. Several different preclinical studies are used to show that transient motions can be found from the raw scanner data. Further developments of the method, and possible alternative uses for the basic concept, are presented in Chapter VII.

Finally, the thesis concludes with a presentation of an integrated demonstration of the methods developed. An abrupt change in position is found in a real, preclinical scan. The scan is split in two at this point and the motion estimated and removed using the pre-reconstruction registration method. The data is the then recombined and reconstructed to form a single, motion corrected image.

Chapter I: PET & Motion

In this chapter, Positron Emission Tomography is explored. In addition to making this thesis self contained the discussion highlights aspects of the modality that will be regarding the effects of motions. That will, in turn, be the subject of the second half of the chapter.

1. Positron Emission Tomography

The technology of PET dates back to 1975 (see [83] & [66]) when it was termed 'PETT' – Positron-Emission Transaxial Tomography, with practical systems available by 1978 [67].Thanks to some technical developments discussed in the coming pages, the capabilities and use of PET have expanded rapidly in recent years.

1.1 Functional Imaging

One of PET's great appeals is that it is a form of ***functional imaging***. That is in contrast to ***anatomical imaging***, which is how the term 'imaging' is probably more commonly understood. Computed Tomography ('CT') and Magnetic Resonance Imaging ('MRI' or 'MR') are the two or the more common forms of imaging, and both are primarily anatomical. CT is best briefly described as a 3D X-Ray, with a corresponding ability to image structures in the body, especially bones. MRI uses the effects of strong magnetic fields, in combination with radio-frequency energy, on protons to form an image based on the proton-density of materials – and, though it struggles with resolving bones, it is more capable than CT when imaging soft tissues as it gives much better contrast in those tissues.

Functional imaging is a different approach: rather than attempting to show the structures of the body, it aims to reveal something of how the body operates in physiological terms. It is still imaging because, in contrast to blood tests etc., it captures local variations within the body. The classic advertisement for functional imaging's potential is in oncology: cancer is a result of defective cellular behaviour, and anatomical modalities can not (directly) investigate that.

Isotope	Positron Range (mm)	Half Life (minutes)
^{11}C	1.1	20.3
^{13}N	1.4	9.97
^{15}O	1.5	2.03
^{18}F	1.0	109.8
^{68}Ga	1.7	67.8
^{82}Rb	1.7	1.26

Table 1 Properties of a selection of positron emitting radio isotopes suitable for use in PET tracers. Positron range is a random process and figure shown is the full width half maximum of a Gaussian fit, and is for the range in water. Information taken from [87].

2. Tracers & Positron Emission

The key to functional imaging is a ***tracer***. This is a molecule designed to be biologically significant, i.e. involved in the physiology being imaged, while also possible to track using a scanner. This could be a protein that binds to receptors on certain cells combined with a molecular structure that fluoresces (for optical fluorescence microscopy). Alternatively, in MRI (which can be used for some functional imaging) a biologically targeted molecule (typically containing gadolinium, which is easily detected using MRI) can be administered. For CT, it could be argued that CT angiography, in which a contrast agent (that is opaque to X-rays) is injected into blood stream during imaging to spot delays in its passage through the coronary arteries indicative of narrowing, is a form of functional imaging, in which the contrast agent serves as a tracer.

With PET, and its close cousin SPECT (Single Photon Emission Computerised Tomography), the tracer is a molecule manufactured such that it includes a radioisotope, which enables it to be tracked. This has two important consequences. First is the flexibility in the design of the tracer – it is easier to incorporate a single atom than, for example, the chain necessary for fluorescence, without interfering with the behaviour of

the tracer in vivo. Second, such tracers are easier to detect, meaning less is needed, which further reduces the chance that the tracer influences the physiology being observed. Further, it means that concerns regarding the toxicity of the tracer are less pressing (MRI contrast agents with gadolinium, for example, need much higher does).

In PET the radio-isotope used must decay by positron emission. A selection of suitable isotopes is listed in Table 1, although the half lives of some of these isotopes can limit their utility: consider the time needed after production of the isotope to refine it, synthesise the tracer, refine the tracer, check the quality of the tracer (to ensure it is safe), transport it to a hospital and administer it to the patient – for most of the tracers listed the only option is to generate the isotopes on site, and with the exception of ^{68}Ga and ^{82}Rb that requires a cyclotron. Short half lives have the advantage, however, of quick decay and so less overall radiation dose for the patients. Other factors to consider are the ease with which tracers can be synthesised with these isotopes and how 'clean' their positron emissions are (some are able to decay by emitting other particles as well).

Positrons are the antimatter form of electrons. During decay one of the protons in the nucleus transforms into a neutron, this means to conserve charge a new particle must be released to carry the positive charge (protons having positive charge and neutrons no charge). The positron also produced carries the extra positive charge. Incidentally, a second particle is released, a neutrino, but this can be ignored in the context of PET imaging.

2.1 From Positrons to Photons

Being an anti-particle to electrons, when the positron encounters an electron the two will annihilate, converting their combined mass into energy. However, the positron will travel some short distance from the site of the decay until it has lost enough kinetic energy to be able to annihilate – it loses energy by interactions with other particles until

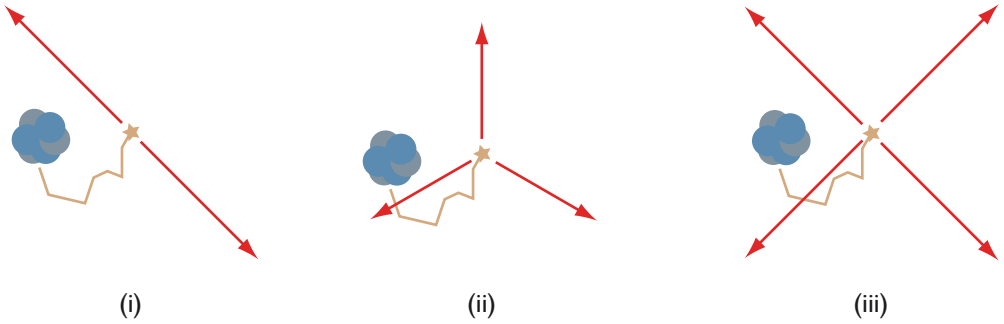

Figure 1 Three sample photon patterns which satisfy the zero sum momentum requirement. In practice that with two photons (i) is most common and the case with three (ii), four (iii), or more are increasingly rare.

it is moving slowly enough that is will remain close to an electron for sufficient time for the annihilation to occur. This ***positron range*** is the fundamental limit of PET resolution, as the scanner detects the photons that result from the annihilation, not the tracer decay itself. Table 1 includes the positron range in water (it does depend on the material that the positrons are bouncing around in).

Conveniently, the fact that the positrons cannot annihilate until they are almost at rest relative to an electron means that there is little or no momentum of the positron/electron pair before annihilation. Therefore, after annihilation, there cannot be much momentum either – conservation of momentum guarantees that. This means that the energy released by the annihilation must be divided into several portions travelling in different directions such that their combined momentum is zero. Hence each annihilation will produce at least two photons (and, in that case, they must travel in opposite directions). More photons are possible – see Figure 1 – although sufficiently rare that the basic theory of PET ignores them, leaving them to contribute to noise and other such phenomena of 'in practice'.

The simultaneous production of two photons that travel in almost directly opposite directions (in practice there is some residual momentum before annihilation, so the paths will not be exactly 180 degrees apart) is key to the sensitivity of PET, and therefore the quality of the images it can produce from small amounts of tracer. To understand

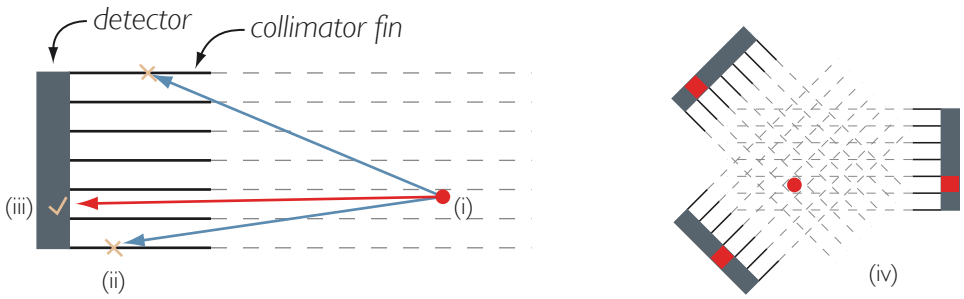

Figure 2 A basic form of collimation using straight fins. In SPECT the source (i) will emit photons in all directions. Many of these will be absorbed by the collimator (ii), reducing sensitivity. This is necessary, however, so that when photons are detected (iii) the position of the source can be constrained to a narrow strip perpendicular to the detector. Considering multiple detector orientations (iv) information about the position of the source can therefore be recovered. (In conventional SPECT it is common for one, two, or three detectors to rotate around the subject in order to capture the required numbers of views.)

this, consider SPECT. Although very closely related to PET, in SPECT the tracer is only required to produce one photon – therefore the range of radio-isotopes is much wider, as any that decay by γ emission are usable. The photon may travel in any direction from the point of decay, and as the photon is only detected at a point in the scanner there is no way to know where in the subject it came from. This compels the use of ***collimation***: a screening of photons such that any that are detected can be assumed to have be travelling in a specific direction. The simplest collimators are arrays of lead fins in front of the detectors, such that any photons not travelling perpendicular to the surface of the detector are absorbed. See Figure 2.

In PET, because two photons are detected, the decay is automatically localised to a ***line of response*** joining the two points in the scanner where the photons are detected. Therefore no decays are ignored as a result of their photons being absorbed by the collimator – in PET a much higher fraction of decays contribute to the data recorded by the scanner, so a better image can be formed (or formed faster, or less tracer used, or some combination thereof). See Figure 3.

This also explains why 3D PET is more sensitive than 2D PET. The latter restricts the scanner to looking at cross-sections through the subject perpendicular to the axis of the scanner. This is done by installing ***septa***, which are a form of collimator: lead fins that

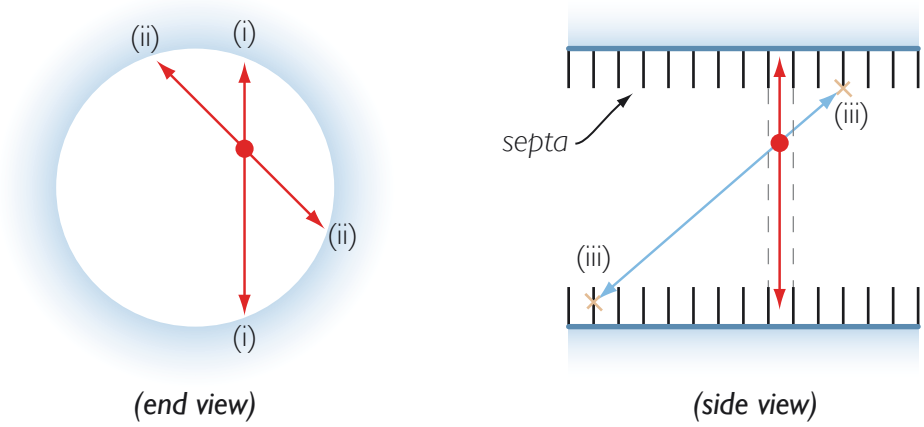

Figure 3 In PET, that coincidence implies co-linearity means that emissions in many more directions are useful: both pair (i) and pair (ii) will contribute to localizing the source, in contrast to the SPECT situation. To restrict a scanner to 2D mode septa are required. This collimation excludes photon pairs such as (iii) that do not correspond to a cross section. Full 3D PET acquisition does not use septa, and so would capture (iii), further aiding sensitivity.

absorb photons travelling at angles that are not perpendicular to the axis of the scanner. Removing the septa allows a greater fraction of decay events to be used. 2D PET is, however, much simpler to understand and implement – not to mention much easier to draw diagrams for!

2.2 The Journey to the Scanner

Between the point of annihilation and detection in the scanner the photons face an arduous journey. Some of the photons will bounce off other matter and thus have their direction changed – meaning that the assumption that the decay must have occurred on the path joining the two points where the photons are detected is invalid. This phenomenon is called ***scatter***. Other photons are absorbed or bounce so much that they lose enough energy not to be able to escape the body or be detected by the scanner. Thus some fraction of the decays will not be registered – this is termed ***attenuation***. The chance that a photon will be scattered or attenuated depends on the material through which it must pass. Therefore, both vary with position within the subject and any attempt to correct scatter and attenuation must be specific to the subject being scanned.

Before discussing attenuation and scatter correction, it is worth underlining that for a decay event to be detected both photons must be observed: any effect on either photon matters. To a large degree it is irrelevant where along the line the decay occurred: if one photon's path is shorter (and thus attenuation less likely, say) the other photon's path is longer (and thus attenuation more likely) by an equal amount.

2.2.1 Attenuation Correction

Correcting for attenuation is conceptually fairly simple. All it requires is an estimate of the fraction of photon-pairs that are 'lost' along each possible path through the subject. (There recording of photons detected by the scanner is quantised, so it is as if there are a finite number of points where photons can be detected and therefore a finite set of paths along which decays are assumed to have occurred – these paths are termed ***lines of response***, although they are actually volumes of course.) Given that estimate, the effect on the image of the counts observed along each line of response can be up-weighted accordingly, to make up for lost counts. Very early on the attenuation was estimated with crude anatomical models [13].

The early popular approach to attenuation correction was to perform a ***transmission scan*** immediately before or after the main PET, or emission, scan. Transmission scans involved another radioactive source formed into a rod that is inserted into the scanner and rotated around the periphery of the scanner's field of view. This arrangement means photons from the rod pass all the way through the subject, so the fraction of counts that make it through the subject from that position to each of the detectors on the other side of the subject, and thus attenuation, can be measured.

Nowadays most PET systems are ***hybrid PET-CT*** machines – a PET scanner and a CT scanner side by side in the same box, such that the bed slides through one into the other. These date back to 2000 – the first being described in [5]. This means that a CT scan can

be acquired without the subject moving from the bed, so that it can be much more easily and reliably aligned with the PET scan than if the two machines were separate. CTs use X-rays and, although these have a different energy from the 511 KeV photons that result from the positron / electron annihilation, it is possible to make a good estimate of the attenuation along every path through the subject using the CT scan of the subject [43].

Hybrid PET-MRI machines are starting to become available – an early example (1997) is shown in [75], and the first commercially available system was the Siemens Biograph mMR in 2010. Attenuation correction remains a key research area with PET-MR machines, because MRI operates on a fundamentally different principle than CT, and does not pass photons through the subject to collect data for the image. Further, that MRI struggles to identify bone make the matter more challenging. A key point that will be discussed later is that with PET-CT the scanners' fields of view do not overlap – it is not possible to do the attenuation correction scan at the same time as the emission scan. In PET-MR, the fields of view do overlap. *Some proposed systems consist of two completely separate scanners, sharing a bed. This does helps reduce misalignment of the two images, but the fields of view do not overlap, so in light of the above point it cannot be considered a true PET-MR.*

2.2.2 Scatter Correction

The main form of scatter correction in use today involves simulating the distribution of photons that result from single scatters – i.e. ignoring the possibility of a scattered photon being scattered a second time (which significantly simplifies the calculation, while not drastically reducing the results' accuracy). This involves an estimate of the tracer distribution and the density of the materials through the field of view. See [91] & [92]. That scatter correction alters the estimated tracer distribution means that a certain amount of iteration is required. The latter requirement is easily satisfied by the density

map needed for attenuation correction. Note, again for later, that the same comments about the separate fields of view are also relevant here.

2.3 Scintillation

Those photons that make it to the scanner reach the crystal blocks that form the first stage of the scanner's detectors. When high energy photons enters the crystal they undergo Compton scattering, and as they do so the energy released causes a cascade of more, lower-energy photons. This cascade results in a flash of visible light for each high energy photon that enters the crystal.

There are several properties of crystals that are desirable; the ***scintillation crystals*** are the single most expensive component in a PET scanner and patents (or licenses thereof) for crystal materials are an area of competition between hardware manufacturers and a crucial source of variations in performance, especially sensitivity.

After scintillation, it takes some time for the crystal to 'recharge' and be able to produce another cascade of photons. A quick recovery is desirable because any photons that arrive too early after an earlier one will not be detected, and this effect is difficult to model and correct for.

Another desirable property is the ***energy resolution*** of the crystal: that is to say the accuracy with which the energy of the original photon can be estimated from the flash of light that results – the two being proportional. There is a similar property called ***time resolution***: a sharper the peak in the light flash is allows the time that the high energy photon arrived to be determined more accurately.

It is also desirable to stop as many of the high energy photons as possible, otherwise they are not detected and that reduces the sensitivity of the scanner – this is referred to as the

crystal's ***stopping power***. There is also a related topic known as ***depth of interaction***: the photon cascade does not start at the surface of the crystal, and a high energy photon can penetrate some distance into the crystal before the cascade begins – possibly travelling from one crystal to the next. The resolution of the scanner can be improved by knowing the point where the interruption occurs more accurately, because this ties down the line along which the annihilation occurs more specifically. This is an ongoing area of research in the field.

2.4 Electronics and Data Acquisition

The light flash that results from scintillation has to be converted into an electronic signal so that the scanner can process it and record the event. The most common way of doing this at the moment uses ***Photo-Multiplier Tubes*** (PMTs). Partly driven by the development of PET-MR there is a push to develop ***Avalanche Photo Diodes*** (APDs) and ***Silicon Photo Multipliers*** (SiPMs) because they are smaller and can be used in the presence of the large magnetic fields required by MRI. Similar to the crystals, properties like recycle time, energy resolution, and time resolution are important, along with electronic noise considerations.

The next major development (at present no commercially viable systems exist) is likely to be ***Solid State Detectors*** (e.g. CZT), which detect the high energy photons, directly producing an electronic signal, and thus obviating the need for separate scintillation crystals.

The electronic signal has to processed, interpreted, and stored: ***data acquisition*** – not a trivial task given the number of possible channels and the rate at which events are detected. The performance of the data acquisition system can, therefore, limit the scanner's performance in very high count rate situations. Just like the crystals can be over-

loaded and miss events by not having time to recycle, the data acquisition system can be overloaded.

2.5 Coincidence and Time of Flight

The two high energy photons are created at the same time and travel at the same speed. Therefore they should be detected at approximately the same time. Only 'approximately' for two reasons: first are the limitations of timing accuracy already mentioned, second is that when the annihilation occurs off centre the paths to the crystals are of different lengths.

This time difference could be used to pin down the location of the annihilation along the line of response. However, this technology, termed ***time of flight***, has only become available comparatively recently – the benefits of an early commercial system is discussed in [41]. That the photons are travelling at the speed of light means that the difference in arrival time is minute: it takes a photon just over a nanosecond to cover a distance of one foot, so a timing resolution of ±500 picoseconds only locates the decay within about a 7.5cm. The time resolution of the whole detection chain must be high enough if time-of-flight information is to be useful; only recently has this been possible, and the time difference still only coarsely locates the decay along the line of response. A discussion of this is in [53], which draws attention to the difficulty for the whole system to offer adequate timing resolution in a practical setting.

Leaving time-of-flight aside, the data acquisition systems still need to identify coincident signals from the detectors and pair them up. There is a window of acceptable time differences, and likewise of acceptable energy levels (because scattered counts and other sources of photons can be screened out to some degree by excluding photons without the right energy level). These ***coincidences*** form the core PET data, recording the pairs of locations at which photons were detected.

2.5.1 Random Coincidences

Random coincidences are between photons that did not originate from the positron emission. Consider two single photons emitted by one of the alternative decay paths that some isotopes have: if detected at the same time they are recorded as a coincidence, although there was no decay along the line or response. Similarly, two actual decays at the same time can get confused with one another, their photons wrongly paired up.

Randoms are a problem because they will be confused with real data and thus influence the resulting image. The solution, or rather correction, applied, is to subtract ***delayed coincidences***.

If 'randoms' are truly random then the distribution of random coincidences should be the same if the definition of 'coincidence' is modified to include an offset: it is equally likely that two photons will arrive almost exactly X picoseconds apart, be $X=4$ or $X=0$ or anything else. True coincidences, on the other hand can only occur if there is no offset.

The delayed coincidences are used to correct for randoms – they are treated as negative counts. Note that this does not actually remove the specific random coincidences, rather it corrects for the pattern of randoms, thereby restoring some of the lost quantitative information. This has two effects worth mentioning: it is possible to have negative counts along individual lines of response, where more delayed coincidences occur than prompt coincidences; also, the subtraction radically alters the statistical properties of the data (as will be important later, and is therefore fully discussed in §2.1.4 of Chapter IV). The number of coincidences, both prompt and delayed, along each line of response follows a Poisson distribution, and the difference of two Poisson random variables is not Poisson – obvious given the possibility of negative counts.

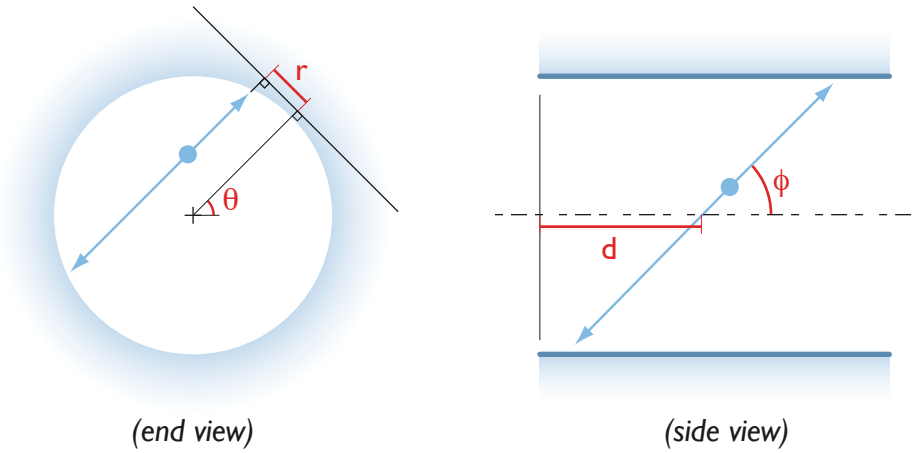

Figure 4 Schematic of sinogram coordinates. For 2D the projection angle θ, and radial offset *r* are enough to describe the line of response joining the two detected photons. Moving to 3D two additional coordinates are also needed: the oblique angle ϕ and the axial offset *d*.

2.6 Data Formats

There are two ways of collecting the counts generated by a PET scan. First is the ***sinogram***, a form of histogram, second is ***list mode format*** (LMF) data.

2.6.1 Sinogram

A sinogram is a histogram of the counts recorded during a scan, or some section thereof. It is divided into many bins, each of which contain the total number of counts observed along a given line of response. In practice this will bring together one or more crystal pairs.

The lines of response are described by their direction and an offset from the centre of the scanner. In 2D, for a slice through the subject, this means that each line of response has a ***projection angle*** θ and a ***radial offset*** *r*. Therefore 2D sinograms are a series (one for each slice through the subject) of 2D histograms, with bins indexed by θ and *r* as in Figure 4.

In 3D it is more complicated, as the oblique angles and offsets must be captured. To fully describe a line of response its angle relative to the bore axis and position along the bore must also be recorded. This means that the full 3D sinogram is a 4D histogram.

Sinograms have two limitations. First, they do not capture any information about time –which coincidences were detected when. If this is needed then the scan can be divided up into sections, or 'frames', in advance and a sinogram for each prepared, somewhat like a cine film. In practice, sinograms tend to be very sparse – many lines of response will have no counts in. This leads to the second problem: they are not very memory efficient, especially as there is not much structure to the sparsity. The problem is particularly acute for 3D, because of the increased number of possible lines of response, and if the scan is divided up into frames, as the counts in each sinogram will be reduced.

2.6.2 List Mode

In list mode data each count is recorded individually, providing a record of the line of response and the time of the event. In practice, sometimes other information is recorded, like the energy of the photons, the timing gap between them, etc. List mode format therefore addresses both of the concerns with sinogram data: time is captured and, as only counts are recorded, there is no need to allocate memory for the zero-count lines of response. It is easy to convert from list mode to one or more sinograms by histogramming the counts, just as one would have done in real time if only recording a sinogram.

Sinograms were used more often originally, largely because the early scanners were 2D and the reconstruction (the conversion of the data to an image) was easier to understand and implement starting from a sinogram. With the rise of 3D scanners recording list mode data became more popular because of the memory advantages, and its use facilitated by the growth of computational power. Eventually (following from the algorithm proposed in with [63]) it became possible to perform reconstructions without needing

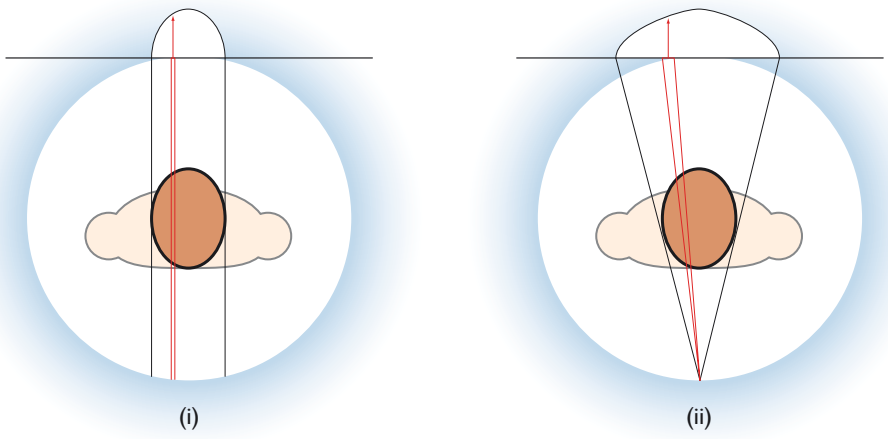

Figure 5 Comparison of parallel beam projection (i) as found in PET with fan beam projection (ii) as in modern CT. In both cases the values at a point on the projections show an integration of tracer activity (for PET) or x-ray attenuation (for CT) through a section of the subject. Note the symmetry in the case of (i): the opposite projection is identical.

to histogram the data. Likewise direct estimations from list mode are now possible for dynamic PET (introduced shortly), as originally proposed in [10] or [80], or, a more recent approach, [74].

A final note: list mode data is in the projective domain, like sinogram data, but it is not in the sinogram domain. This is because real list mode formats record not the angles and offsets of lines of response, but rather the pair of crystals that detected the photons. Conversion from those pairs of crystal 'address' to the sinogram domain proper involves consideration of which discrete bin on the sinogram the line joining the two crystals falls.

2.7 The Radon Transform

The counts are captured in a projective space and so PET data can be described mathematically (albeit crudely) by a ***Radon Transform***.

More specifically, the data is formed by parallel beam projection of the tracer distribution. That is to say focal point of the projection is at infinity, and is opposed to fan beam

projection, with a specific focal point (like CT, where the CT source emits rays in many, diverging directions) – the two are compared in Figure 5.

The Radon Transform converts the original function (the tracer distribution) to a new one in the projective space, whose values give the integral of the original function along a path corresponding to a line with at the relevant angle and position for the coordinates of the point in the projective space. The sinogram is, therefore, a discretised version of the new function.

Although it can be generalised to 3D, in (1) – and for most of this discussion – the 2D transformation is used, for simplicity. The condition on x and y means that the integration is constrained to move along a line at angle θ, offset by r, as Figure 4.

$$S(\theta, r) = \iint I(x, y)\,\mathrm{d}x\mathrm{d}y \quad \text{where: } \tan(\theta) = \frac{y + r\cos(\theta)}{x - r\sin(\theta)} \tag{1}$$

The term 'sinogram' comes from an attempt to capture that it is a histogram that characteristically exhibits a sinusoidal pattern. The latter comes from the impulse response of the radon transform. If the original function is a delta function (i.e. if a unit of tracer is concentrated as a point source) then the Radon transform will produce a function which is zero everywhere except for a set of points whose r value for a given θ satisfies a sinusoidal expression (2), in which x_0 and y_0 are the position of the delta function (and, in fact, A and ϕ are the polar coordinates of the delta function's position).

$$r = A\,\sin(\phi - \theta) \quad \text{where: } A = \sqrt{x_0^2 + y_0^2} \quad \text{and } \phi = \arctan\left(\frac{y_0}{x_0}\right) \tag{2}$$

As the Radon transform is linear it obeys the principle of superposition, and so the sinogram can be considered a superposition of the sinusoids corresponding to every point in the subject.

This is the root of the strong statistical properties of sinogram domain data: the emissions from each point in the subject follow a Poisson distribution (being a radioactive decay process) and thus the value expected along the sinusoid is also a Poisson distributed random variable. (Of course the emissions from that point are spread out over all possible directions, but that just scales the distribution.) Each bin on the sinogram contains the counts from a superposition of many such sinusoids, and so, as a sum of Poisson distributions, is itself Poisson distributed.

As mentioned, the effects of attenuation, scatter, and randoms will be discussed later, but fundamentally the Poisson distribution is a very good model for the sinogram domain data.

Every point on the sinogram is related to many points in the subject (those along the line of response) and every point in the subject contributes to many points on the sinogram (those along the sinusoid). This makes inferring geometrical information about the subject from the sinogram somewhat tricky.

2.8 Reconstruction

Reconstruction refers to the process of converting the projection domain data (be it a sinogram or list mode data) back to a Cartesian, or X, Y, Z space, to form an image of the subject. *What is discussed here is limited to PET, in other forms of tomography (i.e. imaging using multiple projections) the relative merits and weaknesses of the approaches are different.*

2.8.1 Filtered Back Projection

Filtered Back Projection (FBP) is the most basic approach to reconstruction used in practice. Back projection is, in very simple terms, smearing the counts back along the lines of response. In practice this is done by first converting the projections (rows of

the sinogram, i.e. bins with the same θ value) into the Fourier domain, then, using the Fourier Central Slice Theorem, assembling the Fourier spectrum of the image, before converting that into the image. This approach is much faster than other ways of performing back projection.

The key step is the ***Fourier Central Slice Theorem***, which states – for the 2D case – that the 1D Fourier spectrum of a projection at some angle is the same as a slice through the 2D Fourier spectrum of the original function, provided that the slice is at the same angle as the projection. (This will be discussed more fully in Chapter III, where the theorem is needed for the work presented in that chapter.)

The eponymous 'filter' in FBP refers to a high pass ramp filter applied to the projections while they are in the Fourier domain. The reason for this that the central regions of the scanners field of view are over-sampled relative to more peripheral regions. This can be understood in a couple of ways.

First, most obviously, is that as the slices are used to assemble the 2D Fourier spectrum they all pass through the origin of the frequency space (at low frequencies), whereas at higher frequencies (further from the origin) the slices are further apart from one another. Therefore the ramp filter down weights the low frequency components and up-weights the higher frequency ones.

The second explanation is as a correction for the smearing of counts all the way along the lines of response – each count started at a point, but it's effect is distributed along the whole line of response. A high pass filter acts to sharpen the image. As the back projection is done in the Fourier domain anyway the sharpening can be done by multiplying the Fourier transforms of the projections by the ramp, rather than the more computationally intensive act of convolving the resulting image with a sharpening kernel. *§4.3 of* [87] *presents a fuller discussion, but for this thesis the details are not important.*

One point to note is that the noise inherent in real world imaging is amplified by the ramp filter. This is because at high frequencies the noise is much more significant that the true signal. Plain FBP, therefore, produces very noisy images, so the ramp filter is combined with a low-pass filter of some form to limit the noise amplification. This is, however, the equivalent of blurring the resulting image to deal with noise. Phrased like that, the noise is clearly a fundamental problem for FBP.

Lastly, even today, FBP has two advantages. First is that it is very fast to perform compared to other forms of reconstruction. Second is that it is analytically much more tractable than other forms of reconstruction. The consequence of the latter is that it is possible to follow the statistical properties of the data through to the reconstructed image, thereby estimating the distribution of the values in the images. One of the great promises of PET is that it could be a quantitative form of imaging, producing numerical values that have meaning in terms of tracer concentration (as opposed to images that only convey useful information through the shape of the tracer distribution). For this to be fully exploited, however, the statistical distributions of the values would have to be estimated, as the values of point in the images are, in fact, random variables.

2.8.2 Iterative Reconstruction

Dissatisfaction with the noisy image of FBP and the strong statistical properties of PET data let to the development of iterative reconstruction, using the ***Maximum Likelihood / Expectation Maximisation*** (MLEM) algorithm [17], applied in 1982 to emission tomography in [76]. This was initially of little utility as the time taken to perform reconstructions was not feasible in clinical practice. However, in 1994 the ***Ordered Subsets Expectation Maximisation*** (OSEM) implementation [37] – helped a bit by the increase in computational power available – made iterative reconstruction practical, and, indeed, the main form of reconstruction used in clinical practice.

MLEM involves postulating a tracer distribution (i.e. image), projecting it, comparing it with the actual tracer distribution, then back projecting the differences to compute the optimal update of the estimated image. This is done iteratively until the image estimate converges.

OSEM speeds this process up by only using a selection of projections at a time – performing a 'sub-iteration', before moving on to the next subset. The idea is to reduce the number of computations per iteration by using fewer projections, yet gain overall performance because the better converged information from the previous subsets makes the current subset converge faster. Switching back and forth between the image and sinogram domains spreads information from each projection to the others because of the many-to-many relationships between points in each domain.

Iterative reconstruction produces much better images – the noise is better controlled (i.e. the trade-off between having a noisy image or a blurred one is less binary). It also avoids the 'streaking' typical of FBP, a result of smearing back a discrete set of projections (the more closely space the projection angles are the less apparent this is).

There are problems with iterative reconstruction. OSEM is not guaranteed to converge, in practice it often settles into a limit cycle, changing the image depending on which subset has just been used. Even with the benefit of OSEM, iterative reconstruction is still computationally very expensive. The performance of the algorithm depends on the number of subsets, iterations done on each subset, and (theoretically) the initialization (what the image starts as).

More seriously, the final image can depend on those configuration options of the method. The implications are discussed (for dynamic PET, discussed shortly) in [59]. It is common to halt iteration early, before convergence is complete. This is because the final phase of the convergence seems to be noise: the image converges faster than the

noise and so a arbitrary cutoff is used to improve image quality. This is a little better than just blurring the image to hold noise in check. Finally, it is very difficult to do any statistical analysis of the process; unlike FBP the values of the image are just that, values, not mean estimates of random variables with known distributions.

2.8.3 Recent Advances

The first ongoing area to discuss is ***Maximum a Postiori*** (MAP) reconstruction. This also uses the statistical properties of the data during an optimization type algorithm, but almost in the reverse direction. MLEM is a maximum likelihood approach, that is to say it tries to find the image that makes the observed data most likely given that image. MAP uses some assumed prior and Bayes rule to transform the problem into finding the image that is itself most likely given the data.

MAP methods have shown some promise of very good quality images, but it is difficult to maintain the computational performance once the prior is introduced – indeed the method was originally introduced in 1985 [30], but is still not commonly used. The choice of prior has a fundamental impact on the image produced (as well as performance), much more so than in MLEM. This is arguably the inherent flaw of MAP methods: the prior must be assumed.

Time of Flight (TOF) information is used to help reconstruction. Again this was proposed early in the development of PET – it is the obvious way of knowing where the decay occurred given two photons after all. It has only recently been incorporated into to commercial systems, because of the demands for fine timing resolution, indeed at least as far back as 1980 ([1]) research in to how to improve timing resolution can be found for this reason.

With TOF, each event can be restricted to having come from some section of the line of response. As the time difference cannot locate the decay exactly a 'kernel' is used, for instance a Gaussian, to represent the probability of the decay occurring at any given point along the line of response. This could be incorporated into an FBP type reconstruction by only smearing that count out over a part of the line of response, and doing so with the density determined by the kernel, or directly into the back and forward projection of iterative reconstruction.

TOF helps make convergence faster, and seems to do so by limiting the influence of noise: the effect of erroneous events is limited by the width of the kernel to a smaller section of the image, and, as with OSEM, having some better converged parts to the image helps speed up the convergence of the rest. The practical benefits of TOF are discussed in [41] and [42].

Point Spread Function reconstruction (PSF) uses a measured impulse response – i.e. the pattern of counts in the sinogram domain observed for a point source at any given position inside the scanner's field of view – during reconstruction, as an improved way of doing the projection calculations ([61] & [62]). This measurement is specific to each model of scanner. PSF reconstruction further improves the image quality and speeds recovery, and can be combined with TOF, as is also discussed in [41].

2.9 Functional Imaging, Reprise

At the start of this chapter, some stress was put on PET as a functional imaging modality, in contrast to more anatomically orientated techniques. With a functional modality the objective might be to reconstruct something other than an image, something that directly describes the to the behaviour of the tracer.

That is not to say that seeing where the tracer is not useful, indeed in some cases that might be the objective. Nor is it to say that the process of reconstruction is unnecessary sometimes, on the contrary, almost any analysis requires some form of reconstruction to relate spatial concepts, like a region of interest, to the data.

However, if the interest is to study the phamacokinetic behaviour of the tracer it is not necessarily useful to produce an 'image' of the sort that one might think of.

2.9.1 Dynamic PET

Dynamic PET refers to studies that make use of information about the variation of tracer over time. At it simplest this would entail reconstructing several images as frames, so that the evolution of the tracer distribution could be estimated. To this functions describing the time course could be fitted, or parameters of a compartment model could be fitted. Compartment modelling (see [36]) is a common form of pharmacokinetic modelling in which parts of the body – below the resolution of the scanner like blood, inter-cellular fluid, or the interior of cells – or states within biochemical processes are described as compartments, between which the tracer moves at unknown rates. The information of use is then not the geometrical distribution of the tracer, but rather the parameters describing its behaviour in moving between compartments in various tissues.

With this approach there is a trade-off between the temporal resolution afforded by using many short frames and the drop in image quality that results from having fewer counts in each, and therefore a lower signal to noise ratio. The noise in each frame introduces more errors in fitting the models. The situation can be somewhat ameliorated by fitting the models not to each point in the image but rather to a whole region, e.g. a homogenous volume of tissue, as the spatial averaging reduces noise.

As already mentioned in §2.6.2, today it is possible to fit the models, even to regions, directly: instead of reconstructing images and then fitting the parameters they are directly estimated, as the objective of the optimization inherent in iterative reconstruction. Consider MLEM, instead of postulating an image and then iterating back & forth between image and sinogram, it is the parameters of a function describing the time course of trace at each point that are postulated and then improved, as proposed in [10].

Although dynamic PET is less widely used than the alternative (static PET, discussed next), it is arguably the more natural and general form of PET, because it is inherently functional.

2.9.2 Static PET

The opposite of dynamic PET (or at least the extreme case of it) is ***static PET***, in which a short scan (typically 5-10 minutes, as opposed to anything up to 2 hours) is taken to produce a single, static, snapshot of the tracer distribution.

Static PET is useful with tracers whose accumulation is of interest. By far the most common is ***FDG*** – 2-deoxy-2-(^{18}F)fluoro-D-glucose – an analog of glucose where an -OH group is replaced by the positron emitting ^{18}F. FDG is treated like ordinary glucose by the body... until it is used for respiration in a cell. At that point the chemical differences mean that the body cannot do anything with it, so it accumulates in the cells. Its accumulation is, therefore, in proportion to the rate at which the cells are metabolising glucose.

Because of that irreversible (at least on the time scales of interest) accumulation – 'respired glucose' is a compartment from which there is no exit – a short snapshot scan some time after the tracer is administered gives a good estimate of the rate governing the entry into that compartment. Therefore the static image is, in fact, an easy proxy for the

dynamic information needed and hence static PET can be considered to be a convenient (i.e. much shorter scan time) special case of dynamic PET.

2.9.2.1 More on FDG

FDG deserves special mention as a tracer because it has lead the rapid increase in the use of PET. Its principal role is in oncology. Functional imaging is obviously particularly useful in that field, cancer being a disease of metabolism: the behaviour of the suspect tissue, not its anatomy, that is fundamentally important.

In cancerous tissue there are several changes that directly and indirectly increase the demand for energy and the proportion of that energy derived from glucose respiration. Therefore an unusual amount of FDG accumulates in cancerous cells, and tumours appear as 'hot-spots' in the PET image.

Also the functional focus of PET makes it well suited to detecting the response of a cancer to treatment. With PET the effect of the treatment on the vitality of the tumour can be directly observed; for example [45] shows the use of PET to investigate the effects of chemotherapy, and claims 'significant' differences can be observed within 24 hours. Given the unpleasant side effects and range of options for chemotherapy any rapid estimates of its efficacy are a real boon to patients.

Lastly, an interesting point about FDG: the half life of ^{18}F posed serious logistical problems. The solution was a combination of automated synthesis machines and the dispatch of shipments of tracer before the completion of the assaying and other quality control procedures: these are completed while the tracer is out for delivery, so that it is completed before use, but no time is unnecessarily wasted.

This concludes the brief introduction to PET. In addition to background and the introduction of the key processes in PET imaging relevant to the work presented in this thesis the ulterior motive has been to draw attention to two facts. First is that there has been much work on image quality, both in terms of resolution and sensitivity, through hardware design or improved reconstruction methods. Second is that PET's great promise is as a form of functional imaging is a result of its quantitative and dynamic imaging abilities.

For more detail on PET beyond what has been presented in this section the reader is referred to [65] or [6].

As the rest of this chapter discusses motions and their effects on PET imaging. In addition to focusing the attention of this thesis on one class of motions – namely twitches – the following will argue that motion and its effects are currently a major limitation of the modality. Further, the importance of these limitations will only increase as PET develops along its apparent trajectory, both in terms of the technology and its use in practice.

3. Motion and Its Effects

3.1 Categorising Motion

Motion is the process by which the subject, or part thereof, changes position. As such, a great variety of different characteristics can be used to describe a motion. Its speed, or how quickly the change is completed, is one such. Perhaps 'duration' is a better term, as this allows the size of the change to be considered separately. There is also the degree of locality to the motion: ranging from a small part to the whole of the subject moving.

Motions can also be described based on the nature of the change. In the context of medical imaging, some motion classes are more relevant than others. ***Rigid body*** means that the whole subject moves as a solid object, able to translate and rotate, but not change shape, even through scaling. Adding the ability to scale to rigid body motion makes it an ***affine*** motion, although that is uncommon in medical imaging (apart form a scaling between different image resolutions). The most common class is also the most general: ***deformable*** motion. In the context of medical imaging, there might be restrictions to completely free deformable motion: tissue cannot, for example, disappear – it can compress and expand, but not fold over itself or compress to nothing. *Of course, if a comparison is being made between two scans between which there has been a surgery then tissue may have disappeared. This is not a motion, but there is overlap: e.g. the task of aligning an image from one part of a scan, corresponding to one position, to another part of the scan, for the other scan, is very similar to that of aligning one scan to another.)*

Another feature of motions that will prove important is periodicity: does the motion move the subject through a (sufficiently) repeating cycle of positions?

For the purposes of discussion within this thesis, motions will be divided into three main categories. In reality there is no hard boundary between one class and the next. Here

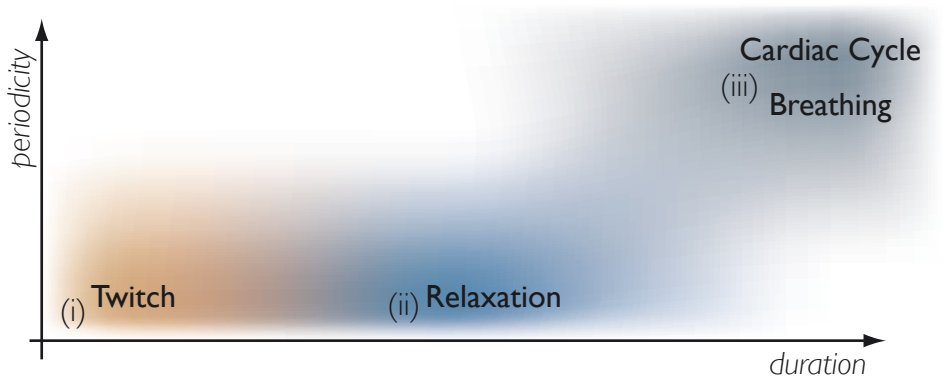

Figure 6 A classification of motions: (i) transient events called 'twitches', (ii) progressive change of pose called 'relaxations', and (iii) the ongoing 'periodic' motions.

three classes will be used, split principally using their periodicity and duration characteristics as Figure 6: '***twitches***', '***relaxations***', and the '***periodic***' ones.

3.1.1 TWITCHES

These are motions that may be thought of as transient events rather than ongoing processes, at least to a first approximation. Therefore they must have a short duration, and necessarily can not be periodic.

'Short' is a relative term, and in this case its meaning comes from the need to be able to treat the motions as transient events: the duration of the motion must be sufficiently less than that with the subject in a stable position, allowing the scan to be considered as data from a sequence of poses, punctuated by effectively instantaneous changes. This means that if the scan is divided into frames at twitches, adjacent frames can be compared against each other to estimate the motion, then suitably transformed to remove the effects, yielding a complete scan as if the subject had not moved.

Depending on the level of precision required it might be necessary to remove some data around the time of the twitch in order to exclude the time spent moving. However, as the twitch is assumed to be of short duration, loosing this data will not reduce the total amount of data available enough to undermine the utility of the scan.

Examples of twitches include muscle spasms caused by neurological conditions or as a result of an anaesthetic wearing off, a cough, sudden motion of a pocket of gas through the digestive tract, etc. Some of these will be localised motions, others will cover a larger region (say a limb), while some may even be a global motion – at least as far as the scan is concerned: consider a head motion during a brain scan. Incidentally, that last example is also one where, thanks to the skull, the motion is known to be a rigid body motion. Most motions will be deformable, although with limb motions more complex models are relevant, like poly-rigid body motion (considering the subject to be a collection of linked rigid-bodies).

3.1.2 Relaxations

Most patients are apprehensive at the start of their scan and their psychological tension manifests as muscle tension, especially in the shoulders and neck. As the scan proceeds they relax and 'settle' into a more natural position on the bed. A gradual relaxation of the muscles in the shoulders and neck causes the head to roll back slightly and the shoulders to spread out. Such motions would be expected to be non-rigid and global.

This motion is an ongoing process over the a significant duration of the scan, as such it is not feasible to separate the scan into 'non-moving' and 'moving' sections, at least if the intention is to discard the moving sections. The motion process (as opposed to just the resulting change in position) must be estimated and used to correct the data, applying a varying transformation to the data, in order to remove the effect.

3.1.3 Periodic Motions

This final class of motions includes such things as the heart beat, breathing, and perhaps theoretically even certain tremors (which would be expected with a variety of neurological conditions). They are ongoing process (as one would hope in the case of the

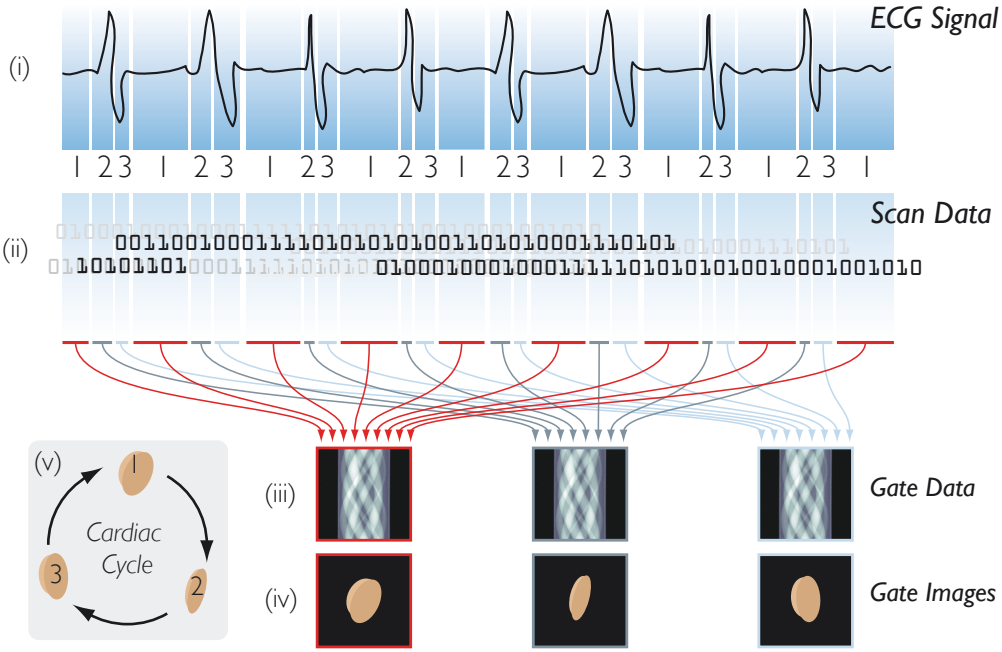

Figure 7 Schematic explanation of gating. The ECG signal is used to divide the scan up into gates corresponding to phases of the motion (i). These gates are then applied to the stream of data from the scanner (ii). For each gate the data is gathered together from across the whole scan (iii) and this collection of data, from many parts of the scan, but corresponding to the same phase of motion, is used to form images (iv). Note that this is only possible because the cardiac cycle (v) – highly simplified here – is responsible for both the ECG signal and the motion, so that the former can be used as a proxy for the heart's position.

heart beat and breathing). However, they follow a periodic structure, where the subject moves through a range of configurations in a loop, so over the duration of the whole scan there will be many short sections of data corresponding to the same configuration. The changes between the configurations will involve a non-rigid and generally localised transformation.

These are probably the most studied of these classes and a common solution exists, namely ***gating***.

3.1.3.1 Gating

This is the act of splitting the data up between several '***gates***', each representing a different phase of the periodic variation of subject configuration – see Figure 7.

During the scan, some easily externally observable signal is used to infer which gate the data currently being recorded falls into. In the case of the heart, the ECG trace is used, as the configuration of the heart is strongly linked to the ECG. For breathing the typical practice is to use a pneumatic belt stretched across the chest to measure the expansion of the lungs. ***Dual-gating*** refers to using two orthogonal sets of gating simultaneously, which means breathing and heart beat in practice; data is sorted into sections so that both the lung and heart configuration match.

The gates can then be treated individually. One use would be to reconstruct them to form images that, when shown in sequence, animate the cyclical motion; this would be very useful in knowing how a tumour in the lungs moves, say for planning radiotherapy. Alternatively, the motions between each phase could be estimated and used to map all the data back to form a single, better, image corresponding to one configuration. For example [55] investigates the over estimation of lung tumour size and underestimation of tracer concentration that results from not gating for breathing. Another option, in a dynamic PET scenario, would be to look at the time course of a tracer over the data in each gate, allowing the pharmacokinetics to be estimated using one gate. Less data would be used than in the previous option, but any errors in the motion estimate or correction would be avoided.

Gating works well for motions that are periodic and easily observable. The cardiac cycle works best; with breathing the motion is less well modelled as periodic process – breaths can be deep or shallow, one breath may not be completed before the next begun, and coughs move both the external signal and lungs in a way that does not match the model at all.

The more gates are used the less residual motion there will be in each gate. However, there will also be fewer counts in each gate and so any images (or phamacokinetic model fitting) will be of lower quality. Dual gating exacerbates this problem because the total

number of gates is the product of the number of gates used for each type of motion. There is also, eventually, confusion as to which gate data belongs it; if there is not enough variation on the external signal to distinguish between two phases then there is no way to allocate the data to gates.

3.2 Effects of Motion

3.2.1 Blurring

Much like a long exposure photograph, if a subject moves during a scan the data from the different positions gets mixed together. This means that there will be some motion blur in the images. This blurring will obliterate fine details and softens the boundaries of regions. In doing so it will reduce the contrast of a 'hot' or 'cold' region (i.e. one with a comparatively or low high tracer concentration) against the background. Hot regions spread out and cold regions are shrunk by the background counts mixing in. This effect, from lung motion is examined in [20], which reports a variation of up to 30% in tracer uptake and a 21% variation in size between the opposite ends of the breathing cycle.

Given that measuring the mean activity in a region relative to the background activity is a common measurement used in interpreting PET images, as it measuring the size of hot or cold spots, this causes problems. Consider comparing one scan against an earlier one, to see how effective chemotherapy is: if the subject moves more in the later scan then the tumour, probably a hot spot, will appear larger, but less intense. What does this mean in terms of, say, whether the treatment regime should be changed?

In the case of a dynamic study the blurring has another effect: attempting to model the variation in tracer concentration in a region it is impossible to tell whether the observed changes are due to tracer being moved in or out of the region as a consequence of physi-

ological activity or because of the bulk motion of the subject, such that the actual region in the body no longer corresponds to the region defined for analysis, as discussed in [51].

3.2.2 Non-Linearity of Iterative Reconstruction

The plot thickens with iterative reconstruction (including the direct or indirect estimation of kinetics) because the methods in question are non-linear. That is to say, with R() denoting reconstruction, and *A*, *B* being data from different positions:

$$\mathrm{R}(\alpha A + \beta B) \neq \alpha \mathrm{R}(A) + \beta \mathrm{R}(B) \tag{3}$$

Therefore, a twitch that moves the subject from one position to another half way through the scan will not produce an image (or model fitting) that straightforwardly corresponds to a combination of the results that would be obtained from the two positions analysed separately. (Recall that Filtered Back-Projection is linear.)

This concern extends beyond the fact that motion will introduce errors: potentially those introduced will not be obvious or expected, nonlinear effects typically being un-intuitive. For instance: because of the iteration between the sinogram and image domains a local motion's effects may affect the convergence of the whole image.

Furthermore, it seems that this concern has not been discussed at length in the literature. Perhaps this is because there is a schism between the 'imaging' side of the field – physics, hardware, low level software including reconstruction – and the 'applications' side – focused more on the medicine or machine vision applications. For those versed in analysing images, there is an understandable inclination to start with the image produced by the scanner – after reconstruction. For those working at the lower level, likewise it is fair to leave the implications to people who understand medicine or molecular biology.

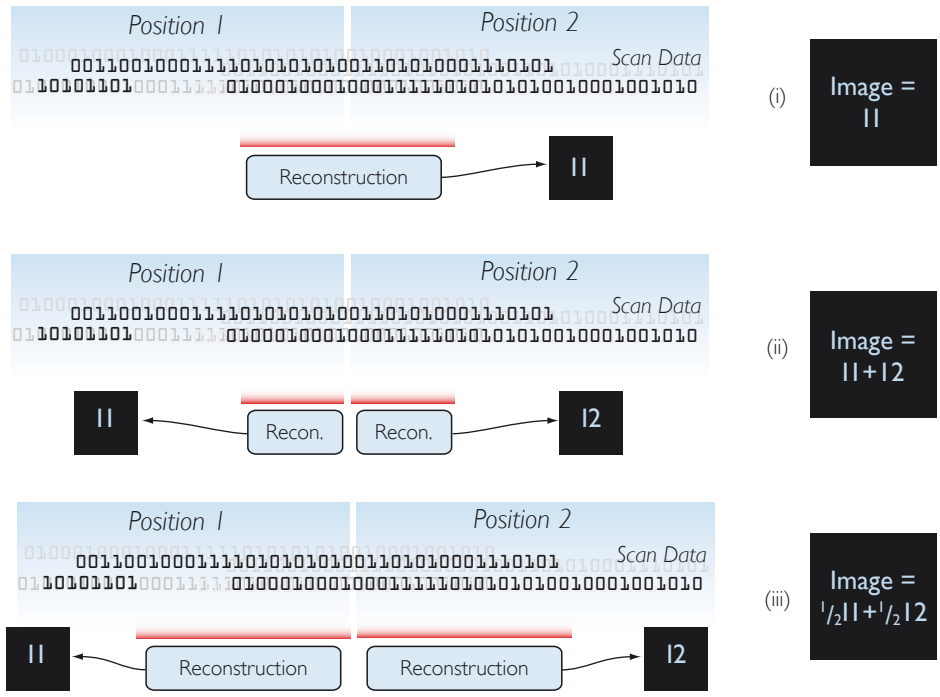

Figure 8 Schematic comparison of the different image formation approaches being compared. In (i), the usual case, a single image is directly reconstructed with the mixed data. In (ii) data from each position is reconstructed separately and the two resulting images are combined to form an overall image. In (iii) two images are again reconstructed separately and then combined, but in this case extra data from each position is used so that each reconstruction contains as much data as the one in (i) does. The values in the separate images must be halved when combining them in case (iii).

This is the reason for the desire to focus the work in this thesis has on performing motion studies – something one would normally consider an application side task – before reconstruction of images. There are also good reasons to do so, which will be discussed in due course.

3.2.2.1 Demonstration

Consider again the example in which a subject that changes position half way through the duration of the scan. As a first investigation of the implications of the non-linearity of iterative construction the result of reconstructing the data from both positions together is compared to that from adding together two images created by reconstructing each half of the scan separately. To control for any effect that the total number of counts might have on the reconstruction, a second comparison is performed, this time against an image produced by adding half of each image from the two separate positions, but

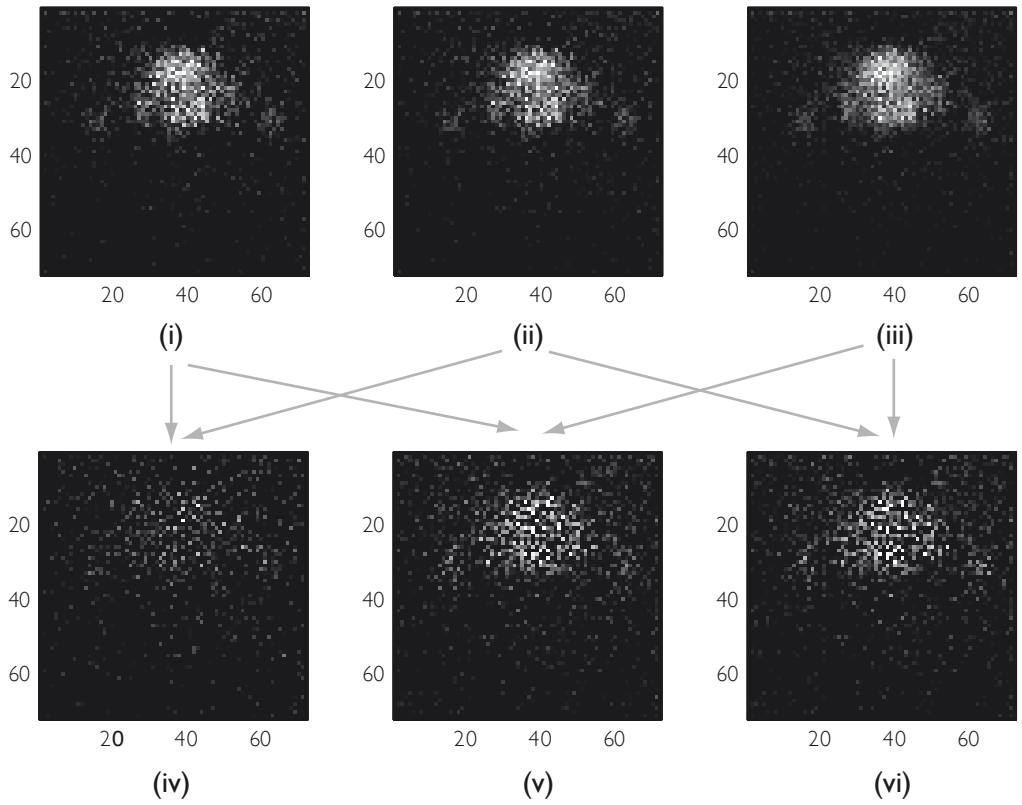

Figure 9 Demonstration of the effect of the non-lniarity of iterative reconstruction. The top row shows the result of OSEM reconstruction of the different schemes – the labels (i) through (iii) matching the schemes described in Figure 8 and equation (4). The bottom row shows the differences: (iv) = (ii) - (i), (v) = (iii) - (i), and (vi) = (iii)- (ii).

such that the amount of data used for each position is the same as for the single reconstruction of the mixed data – Figure 8 outlines the schemes.

In terms of an equation, the comparison is between:

$$\begin{aligned} \text{scheme } 1: &\quad \mathrm{R}(A+B) \\ 2: &\quad \mathrm{R}(A)+\mathrm{R}(B) \\ 3: &\quad \frac{1}{2}\mathrm{R}(2A)+\frac{1}{2}\mathrm{R}(2B) \end{aligned} \tag{4}$$

Note that if there is a significant effect from the amount of data – i.e. between scheme 2 and 3 – then an uneven mix of the two positions will pose more significant problems than a motion half way though the scan.

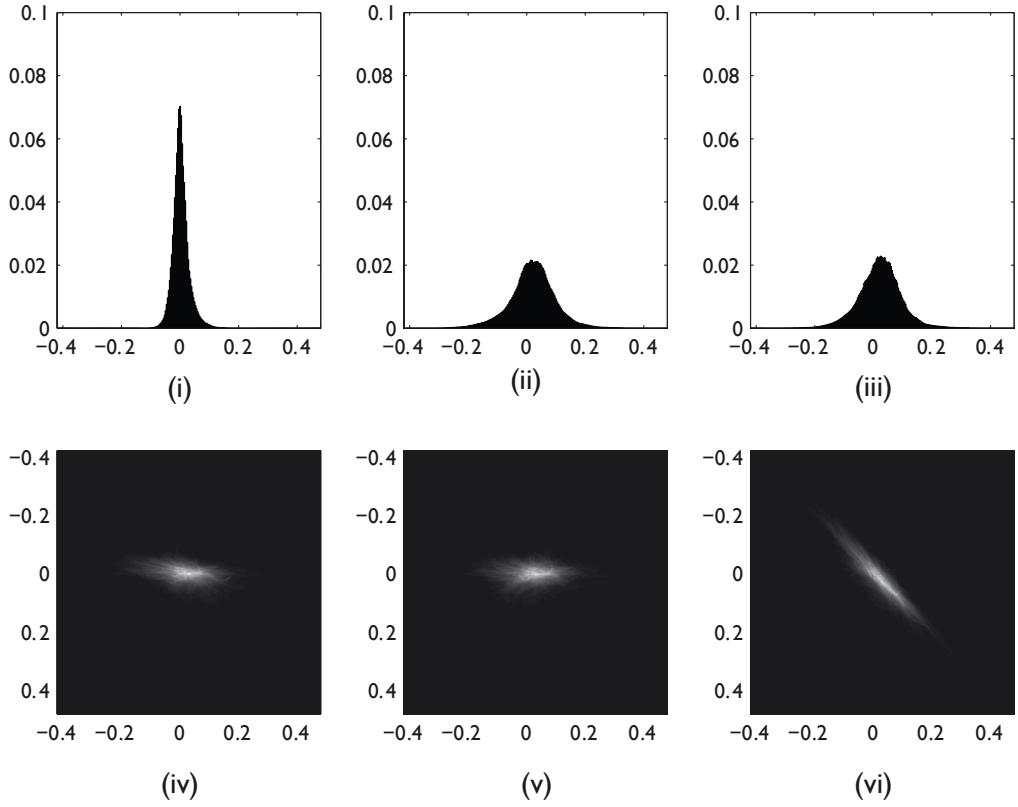

Figure 10 Estimated probability density functions (top row) and joint distributions (bottom row). (i) to (iii) correspond to the value within the rats from (iv) to (vi) in Figure 9. The joint probability density function in (iv) corresponds to (iv) and (v) in Figure 9. Likewise (v) here corresponds to the earlier (iv) and (vi), with (vi) here corresponding to (v) and (vi).

To demonstrate this experiment the above three scenarios have been performed on a set of real data generated for experiments later in this thesis (the 'Rat 1' study, see page 172). A short period of time around the motion was excluded so that the act of moving the animal did not introduce any unusual data, approximately one or two minutes of data was used (depending on the scheme). The animal was moved 9mm along the axis of the scanner. The reconstruction was done using the OSEM algorithm as implemented for Matlab in [23]. The three reconstructions and the differences between them are shown in Figure 9.

Qualitativly, scheme (iii) produces the best (smoothest) image, and the other two produce fairly similar images, with a sharper noise pattern than (iii). This is supported by Figure 10, which shows the estimated probability density functions for the differences – assuming that the noise (which is a result of each individual pixels being random

variables) is spatially uniform and therefore can be approximated by the distribution of values within a region, in this case a mask set to match the outline of the rat. The change between (i) to (ii) & (iii) in Figure 10 implies fewer large differences between reconstruction scheme 1 and 2 than between the images produced by scheme 2 versus 3 or 1 versus 3 – i.e. when scheme 3 is not involved in the comparison the differences are smaller.

The PDFs are estimated using the 'NP-Windows' method [40] which avoids sensitivity to the choice of bin size without having to assume a model of the PDFs. The first PDF is visible different compared to the other two. Also, the joint PDF for the latter two – i.e. that shown in (vi) of Figure 10 – shows a pronounced diagonal pattern, implying that the values of the differences in the values of the differences are correlated, as well as having similar distributions.

These difference PDFs can be used to make a numerical comparisons between the patterns in the images. Using normalised redundancy:

$$R_* = \frac{I_{12}}{H_1 + H_2} \Big/ R_{max} \quad \text{where: } R_{max} = \frac{\min(H_1, H_2)}{H_1 + H_2} \qquad (5)$$

in which I_{12} is the mutual information for two PDFs and H_1, H_2 the entropy of each PDF. This value will vary between 0 if the differences between the images described by the PDFs are independent and 1 when they are identical. The three sets of pdfs in Figure 10 have normalised redundancies of: 0.058394 for (i) vs. (ii), 0.025814 for (i) vs. (iii) and 0.348216 for (ii) vs. (iii). These numbers confirm the qualitative conclusion that the first PDF is different from the other two – and it is when scheme 3 is involved in both of the PDFs being compared the normalised redundancy is an order of magnitude greater.

This result is perhaps surprising: given that in reconstruction scheme 2 each half of the final image is reconstructed from half as much data as any of the other reconstruction it would be reasonable to expect this image to be noticeably worse than the other two. Instead it appears that it is the same as in scheme 1, in which the reconstruction

algorithm has twice as much data in one go, but that data coming from a mix of the two positions. That scheme 3 does produce a better image suggests that the benefit to reconstruction of the extra data is more significant if the extra data is self consistent – which would make sense given the nature of the MLEM algorithm.

> *This is an obvious area for much more work in future. To understand the full implications of the nonlinearity similar experiments would need to be conducted for a wide range of scenarios: different subjects, motions, scanners, dosing, reconstruction etc. Even an initial study would be the work of a thesis itself! As such the above demonstration is merely an illustration of the type of experiment and a signal that further work needs to be done by the field to the effects of the interaction between motion and non-linear reconstruction. To this end some software has be written to allow the raw list mode data to be manipulated, creating data corresponding to the scenarios above that could be fed into commercially available reconstruction systems. That software is discussed in Appendix A.*

3.2.3 Attenuation and Scatter Correction

Both attenuation correction and scatter correction rely on the density map estimated by the CT or transmission scan taken before or after the main emission scan. Motion during the scan means that the position of the subject for at least part of the emission scan is different to that corresponding to the density map.

The result is that when the corrections are applied, the attenuation and scatter estimated will not be accurate, and so applying the correction in fact introduces artefacts in to the image. (Or, of course, phamacokinetic models.) This effect is explored and contrasted

with attenuation correction from a transmission scan, in [54] for internal motions in the bowel & in [60] for breathing.

The effect of erroneous attenuation correction, and their diagnostic impact, is known to be significant [34]. The artefacts are especially pronounced near boundaries between low and high attenuations – for example between the bottom of the lung and the top of the liver. They cause the activity in such regions to be over- or under-estimated.

Given the nature of scatter correction by simulation (as in [92]) motion would be expected to cause errors in the estimated scatter as well. In addition to the miss-alignment of the density map (and therefore errors in the simulation) the correction algorithm requires an estimate of tracer distribution and therefore will be influenced any attenuation correction artefacts.

3.2.3.1 PET-MR

As the MRI scan does not carry with it the radiation burden that a CT scan there is less reason not to take multiple scans to estimate the density map at different times, enabled by the fact that the fields of view for the two modalities overlap.

Leaving aside the need for further work on attenuation correction from MR data, there is also the desire to be able to acquire a clinically useful MRI scan at the same time as the PET study. It is quite easy to switch the MRI scanner between the tasks, and so the ideal scenario would be to conduct a PET scan concurrently with an MRI study, interrupting the latter as and when needed to acquire attenuation correction data following motions.

3.3 Conclusion

Development of PET hardware and reconstruction methods continues to improve the spatial resolution of the modality towards its ultimate limit of the position range. However, the resolution already available is much finer than the displacements encountered in typical motions. In other words, when motions occur they are already a binding constraint on the quality of the images produced and that might impact the medical decisions taken based on the PET images.

The effects of motions will become more relevant in the future: although FDG is currently the mainstay of PET imaging part of the modality's promise is the infinite variety of potential tracers that could be designed. Most of these will, unlike FDG, require dynamic studies, and these can take anything up to several hours to perform, increasing the likelihood of motions to a near certainty.

Further, the quantitative potential of PET renders it more vulnerable to motion artefacts – interpreting the image with errors as a 'picture' is hard enough for a clinician, making decisions based on number that may, or may not, be incorrect is due to motion artefacts harder still. Choosing to use the modality in a quantitative manner raises the standard of accuracy by which the results must be judged.

> *From here, it seems that tackling motion in PET is a worthy objective. There are two sides to this: first detecting what motion, if any, has occurred, second correcting the motion. These topics are what this thesis will concentrate on.*

Chapter II: Experimental Data Sources

This chapter presences a discussion of various sources of data available for experimental work including those used for work presented in the following chapters. First, however, some themes regarding data selection for experimental work are discussed.

This material has been placed here so that it can serve as a common discussion for both the registration and twitch detection experiments, which will be presented separately in later chapters.

1. Themes

1.1 Motivation

Experimental work may serve many different purposes, and, as will become a theme in this chapter, the characteristics of the data need to suit the purposes for which it will be used. For the sake of discussion, it is useful to simplify the range of possible uses to a set of categories, in this case four: testing, demonstration, validation, and assessment.

Testing: To ensure that an implementation works. This can take the form of formal testing, e.g. unit testing, or more informal, ad-hoc testing.

Demonstration: To demonstrate the potential of a method. Exactly what this entails depends on the audience, and what aspects of the methods are to be illustrated.

Validation: Comparison of the results of the method against other established methods, or, if possible, some ground truth results to show that the new method produces 'correct' results.

Assessment: Characterization of the method's performance. For example, how robust is it to noise? How accurate are the results?

Over time, methods become established. That process is a combination of the method being understood by those who work in the field and sufficiently wide variety of experimental results being available that the methods capabilities and limitations are generally known. Reference implementations, canonical uses and examples, publications, et al. all contribute, but a large body of experimental results using a variety of relevant data sets is key.

1.2 Trade-offs

Realism might be considered to be the most desirable characteristic for data to have. However, there are trade-offs to having realism in the data, and they determine what sort of data is most useful for any particular set of experiments.

Consider ***ground truth***: knowing the result that should be returned by the method. For example, knowing when a twitch occurs or the size and direction of the displacement. The most realistic of data – real data of a real subject on a real scanner – has practical limitations regarding ground truth.

In the case of motion experiments, it is possible to install a jig that allows precise, pre-measured, rigid body motions to be made on demand. That is not so easy for non-rigid motions (with regard to the time of the twitch, ground truth is still fairly easy to obtain). In other scenarios, ground truth is nigh on impossible with real data – for work involving physiological measurements ground truth simply may not be available; after all, PET is one of the leading forms of functional imaging.

Related to the need for ground truth are the issues of repeatability and variation. Repeatability refers to the ability to recreate the same event or process several times in a set of test data; for example, creating the same motion for twitch detection. This allows the consistency of a method's results to be assessed. Variation is the opposite, the ability to introduce controlled differences into the data – e.g. progressively increasing the size of a motion, which would allow the limit of a twitch detector's sensitivity to be investigated. Repeatability and variation are, therefore, very important characteristics for data that will be used for assessment or validation experiments.

Convenience should also not be underestimated. Real data requires live subjects and that has consequences – subject welfare, ethical approval etc. – especially for PET where the radiation burden of the tracer and attenuation correction CT must be considered.

1.3 Meaning of 'Realistic'

What does it take for data to be 'realistic'? In the context of this thesis, i.e. looking at motion in PET, especially before reconstruction, there is a range of factors that are important.

The noise in the data should have the correct properties – pattern (i.e. variation spatially – across the image or across the sinogram), distribution (in a statistical sense) and amount. Scatter and attenuation effects should be correct. Tracer kinetics, both physiological and from the decay of the tracer activity, are also important.

The total amount of activity also needs to be reasonable. This is important not just for the method's performance, but also from an implementation standpoint – can an implementation handle the number of counts being generated? Related to this is the data's format – its limitations, any artefacts that can occur in real data (e.g. high count rates overloading the detectors or electronics).

Another, particularly awkward, category of characteristics that need to be right include the shapes in the subject (the distribution of tracer) and the size, speed and type of motions. These are awkward because it is difficult to explicitly define what is normal or reasonable; what is a reasonable approximation to the shape of a brain, or does a child's image need to be considered differently from an adults? In some scenarios,for example the development of registration or segmentation methods, it is even harder: patients are, by definition, unusual in some way – consider registration of a patient with a tumour vs. that of a healthy person.

Some of these aspects of realism can be recreated through simulation. Simulation also allows a choice to be made regarding which aspects of realism are included and which are ignored. This raises a question: are there unrealistic aspects that are, in fact, desirable?

Extreme cases can be helpful: best cases (e.g. without any noise) or worst case (artificially high noise levels) for proof of concept during early development or stress testing while assessing a method, respectively. In the case of motions, the ability to change position instantaneously would be the extreme case of a fast motion and is impossible in reality.

Not to be overlooked is the data format: real data formats can be awkward to use, they are often designed for use in specific ways by software co-developed with the format. They are often proprietary. The work required to read real data formats an import them into whatever environment is being used to implement a method can be significant, it can be desirable especially for early testing and demonstration to have access to data that does not use the 'real' formats.

1.4 Summary

There are trade-offs to realism that must be considered, and sometimes unrealistic aspects are themselves desirable. The correct balance regarding these trade-offs depends on the intended purpose of the experimental work. Further, to achieve the objectives of a set of experiments there may need to be different experiments done on different types of data.

However, there is a place for real data, especially for demonstration and validation studies, and testing of final implementations that will have to be run on real data in practice.

2. Types of Data

2.1 Simulated Data

As touched upon already, simulation provides a means to control the level of realism and the characteristics of the data. Aspects of the imaging process can be selectively included or ignored. Moreover, simulated data usually provides known ground truth, as well as precise repeatability and variation. Compared to performing real scans, simulation is cheap and convenient. Advanced simulation can be time consuming, plus the capital cost of the software and/or hardware required to run simulations can be considerable, but the marginal cost of performing a simulation is only the computational time (and both, compared to the equivalent costs for real data, are small).

There are different ways to simulate data and several approaches will be discussed in turn.

2.1.1 Projection

This involves taking a numerical phantom, i.e. an array of values representing tracer activity levels, and performing a radon transform to create a sinogram. This is about as far removed from reality as possible, all it achieves is ensuring the sinogram matches the numerical phantom. It is, however, a convenient starting point for adding more realism or basic debugging of an implementation of an algorithm.

The convenience is that it can be performed fairly easily in almost any development environment; e.g. in Matlab there is a built in function for the Radon transform, and it would be easy to implement in a lower level language, if not already available in a library.

Note that there is no concept of time in what has been described, and time is vital for motion studies (for obvious reasons). A solution for this, converting static sinograms into 'pseudo list mode data', will be described later in this chapter.

2.1.2 Projection-plus

Building on a basic projection, other effects can be added, manipulating either the resulting sinogram or, in some cases, the phantom before projection.

The bin counts on the projected sinogram can be scaled down to control the count rate: real sinograms are typically quite sparse and consist of only integer counts, the latter means the sinogram bin counts from the initial Radon transform must be rounded off.

Noise can be added by interpreting the sinogram bin values as the means of Poisson random variables and using these to generate new values for each bin (which avoids the need to round off to integer values).

It is even possible to add attenuation: a numerical attenuation map can be projected to form an attenuation sinogram, where each bin is the total attenuation along that path. The bins of the emission sinogram can then be scaled by corresponding attenuation bins (by division, rounded off to the nearest integer of course) – almost the reverse of the early correction methods like [13].

Scatter is more difficult: the phantom and/or the sinogram can be blurred to produce something akin to scatter, a spreading of counts from one part to another. The problem is that the amount of redistribution (and where the redistributed counts end up) depends on the density of the subject – that information is captured by the attenuation map, but not easily incorporated using simple filtering etc. Scatter correction can be based on simulation [92], which could be adapted for use in reverses to add scatter... this,

however, brings the discussion to use of Monte Carlo methods to generate simulated data.

2.1.3 Monte Carlo Simulation

PET imaging is a stochastic process – i.e. what is observed is one of a set of possible outcomes, known only to a probability density function before hand. For a trivial example, consider that the total number of emissions for a region is a Poisson distributed random variable, with mean determined by the amount of tracer in the region. Other processes, such as the effects of attenuation and scatter on photons, are also predictable, but only to a probability density function.

Monte Carlo simulation involves conducting many individual simulations, using the probability density functions modelling the stochastic aspects in order to choose, at random, an outcome for those steps. By following the process, in this case the life of a count from decay to detection by the scanner, over many individual cases yields and estimate for the whole population.

Monte Carlo methods can be used in two ways. First, they can be used to estimate a composite probability density function so that the eventual outcome can be directly estimated for a future case. Secondly, they can be used here to generate a sinogram by simulating many individual photons.

Such simulation is capable of producing very realistic data. Also, because it works by following a process through its constituent steps, it is comparatively easy to design a Monte Carlo simulation from first principles (compared to, say, working out how to incorporate the attenuation map in to a spatially varying blurring filter to apply scatter in the 'projection-plus' framework).

Simulations using this sort of method can take into account most of the details of PET imaging, including the scanner's hardware, with factors like gaps between the scintillation crystals and electronics, like saturation of individual channels of the data acquisition system if count rates are to high.

Further, there is still complete control over ground truth, from the numerical phantoms of tracer distribution and attenuation that the simulations would be based on. (The tracer distribution could itself be simulated using a phamacokinetic model of course...) In fact, it is possible to access more ground truth by intercepting steps in the simulation. For example, the counts that have been scattered could be counted separately during simulation, giving access not just to a realistic sinogram but also a sinogram of scattered counts against which a scatter correction method's results could be compared. Although the computation involved in running a simulation is not trivial, simulations are cheap to run and it is possible to repeat them many times (exactly or with controlled variations).

2.1.4 PET-SORTEO

The Monte-Carlo simulation system available for this work was PET-SORTEO [12], [73]. Others are available, such as GATE [39], PeneloPET [21], SimSET [86], et al. but SORTEO was the one available and is capable of producing realistic sinograms from numerical phantoms for an ECAT Exact HR+ [72]. PET-SORTEO includes a few tricks in how it performs the simulation that enable it to achieve much better performance than would be possible following a naive simulation of the photons produced by every decay.

The key to the system's performance is that the simulation is split in two: an initial, detailed simulation of single photons and then a second 'coincidences' simulation used to generate the resulting sinogram. The first step is used to generate summary probability density functions, combining many individual photon simulations. Then coincidences

(direct and scattered) and randoms can all be generated in the second part of the simulation, using these summary probability density functions instead of the many individual steps, and thus saving time.

With reference to the earlier discussion of Monte Carlo methods in the previous section, the initial simulation generates composite probability density functions and the second set of simulations use these to generate an overall sinogram.

The simulation is also implemented in parallel, with emissions from various regions being performed independently and then superposed to form the overall sinogram. Regions are largely independent and thus their separated sinograms can be combined together by superposition, except for the dead time effects at the detectors in the scanner (i.e. the phenomenon that after photons are detected the scanner takes some time to recover – the scintillation crystals – and electronics must process the data). The initial singles simulation means that the dead time effects can be modelled by a probability density function (the chances of a photon that reaches the scanner being recorded in the data) that is estimated along with the summary probability density functions.

Recent developments of SORTEO [50] have implemented list mode output. However, the version available for use during this work only produced sinograms. Therefore, as with all the methods presented so far, there must be a means of converting a sinogram back to list mode, undoing the histogramming.

2.1.5 Pseudo-LMF

Motion studies obviously require time, and thus list mode data. (Or at the very least a sequence of sinograms, but given that all commercially available scanners nowadays are able to produce list mode data it seems natural to treat the formation of a sequence of sinograms – if required – as a step in a method.) Therefore there is a need to smear the

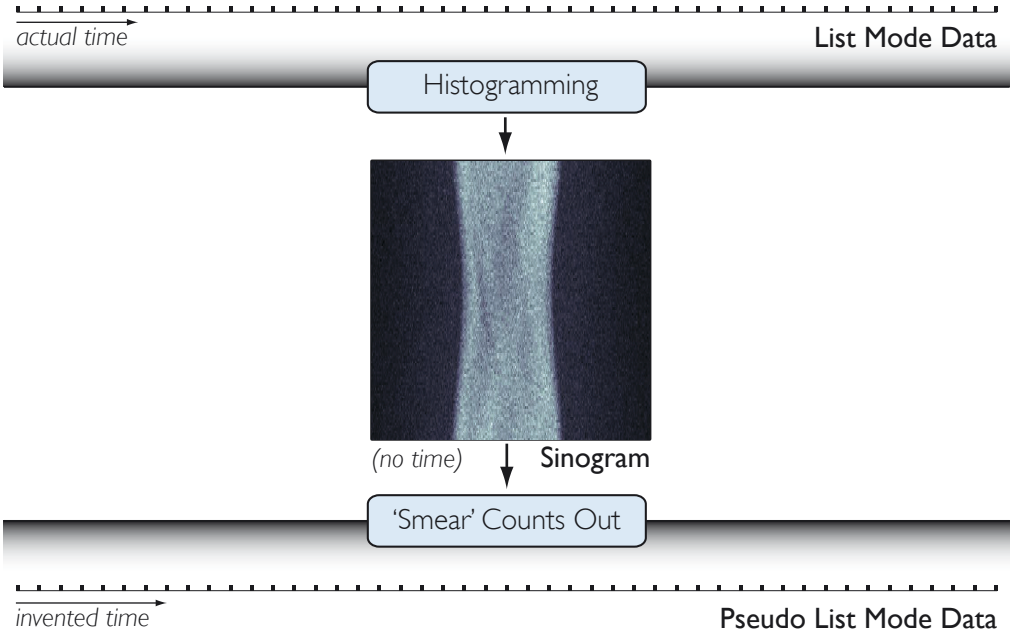

Figure 11 Pseudo list mode data as a recreation of list mode data, undoing the loss of a sense of time that results from the histogramming step.

counts in the simulated sinograms over time, in effect reversing the act of histogramming them, so that pseudo list mode data was available for experimental work – Figure 11.

The time information (when each individual count was recorded) is lost during histogramming. New times must, therefore, be invented for every count. As the decay of the tracer follows an exponential pattern, being a radioactive decay, the gaps between individual events can be modelled as a random variable following an ***exponential distribution*** (6), which has a parameter λ, for the current rate of decay. That varies over time, and depends on the initial count rate and the half-life of the decay. The latter is fixed by the tracer being simulated, and the former can be specified, either directly or by specifying the duration of the scan simulated to produce the sinogram (and thus approximately how long the list mode data should last).

$$\mathrm{p}(x|\lambda) = \begin{cases} \lambda e^{-\lambda x} & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases} \quad \textbf{(6)}$$

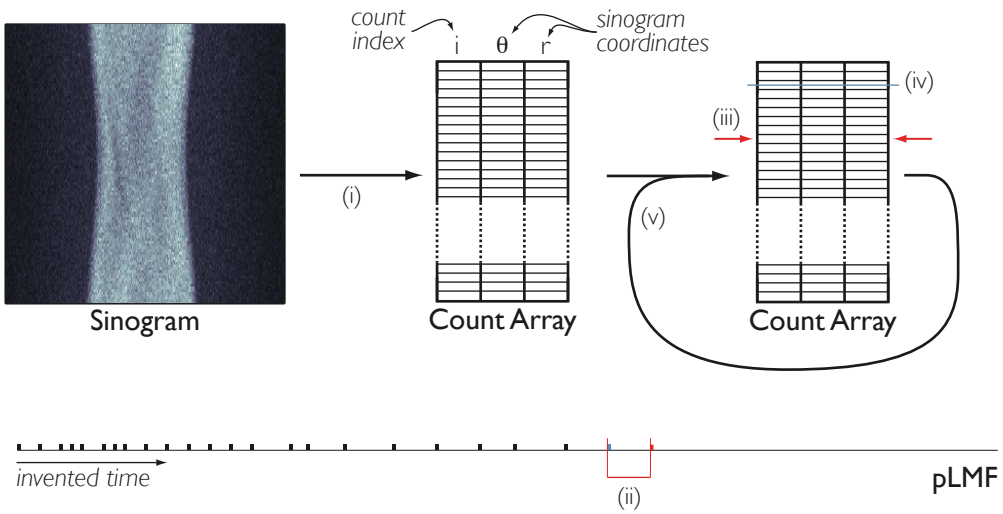

Figure 12 Generation of the Pseudo List Mode Format (pLMF) data. First the sinogram is expanded into a count array, such that every event counted in the sinogram is listed separately with its sinogram coordinates – shown here in the 2D case (i). The gap between events is chosen using an exponentially distributed random variable (ii). This sets the time of the next event, the coordinates are selected by choosing an entry in the count array at random (iii). Once chosen the entry is removed from the count array (iv), and the process repeated (v) until all counts are converted to events in the pLMF stream.

By choosing each gap at random the times for all the events included in the sinogram can be made up. Next, counts in the sinogram need to be assigned to the times – this correspondence is also lost during the histogramming as there is no record of the order in which events were histogrammed.

This is done by choosing a count at random from the sinogram, without replacement, until all the counts have been removed. This can be done by listing the sinogram bin coordinates for each count and selecting one using a random integer to choose the element in that list to be used for any given event. That list element is then removed. That way the same sinogram coordinates can be assigned to another event, but only if there are multiple counts in that bin on the sinogram. See Figure 12. Histogramming the resulting pseudo list mode data (pLMF) will exactly reproduce the original sinogram.

There are some shortcomings to this approach. First, although the tracer's half life is modelled, the pharmacokinetics of the tracer are not. For example, the tracer could start in one region and move to another over the course of the scan; sinogram coordinates

pertaining to the former should be assigned to events early in the list mode more often that those pertaining to the latter (and vice versa), but this cannot currently be done, it would require specifying multiple rate functions, on a regional basis.

Second, real list mode data records the hardware addresses of the two crystals in the scanner that detected photons. This information is lost in histogramming, as multiple crystal pairs contribute to each sinogram bin. This means that the pLMF cannot be used to create real list mode format data. Thus pLMF data cannot provide a source of data to test code for importing real data (for instance), but, as all the methods presented in the rest of this thesis operate in the sinogram domain and do not use the crystal addresses, the pLMF approach can still produce useful data for experiments using implementations of the method that do not include the import and conversion of real list mode data.

It could be possible to assign crystal pairs at random to events (such that the pair assigned would count in the relevant sinogram bin. To do this properly, however, would require a means to estimate the distribution of the crystal pairs within each sinogram bin. Estimating the distributions is a problem, especially given that there will only be a handful of counts in any bin for any given scan, and the distributions will depend on the tracer distribution and attenuation map so are specific to the subject in question (and potentially vary over the course of a scan).

> *Given that problem, and the complexity of authoring real list mode data (discussed in more detail in §1.1 of Chapter V), the pLMF approach was only developed as far as producing data consisting of event times and sinogram co-ordinates. It was used for ad hoc experimentation during development of the basic twitch detection method, but when testing latter implementations built to operate on real data, real data was itself available.*

2.1.6 Clinical and Preclinical Data

Real data is usually categorised into ***preclinical*** and ***clinical*** data. Preclinical refers to small animal (mice and rat) work, whereas clinical work involves humans. (Sometimes experiments are done on larger animals, say pigs or primates, and these need to be done on full size, 'clinical', scanners, which is awkward given the above categorization. In the following, most of the differences that will be highlighted between the two have to do with animal versus human, so that is the sense in which the two terms will generally be used.)

The most obvious difference (and, ironically, one not related to the distinction between humans and animals) is size. Small animals hold less tracer and exhibit less attenuation or scatter. Indeed, it is only comparatively recently that preclinical PET-CT machines have been available, before then attenuation was generally ignored – but perhaps to some degree the perception that it was not a problem may have been a result of there not being a solution against which to compare the results.

Size also affects motion studies because the size of the motion relative to that of the subject and the resolution of the scanner must be considered when comparing pre-clinical against clinical experiments. Likewise, the sizes of spatial details in the subject relative to the scanner's resolution need consideration. A balance needs to be struck between a large enough motion to be visible (with regard to the spatial resolution) yet small enough to be relevant when scaled up to human sizes.

It could, therefore, be argued that clinical data would be the superior form of real data. That ignores the fact that there are some compelling use cases for the work presented in this thesis, especially that on twitch detection, in the preclinical domain. Unlike humans one cannot ask animals to hold still for a scan – not that all patients can. Indeed, animals

are anaesthetised for scans and so often are better behaved, but that does not completely prevent twitches.

This points at the more significant difference between clinical and preclinical work: the practical and ethical concerns of each. It is much easier to conduct experiments involving animals than humans, although, especially in the United Kingdom, there are significant legal restrictions on animal work.

The radiation burden, from the tracer and attenuation correction CT, mean that it is difficult to justify a PET scan without some other motivation. In the case of things like computer aided diagnosis tools it is necessary for healthy 'normals' to be scanned, as well as 'patients'. For the motion studies required by this thesis, it would not be justified to scan people, unless in the context of a procedure that they would be undergoing anyway. Even in such cases there would be a reticence to add extra steps to the procedure, because of the stress to the patient and the extra time taken, when the scanner could be in use with another patient. Lastly, people have to give their consent and this further contributes to a shortage of subjects for experimental work. With animals it is much easier to schedule scans specifically for experiments – and therefore have more freedom in the design of those experiments.

In such respects animal studies are much easier, and it is possible to do some things with an animal study that is not possible with humans. As an extreme example: in the context of motion, it can be desirable to isolate a specific motion from those due to ongoing processes like the heart beat and breathing. This requires killing the subject, so that the only motions are those deliberately created. This can be done, in some cases, with an animal but obviously not with a human.

As already mentioned, animals are routinely anaesthetised for scans, which helps regarding extraneous motions. Note, however, that anaesthetics could well affect tracer

pharmacokinetcs. It is possible to scan an animal for longer, in part because the anaesthesia reduces the stress on the animal.

With anaesthesia (or by killing the animal) it is possible to use a small animal as an 'organic phantom'. Phantoms are another source of data: a container filled with known concentrations of tracer placed in a real scanner. Phantoms range from simple shapes made out of cylinders to idealised brains. Although artificial phantoms provide known ground truth with respect to tracer concentration, and it is usually possible to get numerical versions for simulations, there is the problem of how relevant the shape of the phantom is, as well as not having relevant pharmacokinetics from the tracer. Using small animals as organic phantoms satisfies both of these, which are more relevant to motion studies than the known tracer concentrations. (Incidentally, after the scan an animal can be dissected, and the radioactivity of each organ measured, to provide ground truth tracer concentration in static scans).

The ground truth needed for motion studies as in this thesis is the time of motions (for twitch detection) and the size and direction of the motion (for registration). The former is easily achieved by noting the times at which motions are made using dead or anaesthetised animals, the latter is possible using a jig that allows the animal to be moved inside the scanner by preset amounts on demand. Small animal work was, therefore, considered more useful than phantom work.

3. Conclusion

The recurrent theme through this chapter has been that there are trade-offs between the characteristics of different sources of data. Depending on the purpose of experimental work one type of data may be more suitable than others, and a combination may be needed to achieve all the objectives.

Within the scope of what is presented in the following chapters the experimental work is, predominantly, to demonstrate the methods and explore some of the factors affecting their performance. Some validation has also been considered. In addition to that, during development of the method, various aspects of the methods or their implementation has required testing. In general those experiments do not warrant specific discussion – in so far as the implementations worked well enough to perform the actual experimental work the testing in development can be considered a success!

The main source of data used has been simulation. For the registration methods, during development 'projection-plus' was the main source of data due to its convenience; SORTEO simulations were used for investigating its performance with some preclinical studies used to provide a final demonstration with real data. In the case of the twitch detection method, the characteristics of the noise are more important to the method, so initial 'development' experiments were done using SORTEO simulations (with pLMF conversion), and later investigations used preclinical scans.

The preclinical work used 3 studies, one initial scan with a dead mouse, then a later study using two rats, one alive, the other not. The mouse scan was performed to provide some initial data to help develop and test an implementation of the twitch detection method that used real data. It also led to the design of the jig for moving the animal by preset amounts.

The rat scans were to study the performance of the method with and without the extra motions of breathing and the heart beat. Using the larger rats made it easier to cause small motions relative to the size of the animal. Also, less of the rat is in the field of view of the PET scanner than with a mouse, which is more similar to the case when imaging humans. The jig allowed precise, repeatable, rigid body motions to be made with a range of sizes. In addition, more realistic motions were added by moving a leg or the tail (in the latter case the size of the motion is harder to quantify and repeat, but the timing is still known).

The specific parameters of the specific data used in each set of experiments will be listed latter, along side the relevant results.

Chapter III: Registration in the Sinogram Domain

This chapter concerns the work done on registration – the process of aligning two images by estimating the motion between them and then removing it, by applying the opposite transformation. In the case of PET the reservations about errors in, and the computational cost of, reconstruction discussed earlier suggested analysing the estimation & correction using sinogram data, and thus before reconstruction.

During this work the detection of the times at which motions occurred was recognized as the key problem for tackling twitches. This will be solved in later chapters of this thesis. So that the narrative represents the course of the work and establishes the motivation for twitch detection this work on registration will be presented fully here.

1. Introduction

Registration is the term given to the process of estimating the transformation required to align two images (usually) and then transforming one of them so that it best aligns with the other. The result in this context is a single image, as if there had been no motion. This can be used to fuse the two images together or to enable comparison between them – the latter for treatment follow-up, to give one example, seeing if a tumour has shrunk after (and hopefully as a result of) chemotherapy. The first use, the fusion of images, is more applicable to the overall objective in this work, namely the reduction of motion artefacts.

The situation is more complicated when there are multiple changes between the two images. Consider an object that has moved in front of a stationary background, or, continuing the above example, the change of position of the patient between the two scans combined with the change in the size of the tumour. The registration procedure must be designed to remove the unwanted change, while preserving the differences between the images which are of interest. Figure 13 shows, in schematic form, a registration done to allow local changes to be isolated and understood.

1.1 Registration Terminology

Conventionally the images are referred to as 'template' and 'reference'; the template image is deformed to match the reference image. They are aligned by estimating the parameters of a transformation that best matches them. In other words:

$$\min_{\mathbf{x}} f\left(T_{\mathbf{x}}\left(I_t\right), I_r\right) \tag{7}$$

in which, T_x is the transformation, with parameters x, used to align the template image I_t to the reference I_r. f is a cost function and provides the definition of 'matching', different

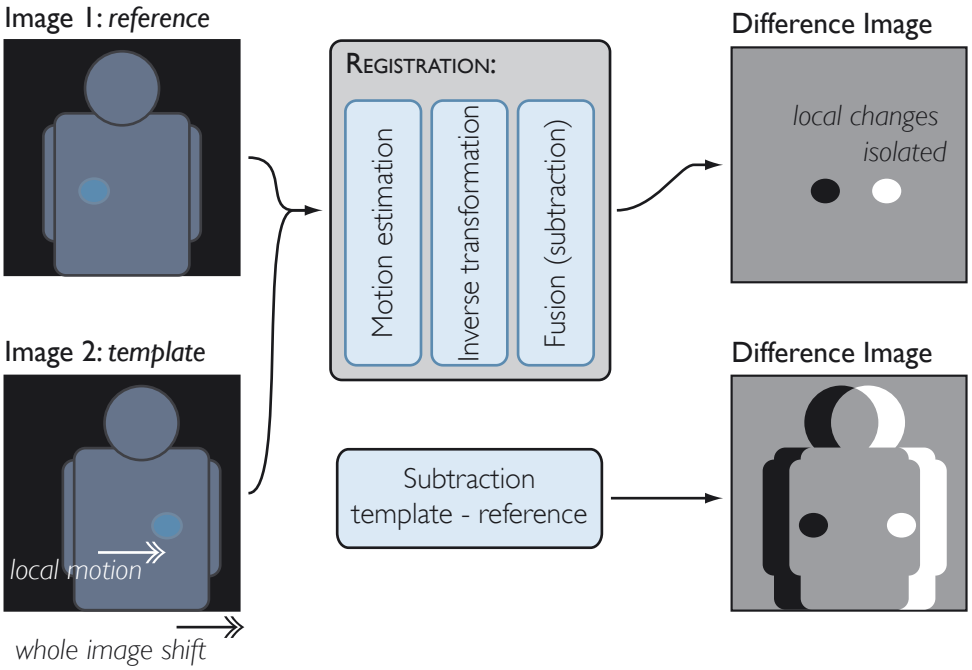

Figure 13 Schematic demonstration of the use of registration to align the position of the subject between two images so that local changes within the subject can be isolated and understood.

cost functions will compare the images in different ways. Varying these items allows the registration to be tuned to the desired use case.

Registration is a very wide field [93], even if just medical imaging applications are considered [48]. Some salient examples of use cases are follow-up, e.g. the first stages of [79], multi-modality fusion – [70] discusses the benefit for combining PET with anatomical modalities (before widespread availability of hybrid PET-CT scanner), and motion correction, e.g. to align the gates of a gated acquisition as in [16].

Closely related to registration, and arguably a subcase thereof, is the field of optical flow [3]; the key difference is that in optical flow there is an assumption that the total image intensity (or whatever aspect of the image is used to drive the algorithm) has not changed between the two images– as a result optical flow is best suited to 'cine' image sequences with fine temporal sampling.

The model of the motion expected is captured in the choice of transformation. Imaging the brain, rigid motions would be expected, but if the objective were to measure shrinkage or expansion (as in [8], which finds that the Jacobian of a deformable registration to be an effective tool for investigating atrophy in Alzheimer's disease) then the interior of the head would have to be allowed to deform – but it would be reasonable to prevent parts of the image folding over, or rearranging themselves in a physiologically unreasonable fashion. Particularly with deformable registration the transformation model must capture the allowable range of motions in a way that is computationally tractable and makes it easy to incorporated constraints (like no folding). For example, [4] makes use of radial basis functions for fusing mammography images with MRI data, and B-splines are another popular choice, used, for example, in [15] to allow volume changes to be constrained.

The cost function often consists of two parts, a ***data term*** and a ***regularisation term***. The former measures how well the two images match one another, the latter uses a priori preferences about the motion to qualify the data term's value such that the overall value expresses the quality of the alignment, rather than just the image similarity – if a smooth motion field is expected it might mean a less than perfect match is the preferred alignment. The data term could be based of image intensity – point-wise comparison, e.g. using sum-of-squared differences, or considering the distribution of many points by measures like mutual information [47] – or featured based, in which key features of the image like landmark points or edges are found in each image and then matched together. The regularisation term allows prior preferences, eg. that the motion field be smooth, to be incorporated during the optimization.

The optimization – the min in (7) – is the way that the parameters of the motion model are chosen. How this is best done, e.g. by exhaustive search (as later in this chapter), gradient decent (typically for smooth cost functions) or a dynamic programming algorithm (to match pairs of features in a feature based approach), depends on the nature of

the cost function and the motion model. It must balance the need to find the best match with the ability to do so quickly.

At this point various enhancements might be added. Multiscale methods, eg. in [14], start by performing a course registration and then proceed to finer resolutions in an attempt to avoid erroneous registrations cased by local minima in the cost function while still using fast optimization algorithms. In a feature matching approach an 'atlas' of typical inter-feature relationships could be used to improve feature identification, as in [81].

1.2 Registration for Twitch Correction

In the use case considered by this thesis, there is a concern that the motion artefacts, for instance wrong attenuation correction in one image, could influence the registration. This suggests that the registration should use images without any attenuation or scatter correction applied. Only after the data is transformed to match the position when the attenuation correction scan was taken should it be reconstructed to form an image. (However – the additional detail recovered by attenuation correction, for instance, may help the registration algorithm sufficiently to outweigh the detrimental effects of the artefacts. Thus, although in principle it is best to perform registrations on uncorrected data, pragmatism may indicate otherwise.)

This concern leads to the concept of pre-reconstruction registration: estimating, and correcting, a motion with the sinogram domain data, before the complication of reconstruction or the corrections within. (See Figure 14 for an illustration.) This also reduces the number of reconstructions required, which, given the computational cost of iterative reconstruction, is also a good thing.

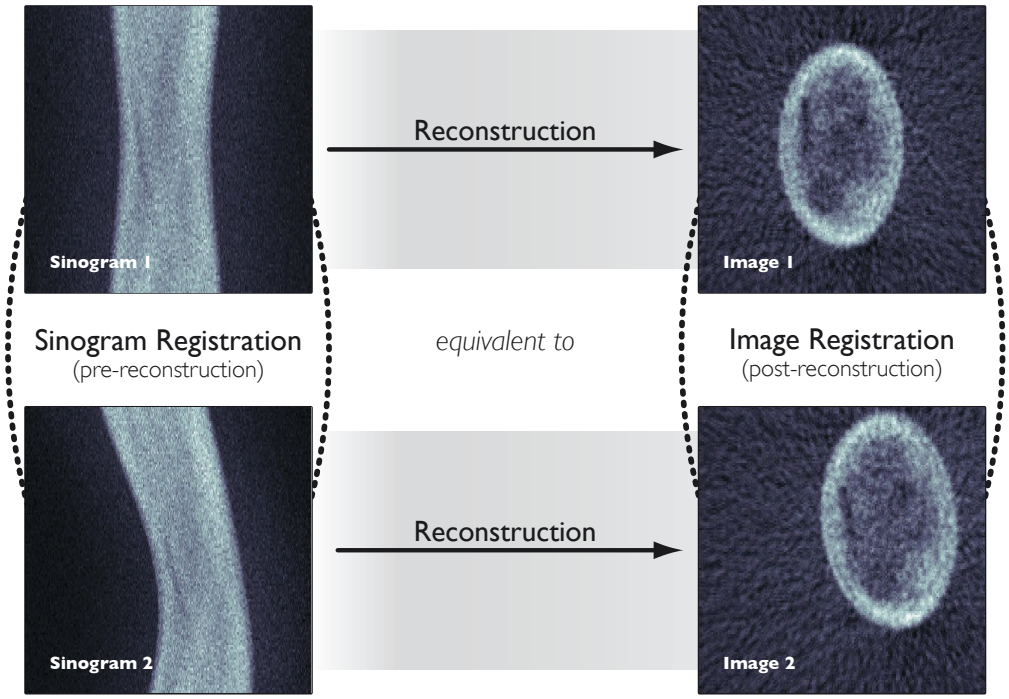

Figure 14 Schematic illustrating the concept of pre-registration reconstruction. As all the information needed to form the images is present in the sinogram data (except, of course, for that introduced by any priors used in reconstruction) the motion could be estimated by comparing the sinogram before reconstruction, as opposed to afterwards by comparing the resulting images.

Of interest was [46], in which a very basic discussion of the manifestation of motion on the sinogram is discussed, with the addition that the motion is estimated from a few bright points in the object (it is suggested that these are additional radioactive markers added to the subject) that produce easily identifiable sinusoid on the sinogram. (It also does not address the need to identify when the motion occurred, although does highlight the possibility to do so from the sinogram).

Motion correction when the scan can be divided into frames by some external means, reconstructed separately, then registered, is discussed in [69] and [82]. In these cases it is assumed that the motions are measured using a motion capture camera system. Manipulation of list mode data to remove a motion is discussed in [27] and [71], so that separate images do not need to be reconstructed and then transformed (which can alter the values through resampling etc.).

While developing ideas for performing registration in the sinogram domain, related work was found in the CT literature: [25], [26], and [24] (all by E. E. Fichard, et al). That method was designed for registering treatment planning CT scans to the data acquired during radiotherapy, and closely paralleled the ideas being developed for this thesis. Therefore, the methods presented below build upon that work, but they are significantly adapted for use with the very different image conditions of PET and with substantial development of the implementation details relative to those published by Fichard.

The method that has been developed successfully performs a rigid body registration of the PET data without reconstruction, if better for translation than rotation. It is reasonably robust to low count numbers and to errors in estimating when the motion occurred, so that part of the data contains a mix of counts from both positions. It also shows a significant amount of flexibility: the registration can be symmetrical (so the choice of which segment of data is the template and which is the reference has no effect) and independent of scale – meaning that if one part of the data is of longer duration or otherwise contains significantly more counts than the other, the registration can still be performed (depending on the cost functions used).

However, the method presented assumes that there is a means to estimate when the motion occurred. Indeed, there is substantial literature proposing other methods of registration in PET (and evaluating their performance) – but only if the knowledge of when the motion occurred is given, but no method is useful without a means of locating when a twitch occurred.

> *It became quite clear in investigating registration that twitch detection is an important step, and one that is usually ignored. This is understandable in the case of work seeking to align two different scans for follow up etc.; but vital if the aim is to undo the effects of the twitches.*

2. Method

Being pre-reconstruction registration, the 'common' elements of §1.1 are somewhat less obvious in the following. They are present, but some aspects are compounded together (e.g. the motion model and regularisation), others spread through the method at different points (e.g. the data term of a cost function).

> *During the examination of the thesis it was realised that certain aspects of the method as implemented by Fitchard were misunderstood. The result is that this implementation is more 'original' than perhaps it might have been, although given the differences Fichard's actual method would likely have performed better. The differences have been highlighted in this version of the thesis, although the experimental work etc. remains the same.*

2.1 Theory

2.1.1 Basic Principles

Consider the two dimensional case. A rigid body motion, such as might be caused by a twitch of the neck muscles while the brain is being scanned, can be decomposed into two parts: a rotation and a translation. Mathematically, the two components may be applied in either order without loss of generality; but, in the following, the rotation is considered to be applied first, about the origin, and then the rotated object is translated. This is conceptually simpler since the centre of rotation is automatically known to be the origin (instead of the point that was at the origin before the translation was applied).

Separately, the translation and rotation have sufficiently direct effects on the projections that make up the sinogram domain data that they can be estimated quite simply – the

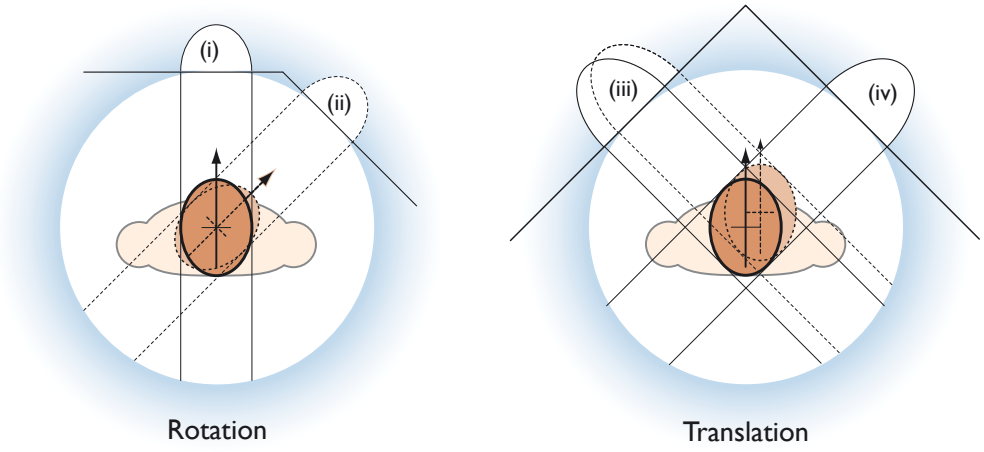

Figure 15 Illustration of the effects of rigid body motion on projections. A rotation about the centre of projection will cause the projection (i) before rotation to correspond to projection (ii) after rotation. A translation causes the projections to shift, the amount depending on the direction of projection relative to the direction of motion. In the case of (iii) the motion is perpendicular to the projection and the maximum shift is observed, matching the magnitude of the motion. For (iv) the motion and projection are parallel, so no shift is seen.

tricky case is when the two sets of effects are confounded. To prepare for that more typical case, the separate cases are considered first.

Rotation of the object about the origin corresponds to a reordering the rows of the sinogram, in effect redefining the 'zero-angle' for projection, see Figure 15, (i) & (ii). Figure 16 shows a similar example, but in this case for the most basic image, a point source, from which more complicated effect can be considered by superposition. It follows that the rotation could be estimated by comparing the rows of the sinograms of the two segments of scan data, and removed by reordering them.

Translation causes a systematic pattern of shifts of the projections – i.e. shifts of each sinogram row, along the row. Consider the projection taken in a direction perpendicular to that of the motion. The projection after the translation will appear to have been shifted by the same amount as the motion. For the projection corresponding to the head on view of the motion there will be no change, see Figure 15, (iii) & (iv). (PET data is formed by parallel beam projection, so these hold exactly.) Between the two extremes the variation is sinusoidal – the amount of shift effect seen as a fraction of the overall

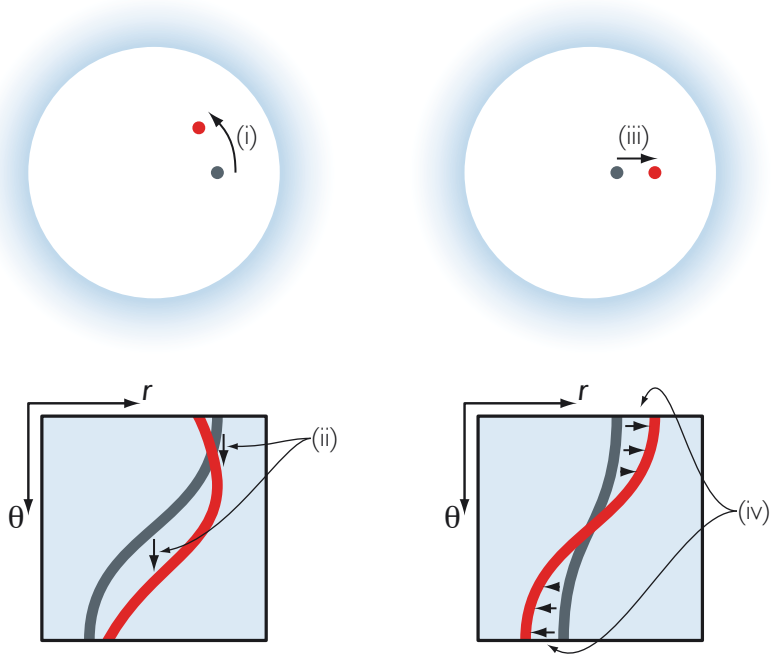

Figure 16 Illustration of the independent effect of rotation and translation on the sinogram. Here a point source is used; as the radon transform is linear it is valid to consider the subject as a set of point sources and the sinogram a superposition of the resulting sinusoids. In (i) there is a rotation and this causes (ii), a phase shift in θ on the sinogram (as discussed in §2.7 of Figure 9, the sinusoids phase is given by the angle component of the point's polar coordinates). The translation in (iii) increases the amplitude of the sinusoid (again, as discussed earlier) and therefore the row-wise shift (iv) varies sinusoidally itself.

displacements varies as the sine of the angle between the projection and the motion, again see Figure 16. Thus the overall translation – characterised by the magnitude of the displacement and its direction – can be found from the magnitude and phase of the sinusoidal pattern of intra-row shifts with respect to projection angle.

Thus the two tasks are individually straightforward. However, the situation when the two effects are mixed is more complicated, and this is addressed next, along with the details of implementation, by walking through the process of performing a registration between two sinograms in 2D.

2.1.2 Rotation, Independent of Translation

To decouple the translation and rotation effects, the rotation is estimated in the Fourier domain – crucially, taking the magnitudes of the Fourier transform removes the effect

of any translation. (This also reinforces the decision to consider rigid body motion as a rotation followed by a translation.)

$$\mathrm{abs}\left(\mathfrak{F}\left(\mathrm{T}_{x,y}\cdot\mathrm{R}_{\alpha}\cdot I\right)\right)\equiv\mathrm{abs}\left(\mathfrak{F}\left(\mathrm{R}_{\alpha}\cdot I\right)\right) \quad \textbf{(8)}$$

Where F is the Fourier transform, $\mathsf{T}_{x,y}$ is a translation operator, and R_{α} a rotation operator.

Ignore, for the moment, the need to avoid reconstruction. Rotating an image will cause a redistribution of horizontal and vertical frequency content such that the 2D frequency spectrum of the image is rotated by an angle equal to the image's rotation. Thus the rotation between a pair of images can be found independently of any translation by finding the angle of rotation that best matches the 2D frequency spectra.

$$\mathfrak{F}\left(\mathrm{R}_{\alpha}\cdot I\right)\equiv\mathrm{R}_{\alpha}\cdot\mathfrak{F}\left(I\right) \quad \textbf{(9)}$$

The key to performing this process without reconstructing two images is the ***Fourier Central Slice Theorem***, as demonstrated in Figure 17. This is a well known theorem in tomographic reconstruction as it forms the basis of back projection algorithms (which are still widely used in CT). It states that the one-dimensional Fourier transform of a projection is identical to the slice through the two dimensional Fourier transform of the image being projected, where the slice is made through the origin of the 2D transform at the same angle as the projection. Equation (10) is perhaps best understood using Figure 17.

$$\left(\mathfrak{F}_{2\mathrm{D}}\cdot I\right)[f,k]\equiv\left(\mathfrak{F}_{1\mathrm{D}}\cdot\left(\mathfrak{R}\cdot I\right)[\theta,r]\right)[\omega] \quad \Big| \quad \tan\left(\theta\right)=\frac{f}{k} \quad \textbf{(10)}$$

In this method each row of the two sinograms (each projection angle) is Fourier transformed separately and the vector of the magnitudes stacked up in the same order to produce a new pair of arrays, in which rows correspond to angles and columns to frequency components. For want of a better name, let these arrays be called '***sliceograms***' - Q in equation (11). See also Figure 18.

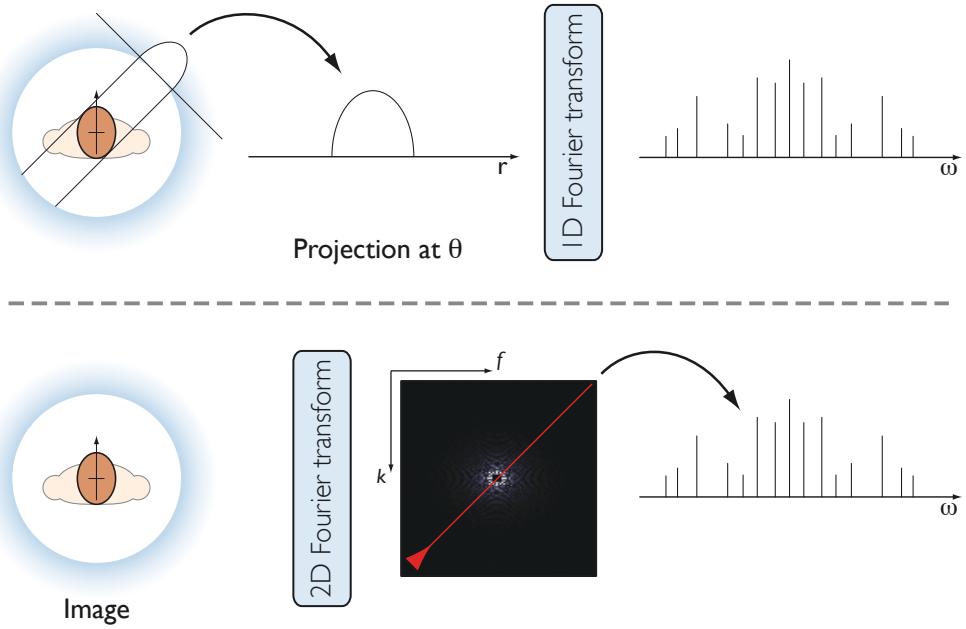

Figure 17 The Fourier Central Slice Theorem states that the 1D Fourier transform of a projection (at some angle) – top row – is the same as a slice through the 2D Fourier transform of the original image at the same angle – bottom row.

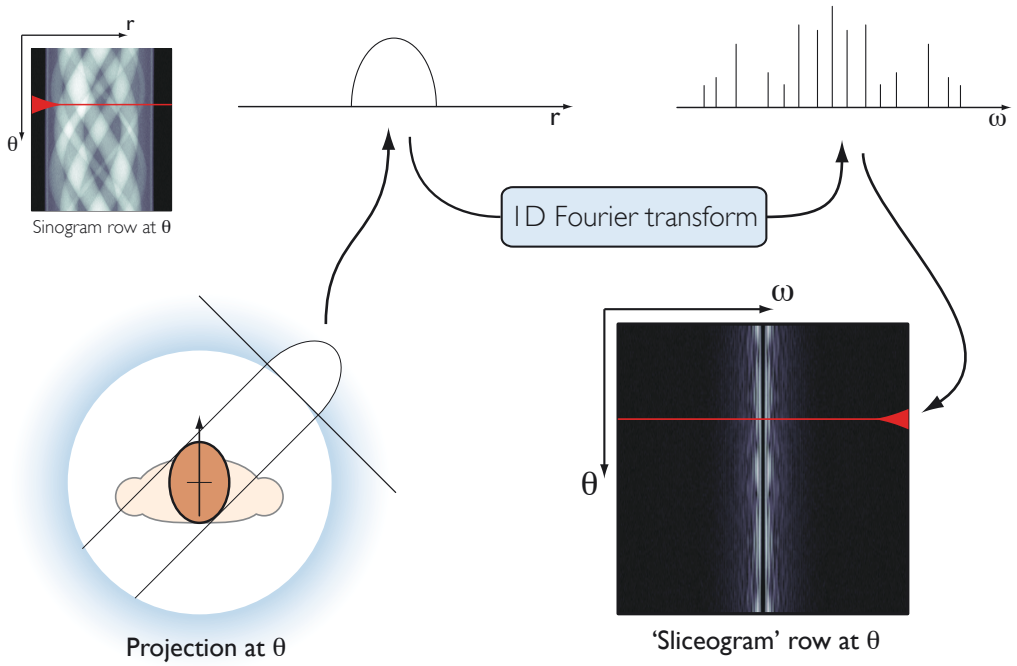

Figure 18 Schematic illustration of the definition of a 'sliceogram', illustrating the calculation of one row of the array. NB only the magnitude of the components are considered, to ensure any translation is ignored.

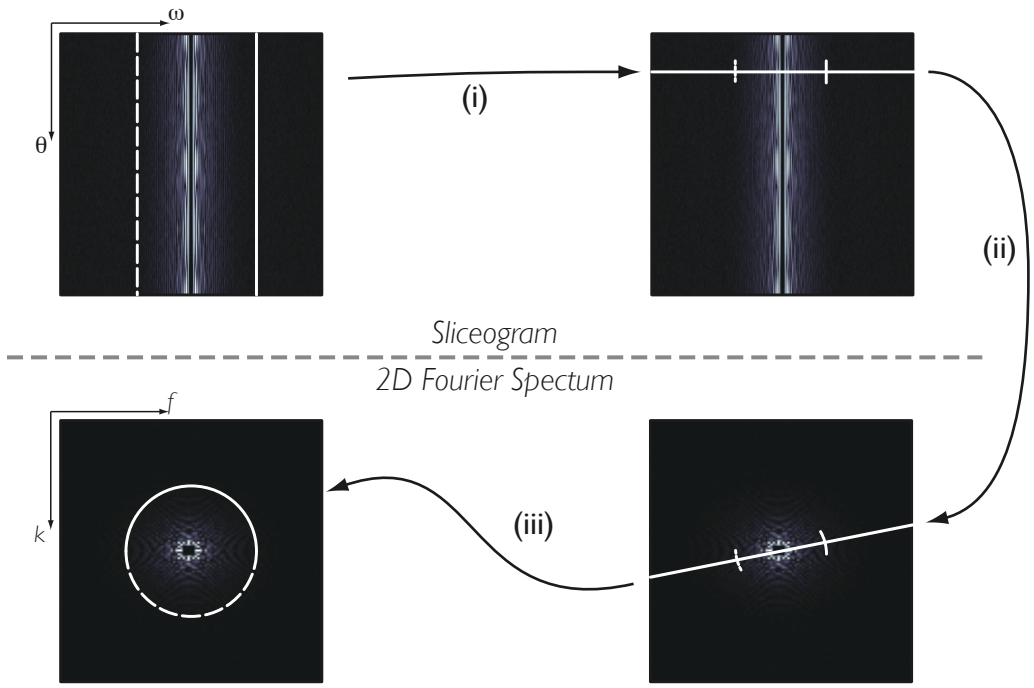

Figure 19 (i) To follow the columns on the sliceogram consider one row, tracking the elements corresponding to those columns . (ii) As each row of the sliceogram is a 1D Fourier spectrum of a projection it corresponds to a slice through a 2D Fourier spectrum by the Fourier central slice theorem. (iii) Repeating this for all possible angles traces out the ring in the 2D frequency space. Note the inherent symmetry between positive and negative frequencies (dashed versus solid lines).

$$Q\left[\theta,\omega\right] \triangleq \left(\mathfrak{F}_{1\mathrm{D}} \cdot \left(\mathrm{R}_{\alpha} \cdot I\right)\left[\theta,r\right]\right)\left[\omega\right] \qquad \textbf{(11)}$$

By the Fourier central slice theorem, these columns correspond to rings in the 2D frequency spectrum of the image that would result from reconstruction, as Figure 19.

Thus by comparing the columns of the two sliceograms and finding the number of rows offset that best matches them against each other the rotation can be estimated. Not that the alignment of each sliceogram column is itself a little 1D registration task. Note that there is a symmetry, which follows from the co-linear photons emitted by positron emission (the same symmetry that makes the projections from π to 2π redundant) – the columns wrap around and link up with the column for the same frequency component of opposite sign.

Finding the row offset that makes the best match requires some form of metric – **g()** in equation (12) – for which the maximum implies the best match so that the row offset is

the **argmax** over all possible rotations **[-π,π)**. One example of a measure would be cross-correlation – this metric will be discussed later after both the rotation and translation methods have been explained, as both require a metric.

$$\hat{\alpha}[\omega] = \arg\max_{\delta} \; \mathrm{g}\left(Q_t\left[\theta+\delta,\omega\right], Q_r\left[\theta,\omega\right]\right) \qquad \textbf{(12)}$$

(In which Q_t and Q_r are the template and reference sliceograms respectively.)

In theory, the rotation estimated from every column of the sliceogram should be the same. This is because the row offset directly corresponds to a rotation angle so even as the higher frequency components, and thus rings in frequency space of greater radius, are compared the increment of shift is also increased. In practice, that different frequency components contain different information is important. This will be addressed later in §2.3.1.

> *Please note that in Fichard's work, the whole array is registered in one go, rather than individual estimated for each column. This would be better in principle as it allows consistent local maxima to influence the outcome, whereas in the approach used here only the global maximum of the metric on each column makes any contribution.*

The rotation can be removed by reordering the rows of one of the two sinograms. Remember that projections that loop off the ends of the sinogram to the other end must be flipped, because of the truncation of the sinogram to **[0,π)**. In reality, the rotation would move rows off one end to the other end of a **[0,2π)** sinogram, and thus push projections past π - it is the symmetry about π that causes the apparent flipping.

The ability to rotate the image by modifying the indexing of the data means that the numerical considerations normally encountered when applying rotations are avoided. Rotations are often described using a rotation matrix, but this contains several redundancies (6, in three dimensions) and the consistency required by the redundancies cannot be guaranteed with limited numerical precision – this compels the use of other formulations for expressing rotations, like quaternions, without the redundancies. Performing the correction for the rotation to the data in the sinogram domain avoids all this.

The intermediate result at this point is a pair of rotation registered sinograms: the reference sinogram has not been altered but the template sinogram has had the rotation removed so that the only remaining differences derive from the translation. This 'rotated' sinogram can then be compared to the reference sinogram to estimate translation.

2.1.3 Translation, rotation removed

From this point, with the rotation removed, estimation of the translation follows very much as outlined earlier. Each pair of corresponding rows from the reference and rotated sinograms are compared using an appropriate form of metric and the column offset that best matches the pair found. Again, this sub-step is a little 1D registration task. During this comparison, it makes sense to zero-pad the rows: zero padding implies that outside the field of view of the scanner there is nothing – which is a perfectly valid restriction given that the objective of the scan is to image the subject, not the surrounding world!

As explained earlier, the translation-induced shift varies sinusoidally with projection angle as a different component of the translation is visible from different angles. For PET, having a parallel-beam projection geometry (unlike CT, for instance, which has a fan-beam geometry), this relationship is exact: the pairs of rows should exhibit only a shift in the projections and the variation with projection angle is as $m \sin(\phi - \theta)$, where

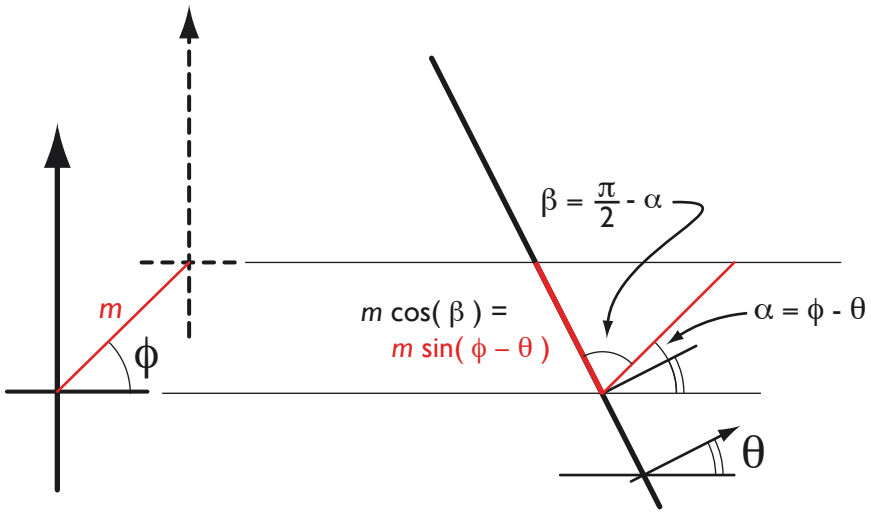

Figure 20 Shift for a translation of size *m*, as seen by on the projection at θ.

m is the distance of the translation (measured by columns of the sinogram space) and ϕ the direction (in terms of the projection, or rows of the sinogram) – Figure 20.

Note that the frequency of the sinusoidal variation is known – it is a consequence of the projection, not of the motion. For a given motion, *m* and ϕ are effectively constant, therefore *m* sin(ϕ - θ) varies only with θ.

This provides a convenient means to estimate the magnitude and phase: they can be found from the fundamental component in the Fourier transform of the row-wise shifts. This automatically combines all the row wise estimates and in doing so makes the estimation more robust, as in practice individual pairs of rows may produce erroneous shifts due to noise etc.

It should be possible to directly estimate all both the phase and the magnitude, rather than estimating row-wise offsets and then fitting a sinusoid through them, and this might produce better results, by not restricting consideration to only the global maximum of each row's alignment. This the same argument as for Fichard's rotational

correlation of the whole sliceogram, although it is quite clear in those papers that the translation method is as described here.

The translation can be removed by shifting the rows of the rotated sinogram by the opposite sinusoidal pattern of shifts. This produces a registered sinogram, whose contents can then be combined with the reference sinogram to form a single data set ready for reconstruction.

2.2 Metrics

Up to this point, the means used to compare the two segments of data have deliberately not been made explicit. That is because the choice of measure is the only parameter inherent in the basic method.

A ***metric*** is defined to be a function that takes two vectors in $\mathbb{R}^n$ (where *n* is the length of the vectors) and calculates a scalar value expressing their similarity. (The term 'metric' has specific meaning in the context of vector algebra, and in that case there are additional properties required. Here this more relaxed definition will suffice.) For the comparison while estimating rotation, the length is equal to the number or rows on the sliceogram (which is equal to the number of rows on the sinogram). For translation, the length must allow for motion, so may be somewhat greater than the number of columns on the sinogram.

$$\begin{aligned} \mathrm{f}_r &: \mathbb{R}^n, \mathbb{R}^n \rightarrow \mathbb{R}^1 \\ \mathrm{f}_t &: \mathbb{R}^n, \mathbb{R}^n \rightarrow \mathbb{R}^1 \end{aligned} \tag{13}$$

For the sake of convention, let the maximum of the metric correspond to the greatest similarity. Further, let the 'comparison function' refer to the metric wrapped in such a way that the vectors are shifted appropriately (with the looping for rotation, zero-

padding for translation) by some amount, then compared, such that the best shift for a given pair of vectors can be found as the argmax of the comparison function:

$$\begin{aligned} \mathrm{f}_r \,|\mathbb{R}^n, \mathbb{R}^n : \mathbb{R}^1 \rightarrow \mathbb{R}^1 \\ \mathrm{f}_t \,|\mathbb{R}^n, \mathbb{R}^n : \mathbb{R}^1 \rightarrow \mathbb{R}^1 \end{aligned} \tag{14}$$

Although one could argue that there is merit in simplicity and consistency to using the same metric for both rotation and translation it must be remembered that the two parts of the registration involve different sorts of data and so the meaning of the metric may be different: the rotation metric operates on vectors consisting of a given frequency component from several separate 1D Fourier transforms, the translation metric operates on the rows of the sinogram – so line integrals of tracer activity through the subject.

2.2.1 Properties of the Metrics

The properties of the measures directly impact some of the properties of the overall registration algorithm.

2.2.1.1 Meaning of similarity

What is meant by 'similarity' is determined by the measure functions. Simple metrics based on a sum of differences (e.g. absolute or squared) perform multiple local comparisons and aggregate the total. Using squared differences places more emphasis on elements of the vectors which have greater differences than does using absolute differences. More complicated metrics could include a more inherently global comparison, for example the comparison could also be made in a statistical framework, akin to the comparison done by a distance measure such as Kullbach-Leibler for comparing probability density functions. Alternatively the metric could seek to identify features in the vectors and map them to each other.

2.2.1.2 Robustness

Robustness has, in this scenario, two possible meanings. One is robustness to low count numbers (and therefore low signal to noise), the other concerns what happens when the scan is incorrectly divided into sections so that one sinogram actually contains some mix of data from both positions.

– To Low Count Numbers

The former is strongly influenced by averaging many separate estimates of rotation from comparing multiple columns of the sliceogram or from the averaging inherent in the estimation of the sinusoidal pattern during the translation step. There are a few implementation tweaks mentioned in §2.3 as to how to do this averaging. What is, however, relevant to a discussion of metric choice is that with low count numbers the discrete nature of sinogram data becomes important. (Note, here discrete refers not to the sampling, rather the quantization of each sinogram bin's count that follows from the fact that one can only observe an integer number of photon pairs – there is no such thing as a half-decay. Although the underling Poisson processes may have smoothly varying means, any given set of observations can only be integers, resulting in Poisson noise.)

With PET, given the Poisson nature of the data, signal strength (count numbers) and SNR are linked. Thus note the nature of the noise – at low count numbers the difference between Poisson and Normal distributions is important and so one should be cautious of any implicit assumption of normality in the metric.

For the translation case, these effects are directly visible to the metric function. For the rotation case they are masked by the Fourier transform step and then further compounded by the fact that the vectors being compared are a composite of many different Fourier transforms. There is also a variation of the information present in the different frequency

components. These two factors become relevant when averaging the multiple estimates of rotation together, which is discussed later on page page 86.

– **To Framing Error**

Robustness to framing error is another matter. In large part, this is something that has been assumed negligible – detecting when motions occurred is clearly an important step, enough to warrant specific work on that subject, discussed later in this thesis. For a quick motion (where quick means that the duration of the motion is very small compared to the duration of the data for one of the sinograms) the effect is a mixing of the data from the two positions. This would appear on the projections as a ghosting, of a shift from translation plus another row from rotation.

So for small motions, this error would blur projections, for larger motions the projection would start to exhibit a secondary pattern. This is most relevant if the metric function seeks to match patterns, for the translation stage. For the rotation stage the Fourier transform further complicates matters, as the spatial separation of the two patterns is lost during the transform. For both, it should be expected that the more significant the misestimation of the motion time, the harder the registration task and, crucially, there will be a bias towards underestimating the motion.

2.2.1.3 Behaviour of Comparison Function

Another consideration for the metric function is the behaviour of the comparison function around the maximum. Ideally, there would be one well defined, global maximum corresponding to the correct alignment of the two vectors. If there are to be local maxima then the correct alignment should be significantly more pronounced.

In practice, given the sizes of the vectors involved and the range of reasonable offsets that need to be checked, an exhaustive search to find the maximizer is possible, especially with computationally simple metrics. This means that the behaviour of the comparison function is somewhat less important than if a more complex optimization scheme were used. So long as the correct maximum is the global maximum the right offset will be found.

This counsels against using a complex metric – seeking to match features would not only be computationally much more expensive but also more prone to local maxima, and it is more difficult to ensure that a spurious local maximum will not wind up being the global maximum.

> *Here again, note that performing a correlation on the whole sliceogram simultaneously would make a difference to the argument, as what is 'local' in a column-wise sense may be global in an array-wise sense.*

2.2.1.4 Inverse consistency

It should not matter which segment of the scan is chosen to be the reference and which the template: the transforms estimated in each case should be the inverse of the other. However, the rigid body motion is usually only an approximation to the real motion and so the estimates will be partly erroneous and it is less obvious that the opposite errors will be estimated.

This property depends on the symmetry of the metric function: if the two vectors are exchanged, the offset that maximises their similarity should have the same magnitude, but opposite sign. This will then mean that the overall transformation estimated will be identical, but in the opposite direction.

2.2.1.5 Invertability

An invertible transformation can be mathematically inverted to create another transform that does the opposite motion, as needed to undo an estimated motion. For non-rigid registration, where a mapping can contain folds etc., this is a real concern. In these cases extra constraints must be imposed to prevent folding etc. Given that rigid body transformations are inherently invertible there is no concern about this (although, again rigid-body motion may only approximately describe the motion). Also, with rigid body motions, numerical problems when computing the inverse transform as less likely.

2.2.1.6 Scale Invariance

Lastly, a key feature which the comparison should have is some form of invariance to scale, especially relative scale. In other words – the estimation of the offset made by the comparison function should not be altered if the ratio of the magnitudes (by some norm) of the vectors changes.

In practice, twitches do not occur at regular intervals, therefore the segments of data that must be aligned to each other will not have similar numbers of counts. Indeed, it is not unreasonable to expect very large ratios of counts, for example if a spasm moves the head to a different position, but then later it returns to the original, more neutral, location.

At first glance it is only the location of the maximum that seems to be important, but, in practice, scale will have other effects that require consideration. Consider a metric based on sum of squared differences:

$$\mathrm{g}\left(\mathbf{x},\mathbf{y}\right)=\sum_{i}\left(\left(x_i-y_i\right)^2\right) \tag{15}$$

The maximizer of the comparison function based on this does not depend on the relative scale of x & y. However, if $y = \lambda x$ (so the correct shift is 0), then the larger elements of

the vectors have proportionally greater differences after being scaled and thus will exert more influence on the metric than other points, meaning that the behaviour of the comparison has changed.

2.3 2D Implementation Details

There are several implementation details that bear mention. These relate to how the multiple estimates of rotation and translation are averaged together to make the overall process more robust.

The exhaustive search of the comparison function for the best offsets, repeated many times for both translation and rotation, is not terribly efficient, neither is the calculation of the rotation estimates for all frequencies, nor, arguably, is the calculation of the shifts for all rows of the sinogram during the translation stage completely necessary. However, the method is ***non-iterative***. Thus one pass yields the best estimate of the motion, and even ignoring the other points the registration has proved quick enough in practice (using an un-optimized implementation in Matlab) with basic metrics like cross-correlation or sum-of-squared differences. It is the comparison function that consumes most of the computational time and thus determines the overall running speed. Note also that the 1D sub-registrations in both steps are ideal targets for parallel computation, being completely independent in terms of the data they use.

2.3.1 Rotation Implementation Details

In the case of rotation, the multiple estimates come from the multiple columns of the sliceogram, each corresponding to a different frequency component. This means that the estimates from different columns of the sliceogram will be influenced by different scales of structure within the image. The lower frequency columns will be related to large scale structures, higher frequency columns to finer scale details.

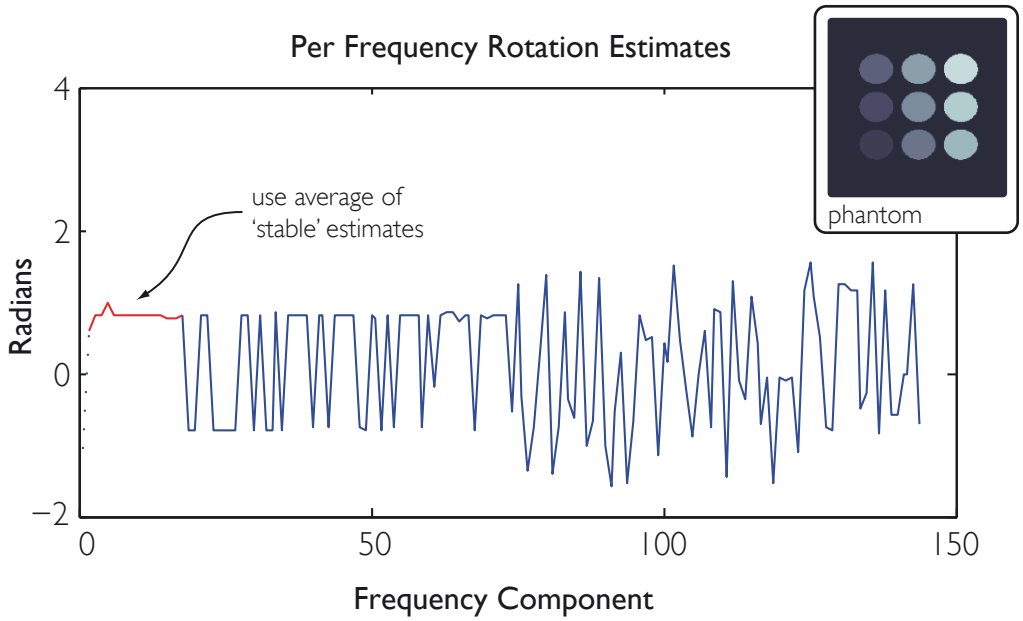

Figure 21 Example of the variation with frequency of the individual estimates for rotation. Low frequency components provide more stable estimates of the rotation and so an average of the stable estimates is used. *In this case the data used is a simulation of a software phantom (inset). The 'alternating' pattern seen in the middle frequencies is thought to be a result of the repeating structures in the phantom. At higher frequencies the noise prevails and the estimates show no pattern.*

Given the global nature of a rigid body rotation and the low spatial resolution of PET data (relative to the anatomical structures in the body) the low frequency estimates would be expected to prove more reliable and the high frequency terms would probably not contain any strong information. This has indeed been found to be the case – the low frequency estimates form a stable set of values, whereas among higher frequencies there is no consensus at all. It follows that the obvious strategy is to take an average of the low frequency estimates as the overall estimate, as illustrated in Figure 21.

The cut off between the stable (LF) and unstable (HF) estimates is chosen adaptivity, since this avoids an a priori choice. This is important in practice as well as being a philosophical nicety. The point at which the breakdown occurs depends on the level of noise (for instance, the quantization noise mentioned earlier) and the amount of rotational structure at that scale in the object being scanned. The relative impact of the two depends on the metric function, of course, but just as the amount of noise depends on the duration of the segments of the scan the structure is dependant on the object being scanned.

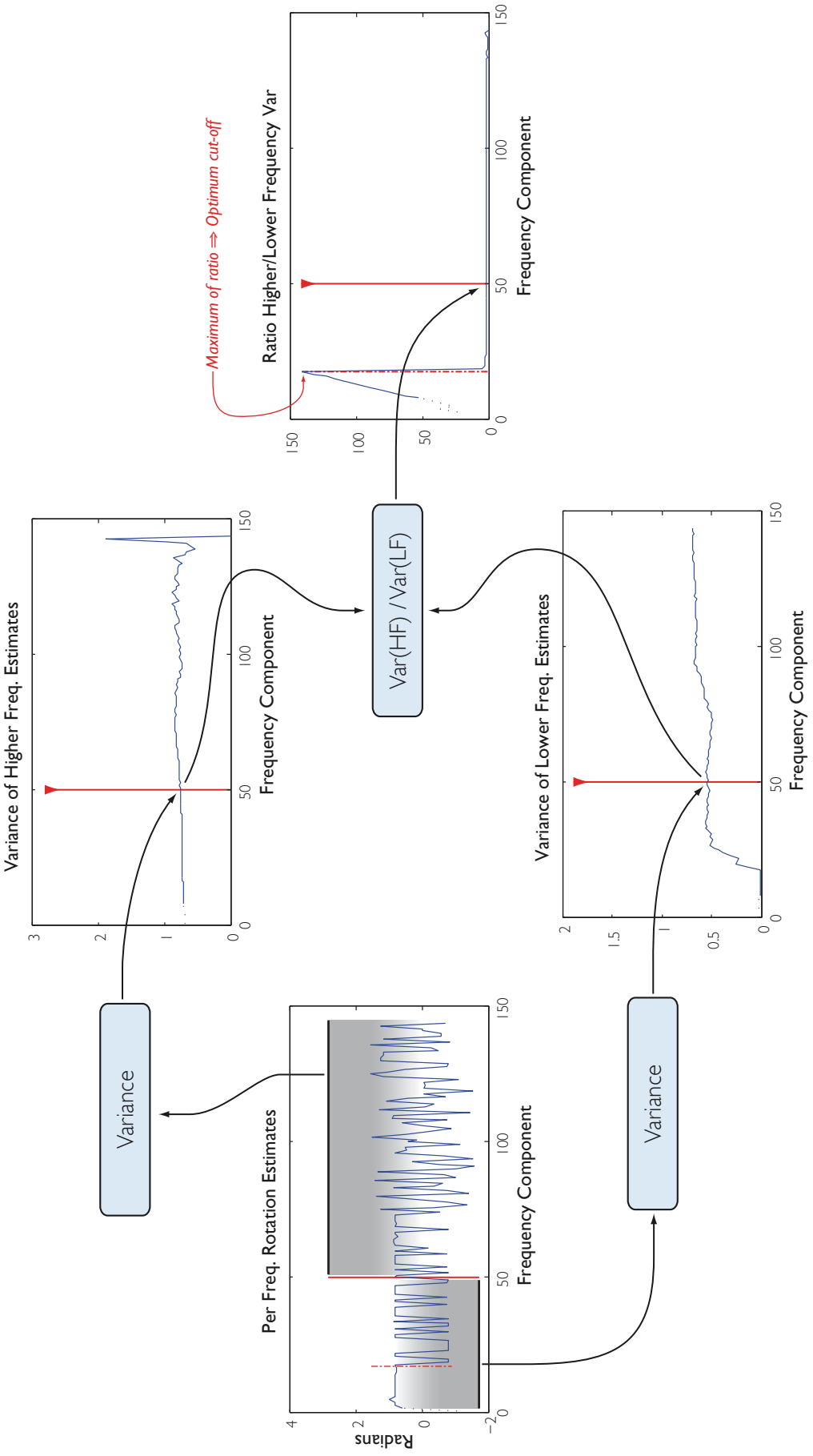

Figure 22 Schematic outlining the adaptive choice of the cutoff between stable, low frequency, rotation estimates and the unstable, high frequency estimates. Here the ratio is being computed for component 50. (The actual cutoff between stable and unstable frequencies is lower.)

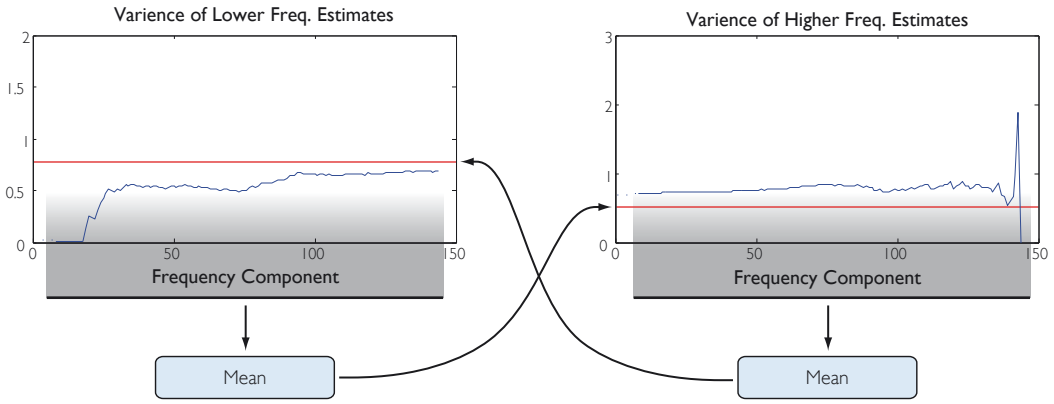

Figure 23 The validity check for rotation involves comparing the mean of all higher frequency components against that of all the lower frequency components. If there is a clear stable st of estimates then the mean of the lower frequency estimates will be lower than the mean of the higher frequency estimates.

The method that has been employed to identify the breakdown point is to find the frequency component for which the ratio of the variance of the set of higher frequency components to the variance of lower frequency components is a maximum – the process is summarized in Figure 22.

Variance is used as a convenient proxy for 'stability'. This approach is based on the premise that there are two populations of estimates – one characterized by being stable and consistent, the other by being the opposite. Further, the populations are expected to be contiguous so separating them is a matter of picking a threshold frequency. Picking this threshold using the ratio of variance seeks to make each population most like itself. Excluding a large number of stable estimates makes the unstable population's variance greater; excluding the unstable estimates from the stable population makes its variance lower.

Should the mean variance of the higher frequency estimates not be higher than the mean variance of the lower frequency estimates (Figure 23) then there is no clearly defined difference between stable and unstable estimates: failing this final validity check suggests that all the estimates are just noise and so the rotation estimate is set to zero.

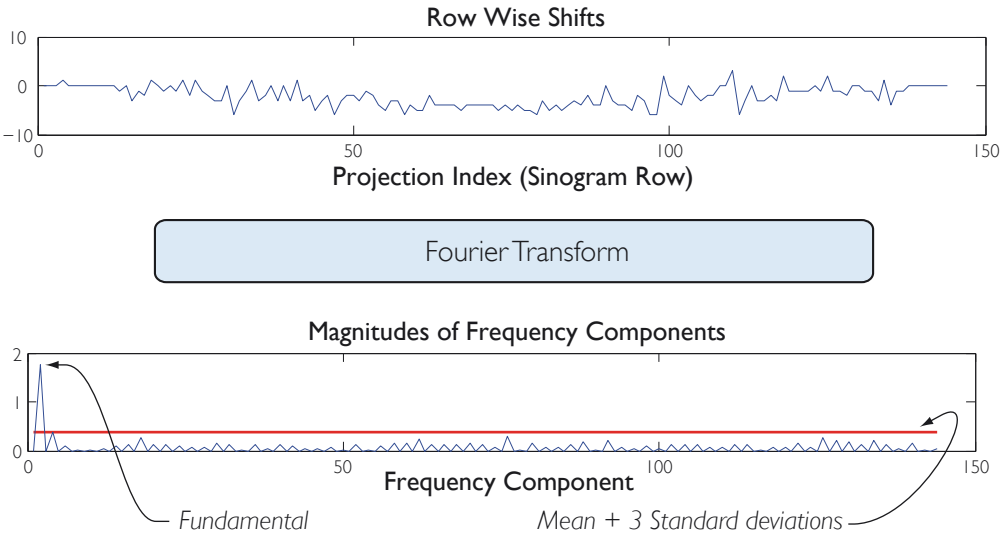

Figure 24 The validity check for translation is done after taking the Fourier transform of the individual, row-wise shifts. The magnitude of the fundamental is compares with a threshold set at the mean of all the magnitudes plus three standard deviations of the magnitudes.

If the whole sliceogram were cross-correlated simultaneously this process would not be necessary. However, there is a marked difference in the quality of estimates from different frequencies, albeit only their individual global maxima (all their secondary local maxima might well agree perfectly) – this suggests that some weighting might be beneficial even in a whole slicogram cross-correlation.

2.3.2 Translation Implementation Details

The translation step is simpler: as already discussed, the use of a Fourier transform to estimate the phase and magnitude of the sinusoidal pattern performs an averaging over all the row-wise shifts, which adds a good degree of robustness. There is no reason to be selective about which estimates are used as in the rotation step. The only implementation detail over the basic method is a validity check, mirroring that for the rotation.

Recall the Fourier transform used to estimate the phase and magnitude of the sinusoidal pattern: as the only frequency component (ignoring windowing effects) corresponding to the translation is the fundamental, if its magnitude is not significantly greater than

the others it is assumed better to ignore any estimated motion. 'Significantly greater' has been taken to mean three standard deviations above the mean (mean and standard deviation estimated from all other components except the fundamental) - see Figure 24.

This assumes a normal distribution – tenuous given the magnitudes cannot be negative. However, given that this step is only a fail safe it is deemed to be an acceptable approximation.

3. Experimental Work

Following from the discussion in Chapter II the purpose of the experimental work presented here is two fold. First is to investigate the performance of this approach to registration, underlining the role of the metric function one the outcome, with reference to the topics in §2.2. The second aim is to demonstrate the method.

The former objective will require ground truth to be available and this suggest the use of simulated data, as will the need to be able to control the properties of the data, chiefly the number of counts. For demonstration purposes it is easier to use real data: the data available has limitations discussed shortly, but it still makes for a more compelling demonstration than the simulated data, for reasons related to the shortcomings of the simulated data that will be described shortly.

3.1 Counts vs. Time

Throughout the following the amount of data used (from each position in registrations, for example) will be described in terms of an approximate number of counts, in contrast to using the duration of the data. Although the latter might be more intuitive the approach used is more portable.

More counts bring more information, and thus produce better images which should, for example, produce better registrations. Continuing with the image example: reconstructing a longer section of scan produces a better image than from a shorter section because the longer section will contain more counts. This, however, does not hold in the case of tracers with very short half lives if the longer section of data is drawn from sufficiently later in the scan than the short one because the decay of the tracer reduces the level of activity as the scan progresses. Similarly, between scans the amount of activity

initially injected can vary significantly and so the same duration of data may not produce the same quality of image.

Measuring the amount of data in terms of a number of counts avoids all of these questions. The results can be interpreted in any context by considering the count rates typical of the specific context.

3.2 Metrics

Three metrics were selected for evaluation. First is correlation, ***'Corr'***, as was used in the original papers by Fichard et al. Second is the sum of squared differences, ***'SSD'***, a typical and intuitive way of comparing vectors. With reference to equation (13), these metrics are:

$$\begin{aligned} &\text{(i)} \quad v_{corr} = V_1 \times V_2' \\ &\text{(ii)} \quad v_{ssd} = \sum_i \left(V_1[i] - V_2[i]\right)^2 \end{aligned} \tag{16}$$

For the rotation method, when the comparison is made in the frequency domain the magnitudes of the frequency spectra are compared, and the phase information ignored. The results of using these metrics proved virtually identical, indicating that they both attained a maximum at the same alignment given the same data. For brevity only those of the SSD metric are displayed here.

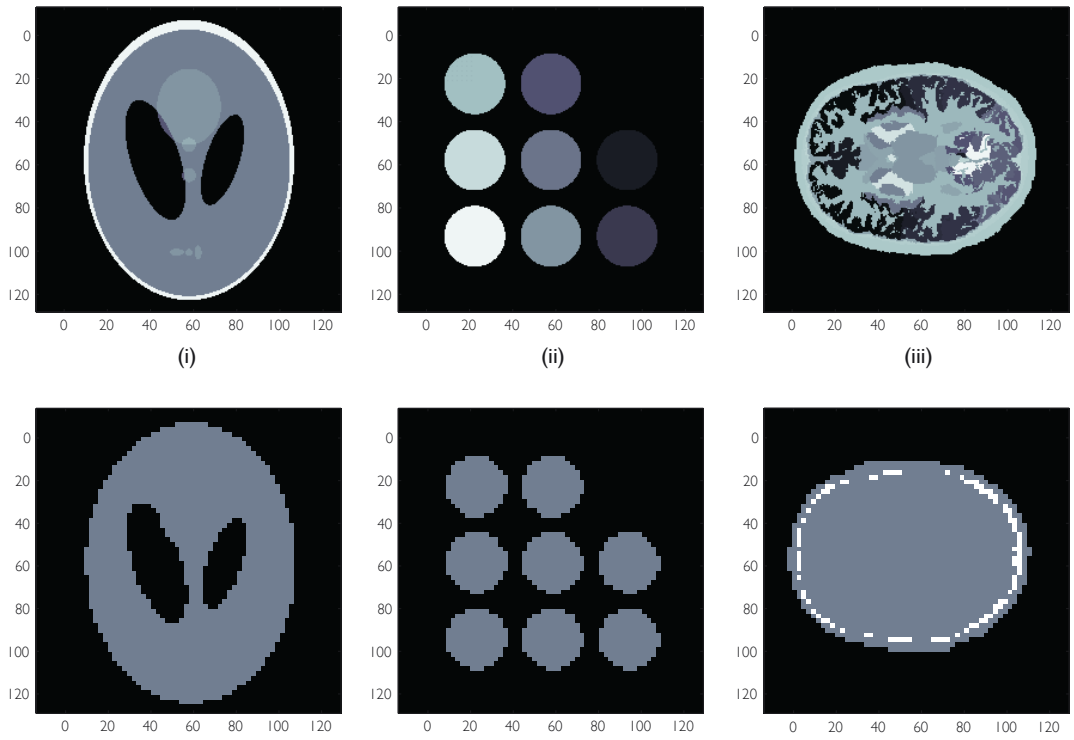

Figure 25 Cross sections of the phantoms used. (i) the modified Shepp Logan, (ii) the grid phantom, (iii) the brain phantom, and (iv)-(vi) show the attenuation maps for each.

3.3 Performance Investigation

3.3.1 PET-SORTEO Simulations

PET-SORTEO produces realistic sinograms given two volumes (labelling the emission and attenuation regions of the phantom) and a set of time-series, describing the activity in each emission region.

The choice of phantom is difficult: there is no obvious 'universal' phantom and the properties of the phantom will influence the performance of the registration – through the esoteric effects encountered in Figure 21, or through more mundane phenomena, such as rotational symmetry (e.g. it is impossible to determine the rotation of a uniform circle).

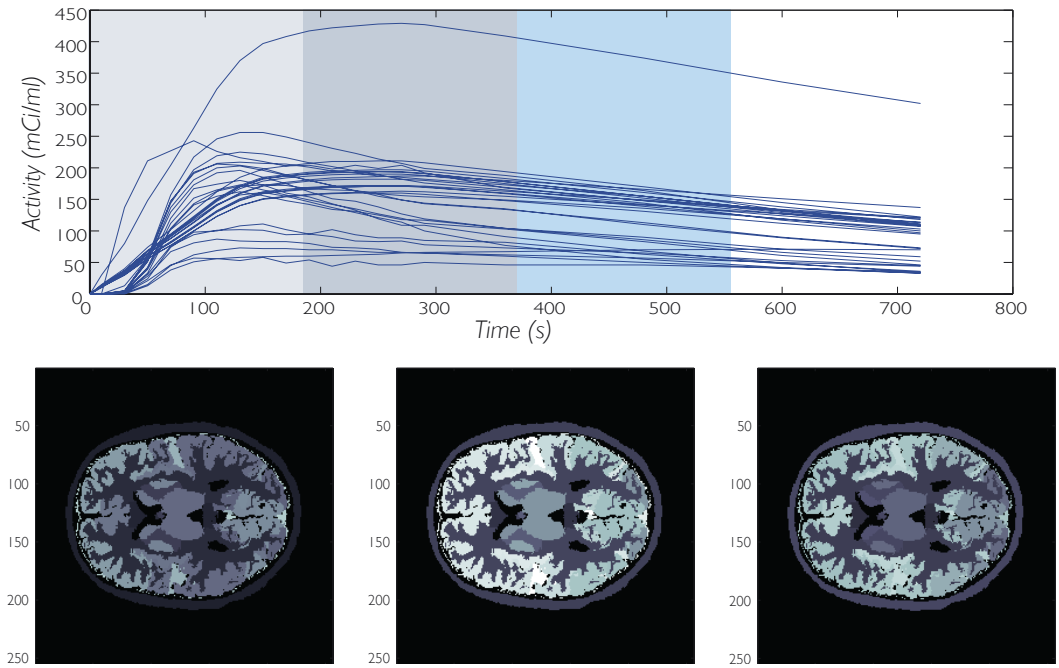

Figure 26 The three setcions of the time activity curved selected and the resulting tracer distributions.

All the following phantoms were specified as 2D cross sections as Figure 25. These were first transformed to put the phantom in various positions, then extruded to form a 3D volume (as required for simulation). The constant cross section is no limitation for the 2D registration and, in fact, allows data from several cross sections to be combined together (see later).

3.3.1.1 Modified Shepp-Logan Phantom

The Shepp-Logan phantom [77] is a classic phantom used in tomography, and so has been included as the closest thing there is to the elusive 'classic' phantom. However it was developed for transmission tomography (like CT) not for emission tomography so the phantom has been modified, re-mapping the values such that the interior has higher activity and thus is not so heavily outweighed by the exterior shell (which was originally designed to emulate the attenuation of the skull). For attenuation the entire phantom was considered to consist of water.

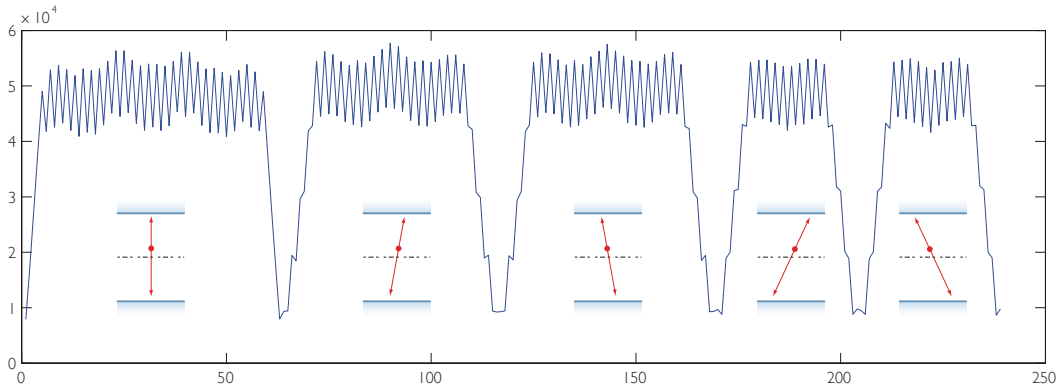

Figure 27 Variation of the total number of counts with 'slice' in the ECAT 3D sinogram format. Each 'slice' is a 2D array, indexed by radial offset and projection angle. Clearly visible are the blocks of sinograms corresponding to the different sets of 3D projections at increasing angles from the axis of the scanner (inset schematics).

It should be noted that this phantom exhibits a very high degree of rotational symmetry, especially relatively high concentration of activity around the outside of the phantom is considered. Note also that the effect of attenuation will be to further emphasise the edge of the phantom.

Given that PET-SORTEO produces sinograms and the above decision to use the count rate as the measure of the amount of data the variation in activity over time is not important. With no other means to allocate the time activity curves with for this phantom they were set to be constants. The values of the activity levels were set such that the highest labels in the phantom corresponded to approximately 34 nCi of ^{18}F – the tracer being important for its positron range, not half-life (which is ignored by the constant activity level). The other regions were linearly mapped to activity levels below 34 nCi. Because of the integer labelling of regions the activity through other regions was mapped to integer values; e.g. 27 nCi.

3.3.1.2 Grid Phantom

This phantom was introduced as counterpoint to the high rotational symmetry of the Modified Shepp Logan, and provides almost ideal raw material for any registration algorithm, having low rotational symmetry and many steep gradients in all directions to

Translation Distance (mm)	Translation Direction (°)	Rotation Angle (°)
0	0	0
0	0	2
0	0	5
0	0	10
2	40	10
7	6	10
19	17	10

Table 2 The positions of the phantoms that were simulated.

aid matching the two positions. It consists of a grid of circles, of varying activity. The activity levels and TACs were set up as for the modified Shepp-Logan phantom.

3.3.1.3 Brain Phantom

A slice through a human brain was used as one pattern for a phantom. Given that rigid body motions are typically encountered in brain studies (because of the skull) this seems a reasonable piece of anatomy to consider – especially with reference to the rotational symmetry point: the brain is approximately ovoid, and the difference between major an minor diameter is a factor in the amount of rotational asymmetry. For the attenuation volume there was a thin shell of bone to represent the skull, with water representing the other material inside the head.

The cross section was taken from a numerical phantom that was installed alongside PET-SORTEO, as described in [11]. A typical set of time activity curves (TACs) was available with the cross section. The physiology being represented is a detailed approximation of an FDG scan, consisting of over 30 different TACs related to specific anatomical structures in the brain. These TACs do, however, present the opportunity to generate

two different images by simulating different times. Here three 3 minute sections have been simulated: 0 to 180 seconds, 180 to 360 seconds and 360 to 540 seconds. This produces three different sets of sinograms, and is an instance of the variation that will be caused by different tracer distributions, even if the underlying geometry is the same, see Figure 26.

3.3.1.4 Simulation

For the gird and modified Shepp-Logan phantoms a 5 minute section of scan was simulated, and 9 minutes total for the 3 variants of the brain phatom. PET-SOTEO simulates an ECAT HR+ (Siemens Healthcare), which produces 3D sinograms. These are stored as a stack of several sets of 2D histograms. The first set are 2D histograms, and are the ones of interest for the present implementation of the registration method. The rest are as Figure 27.

Because of the constant cross section of the phantom it is fair to combine together several of the histograms from the middle of the first set in order to boost the maximum amount of data available without extending the duration of the simulation and, therefore, the time taken. Those near the edges show some edge effect from the limits of phantom's extent and so are not used. Combining different numbers of these 2D histograms together allows the amount of data to be increase without the time required for longer simulation.

The combined set of 2D histograms, complete with scatter and randoms were converted into pseudo-LMF form as per §2.1.5 of Chapter II. Selecting which section of pLMF to use provided a way to introduce variability into the experiments (not as much as simulating the phantom multiple times, but a lot more conveniently).

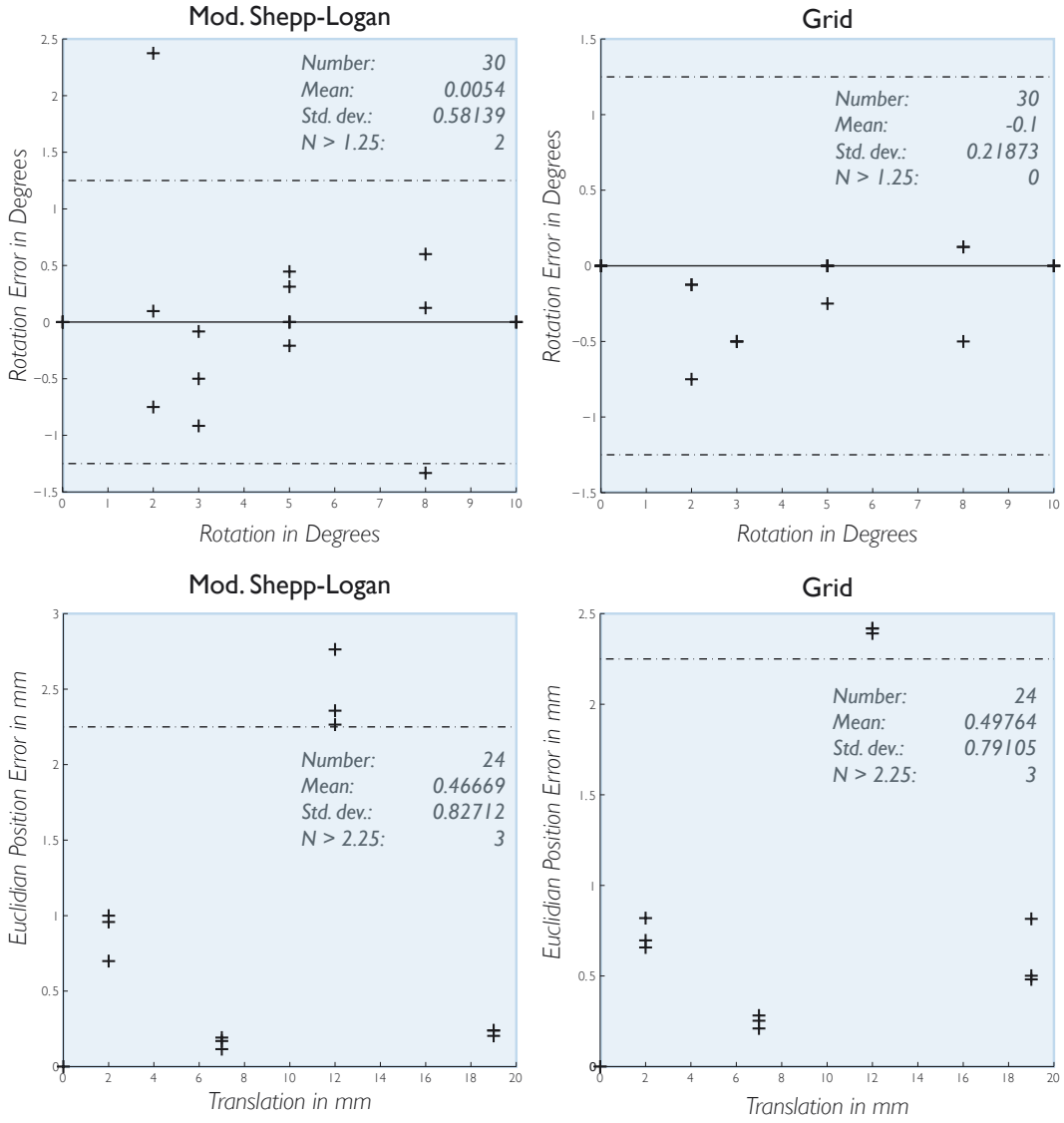

Figure 28 Scatter plots assessing the independent accuracy of Rotation (top row) and Translation (bottom row), for the modified Shepp-Logan phantom (left) and the Grid phantom (righ). Only pure rotations or translations are used. Also shown are the limits of useful accuracy. Also shown are the limits of useful accuracy. Also, inset, are the number of points plotted, their mean, standard deviation, and nuber outside the limits of useful accuracy.

3.3.1.5 Positions and Motions

The SORTEO simulation step is the slowest and the motions are created by drawing sinograms from simulations of the phantom in different positions. Therefore a set of positions was chosen such that various combinations of the positions would produce a suitable range of motions. A total of 7 positions have been simulated, Table 2. From this 49 motions can be made, but half of these are reverse motions, and that number

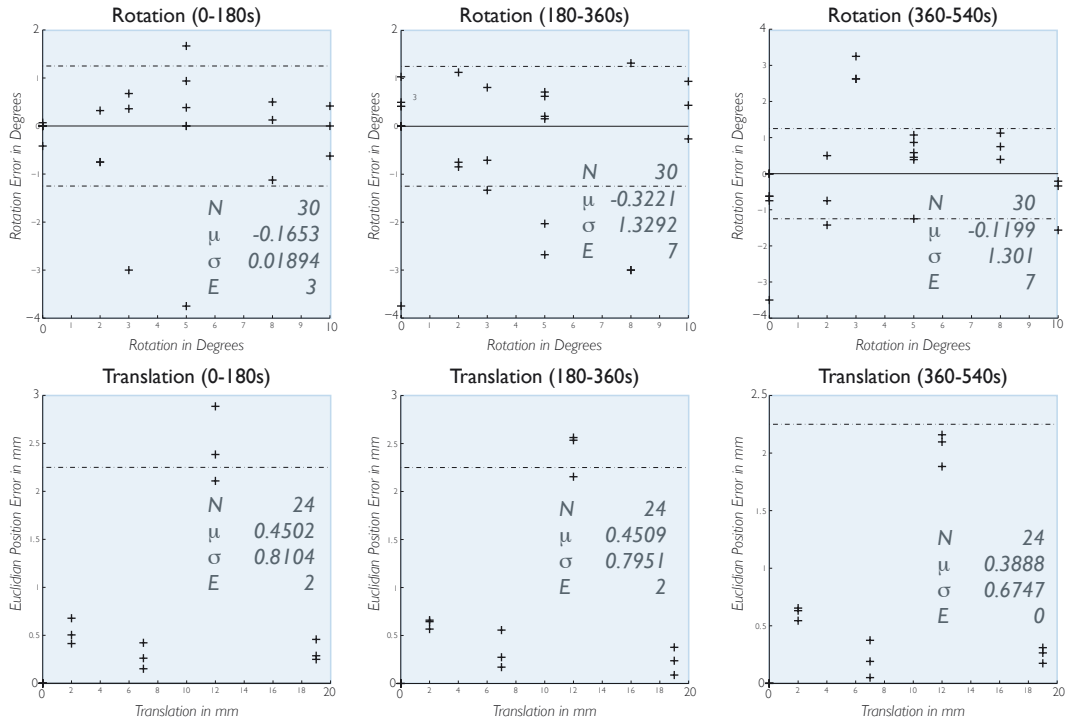

Figure 29 Scatter plots assessing the independent accuracy of Rotation (top row) and Translation (bottom row), for the three different realisations of the brain phantom. Only pure rotations or translations are used. Also shown are the limits of useful accuracy. Also, inset, are the number of points plotted, their mean, standard deviation, and nuber outside the limits of useful accuracy.

also includes 7 'null' cases, where there is no change of position. To keep the number of experiments tractable a subset of half the possible motions were used, chosen to give a good spread over the range of possible motions.

3.3.2 Results

In the following sections various aspects of the performance of the algorithm are considered. For each of the possible combinations of positions multiple sinograms can be generated with varying levels of counts from the pLMF as described above; it is possible to generate very large populations of sinogram pairs to register.

Each sinogram pair is then registered and the results stored. From this large population of registration results subsets can be extracted having the right combination of

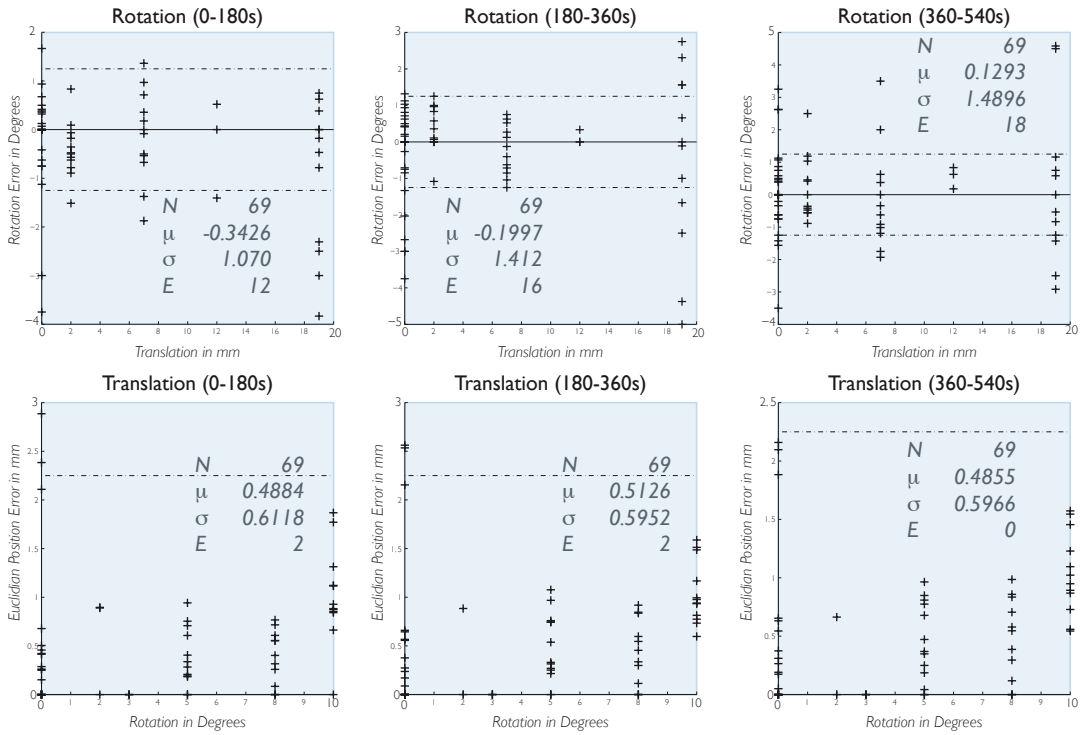

Figure 30 Scatter plots assessing the independence of Rotation (top row) and Translation (bottom row) – error of one plotted against magnitude of the other motion, for the three different realisations of the brain phantom. Also shown are the limits of useful accuracy.

properties, for example number of counts or type of motion (e.g. translation only), for individual experiments.

Measurement of Accuracy: To display the accuracy of the translation algorithm the Euclidean error in position, as equation(17)(17), is used because it is influenced by both the magnitude and direction estimates.

$$E = \sqrt{err_x^2 + err_y^2} \qquad \textbf{(17)}$$

For angles the error in the rotation estimate can be used directly. For reference the angular resolution of the sinograms are 1.25 degrees and, as the present implementation only considers whole row shifts of the sinogram, this is the limit of useful accuracy (finer estimates of the rotation may result from averaging the many individual estimates). Similarly, after reconstruction, the voxel spacing is 2.25mm.

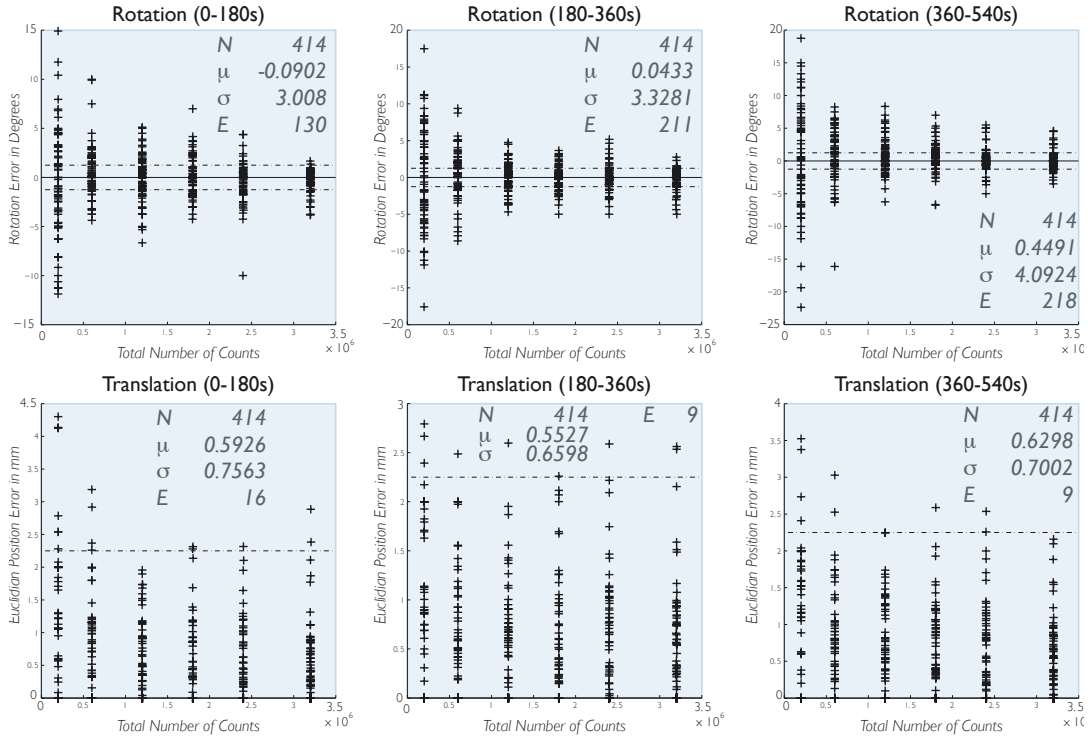

Figure 31 Scatter plots assessing the robustness to low count numbers, for Rotation (top row) and Translation (bottom row) – errors are shown against the total number of counts from both sinograms, for the three different realisations of the brain phantom. Also shown are the limits of useful accuracy.

3.3.2.1 Independent Accuracy

To assess the performance of the rotation and translation steps in the method individually a set of motions were selected such that either rotation or translation were present. The total number of counts (i.e. in both the template and reference sinograms) is on average 3.2 million. The results are presented over the following pages. Those for the Modified Shepp-Logan phantom and the Grid Phantom in Figure 28, and the Brain phantom in Figure 29.

There is less effect from the different phantoms than expected, especially for rotation. This indicates that the overall ovoid shape of the Shepp-Logan phantom provides enough rotational asymmetry to drive the registration; the phantom was, of course, designed to emulate a skull. The three manifestations of the brain phantom have similar patterns

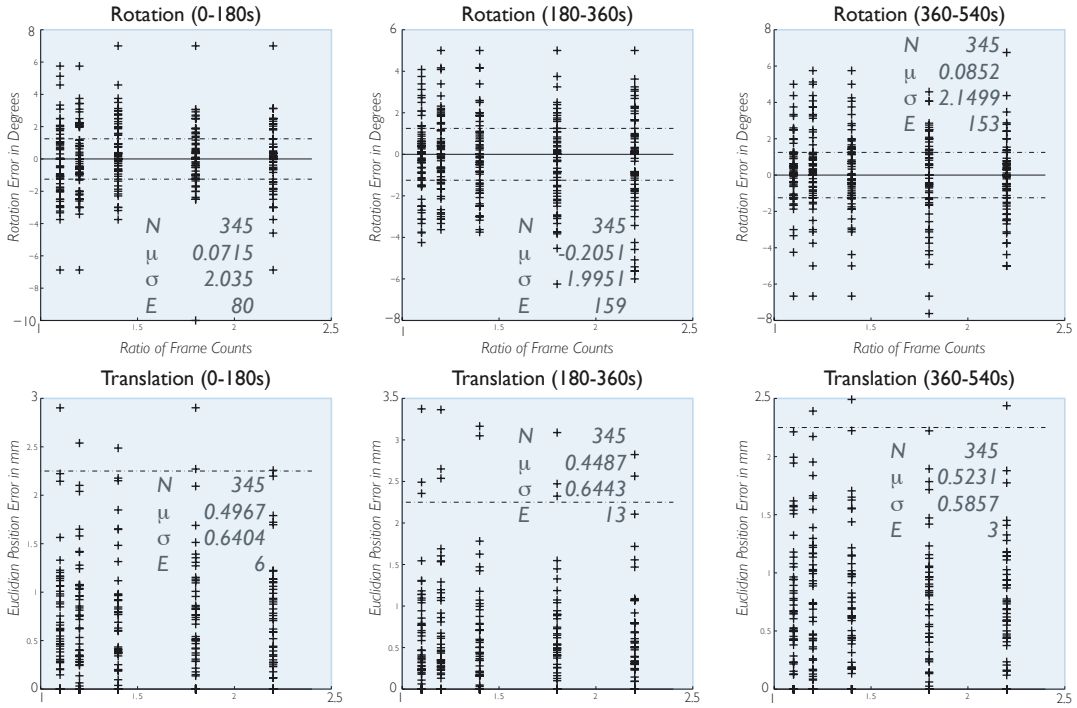

Figure 32 Scatter plots assessing the robustness to frame asymmetry, for Rotation (top row) and Translation (bottom row) – errors are shown against the ratio of the number of counts in the sinograms, for the three different realisations of the brain phantom. (The total number of counts is approximately 3.2 million.) Also shown are the limits of useful accuracy.

to each other and the other two phantoms. As a result of that and as the most realistic phantom, the rest of the experimental work uses these three, rather than attempting to display the results of all.

Overall the results are very good, clustered within the limits of useful accuracy (the angle equivalent to the change in projection angle for each sinogram row and the spatial resolution of the voxels). There is one motion that seems to produce usually high errors for translation (the 12mm case). Especially in light of the fact that the errors seem to be clustered around the voxel spacing this is thought to be an aliasing problem: the current implementation only allows whole bin shifts and therefore there are rounding errors.

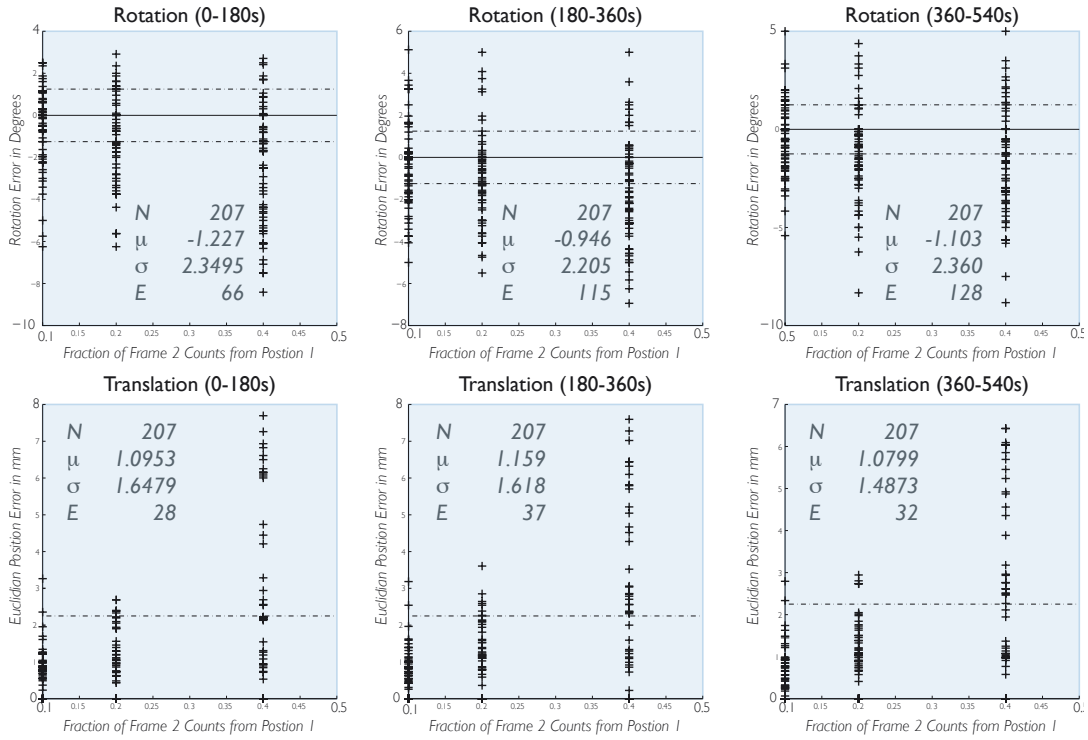

Figure 33 Scatter plots assessing the robustness to framing error, for Rotation (top row) and Translation (bottom row) – errors are shown against the fraction of the counts in the second sinogram that came from the first position, for the three different realisations of the brain phantom. (The total number of counts is approximately 3.2 million.) Also shown are the limits of useful accuracy.

3.3.2.2 Independence of Translation and Rotation

Figure 30 examines the effect that the translations and rotations have on the estimation of the other. The errors in the rotation estimate are compared against the magnitude of the translation that also occurred, and vice versa. Again approximately 3.2 million counts are used, evenly split between the two sinograms.

As there is no significant trend compared with the accuracy of the independent components it is deemed acceptable to use mixed motions henceforth.

3.3.2.3 Robustness to Count Rate

In Figure 31 the accuracy of the rotation and translation estimates are plotted against the total number of counts in the two sinograms, with those counts evenly split between the two sinograms. As expected, with fewer counts the rotation errors are worse, but what is surprising is that the translation algorithm does not seem to degrade much. The difference between the performance of the two can be explained by the fact that with lower counts the rotation algorithm's perception of the rotational asymmetry (and thus its ability to detect the rotation – many of the errors correspond to an erroneous decision that there had not been any motion) is compromised before the detail needed by the translation method – for example the fall off in activity at the edge of the brain.

3.3.2.4 Validity with Asymmetric Frames

To confirm the validity of the method when the frames are asymmetric, i.e. contain significantly different numbers of counts, the ratio of counts was varied and the errors in the estimation of rotation and translation are shown in Figure 32. The total number of counts is approximately 3.2 million. Again the rotation results suggest that sometimes the method is erroneously deciding that there was no rotation.

There is no discernible trend, which indicates that there is no effect from the asymmetry. Both the metrics considered performed near identically at this task too – on page 85 sum of squared differences had been used as an example of where the metric might not be scale invariant. However, note that both should reach be maximised by the correct alignment of the vectors they compare, the question was over whether the non-linear effect of the squaring would cause some element to have more effect than others and therefore produce different maxima when noise is present. That appears not to be the case.

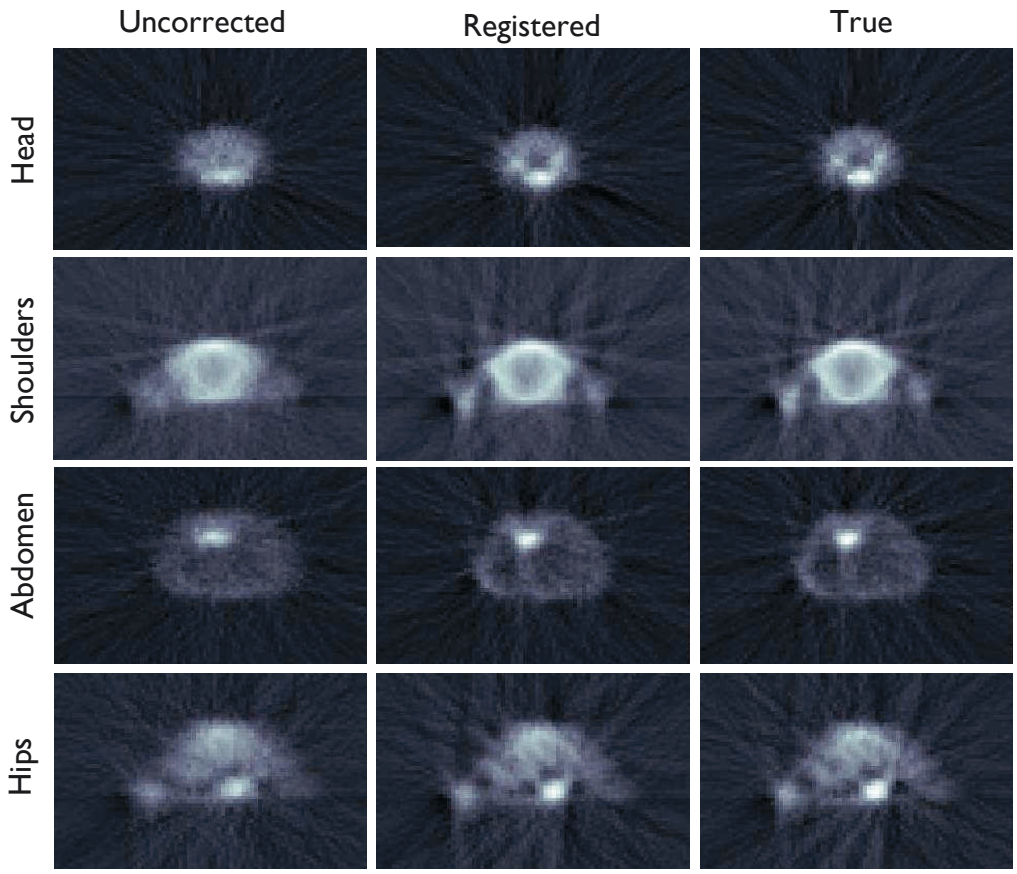

Figure 34 Demonstration of registration on three minutes of data from a mouse, in which there is a small translation half way through. The 'true' column uses a reconstruction of three minuets of data from only the first position.

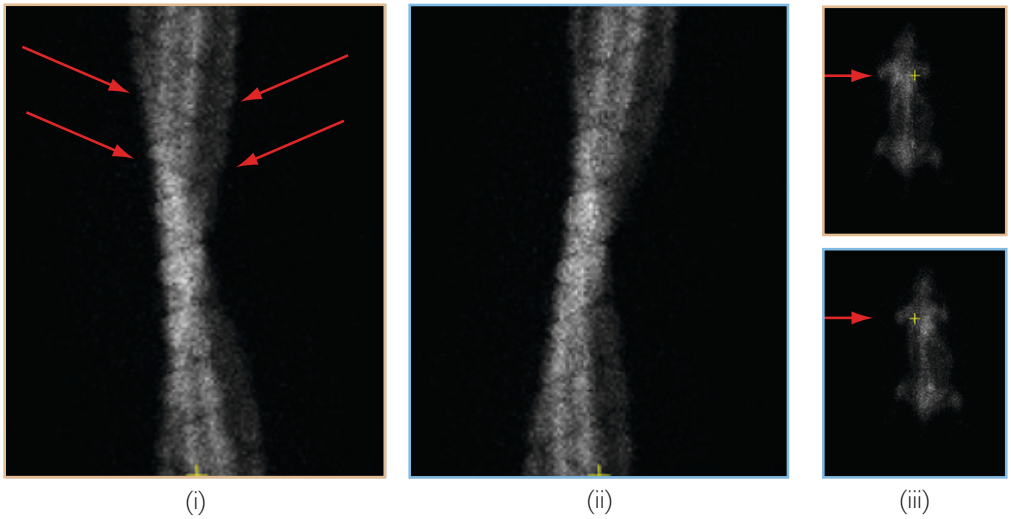

Figure 35 Sample sinograms (i) and (ii) from the two positions. For illustration the coronal projections are shown in (iii), with the top projection corresponding to the sinogram in (i). indicated on (iii) is the cross-section corresponding to the sinograms. Highlighted in (i) is the diagonal pattern from the block gaps. Also present is the effect of scatter, in the counts outside the main core of the sinogram.

3.3.2.5 Framing Error Robustness

To investigate the robustness of the method to inaccurate measurement of when the subject moved, as well as to account for the progressive nature of real motions, two equal frames of simulated data are registered. However in this case some fraction of the second frame's counts are in fact from the first position. Figure 33 summarises the impact that this framing error has on rotation and translational accuracy. Again 3.2 million counts are used, evenly split between the two sinograms, but some allocated to one sinogram come from the simulation of the phantom in the other position.

As would be expected the performance of the registration deteriorates with framing error. Again some of the errors are explained by the algorithms deciding that there was no motion, which is of course more likely with this sort of framing error.

3.4 Qualitative assessment

In addition to the above investigation of the performance it was felt important to demonstrate the method on real data early. To this end an early set of preclinical data was used. This data uses a mouse, which was administered a large dose of FDG and then euthanized (after about an hour to allow the tracer distribution to stabilise). At the start of the scan the total activity in the scanner was measured as 21.4 MBq. This is quite high, but for the first study it seemed sensible to gather as many counts as possible (after all, it is possible to histogram a shorter section to get lower numbers of counts). The animal was strapped to a small pallet that allowed it to be moved over the bed in the scanner during the scan. This means that the motions are not very accurate: their size is not known beyond being 'about 5mm' and there is no guarantee that they are entirely in plane.

The data was histogrammed by the commercial reconstruction software and the sinograms were exported so that they could be transferred into Matlab, in which the

registration was implemented. As discussed, as attenuation correction and scatter correction potentially introduce artefacts, motion estimation and correction should be done using uncorrected data.

Figure 34 presents the registration of several cross-sections of the mouse. 120 seconds of data were histogrammed from each position. The images are reconstructed by using the filtered back projection method. The registration used the sum of squared differences metric for translation and the correlation metric for rotation, illustrating that the metrics need not be the same.

Although rudimentary compared to later work with real data this demonstration served its purpose: there is a clear improvement in the image with reregistration. Indeed, there are details visible in the registered and true column that are completely missing, not just obfuscated, in the uncorrected images.

The demonstration also showed up an interesting phenomenon on the sinogram. The gaps between blocks of detectors in the Inveon mean that the are empty sinogram bins: the 'wire mesh' pattern in Figure 35. Any motion takes place behind this unmoving grid. of empty bins. Fortunately, the metrics seem to have coped well enough with this effect in these experiments.

There is a further demonstration of the registration in the conclusions to this thesis, in the context of an integrated demonstration of twitch detection and correction using all the methods presented this thesis. That example makes use of the software written for late chapters to allow the histogrammed list mode data to be imported directly into Matlab.

4. Conclusions

The adaptation of the method of Fitchard et al. has been investigated for use in PET. The method runs on the order of a few tens of seconds; this speed is because it is non-itereative. The results presented demonstrate the potential for pre-reconstruction registration, and illustrate the point that the division between image domain and sinogram domain is not absolute. They also highlight the fact that rotation is harder to identify and correct for than translation – with the brain there are fewer rotational details to rely on than for translation: the head is reasonably symmetrical under rotation but there are distinct edges to the activity on each projection, which drives the translation algorithm.

The two metrics compared have produced very similar results. This is not too surprising given their similar geometric interpretations: if the two vectors begin compared are considered to be points in a multidimensional space then the sum of squared distances is the Euclidean distance between them and the correlation is the inner product – the product of the magnitude of one vector with that of the projection of the other onto the first.

The choice of metric is the most obvious avenue for continued development. For example, sinogram data is well modelled by the Poisson distribution. Therefore, the Poisson log joint likelihood, of observing all the pairs of elements in the vectors would form the basis of a theoretically grounded maximum likelihood metric. This will be discussed at length in the next chapter as it forms the basis of the twitch detection method introduced there. Because it is the intrinsic measure of the consistency of PET data in the sinogram domain it should perform well in this role too. *Unfortunately, the such a metric would only be relevant to the translation case, the Fourier transform of a vector of Poisson variables does not produce a vector of Poisson variables, because a weighted sum (inherent in the Fourier transform) of Poisson distributed variables is only Poisson distributed if all the weights are one.*

Future experimental work should include investigation of a wider range of metrics, and more importantly a wider range of phantoms (both in terms of geometry and the tracer time activity curves). At present the results have not been systematically optimised: if the method is to be used in a specific scenario – e.g. clinical or pre-clinical imaging, for specific tracers – then the metric can be optimised to perform best in that setting. At that point it will be important to compare the method against state-of-the-art algorithms.

From the experimental work done so far it seems that the 'validity checks' sometimes rule out motions erroneously. It is also felt that the current means of selecting the cut-off of stable rotation estimates is too conservative: at times a small perturbation in the middle of the stable region (which would be suppressed by the averaging) is enough to cause the end of the stable region to be marked.

Further on the multiple estimates of rotation and the relationship between feature scale and frequency: [64] discusses the relationship of Wavelet transforms and the Radon transform, specifically how a transform of the projections can be related to an equivalent transform in the image domain. This logic could be used to make better conditioned registrations, by using wavelet techniques to focus the algorithm on information corresponding to structures of interest.

Another option for further development would be to implement a 'sub-pixel' registration: incorporate an interpolation of the sinogram data to allow finer motions to be estimated and corrected. The caveat is that any interpolation makes implicit assumptions about the data and these can bias the results: for example a linear interpolation of a quadratic curve will always underestimate the magnitude of points. It is desirable to remain true to the intrinsic characteristics of the data, especially in PET where there is a strong statistical model and the promise of quantitative imaging. Thus careless interpolation should be avoided – especially before reconstruction, as modern, iterative reconstruction algorithms rely on the statistical properties of PET data.

The last obvious area for further development is the extension of the implementation to 3D. Of interest is [84], which focuses on the corrections needed to compensate for the uneven exposure of sinogram bins caused by translations along the axis of the scanner and rotations perpendicular to that axis.

> *The intention had been to develop this method further, including to 3D, and then to investigate the possibility of estimating deformable motions from the sinogram domain data, however it was realised that the salient issue in correcting for twitches is detecting the time of the twitch. The solution developed is presented in the following chapters.*

Chapter IV: Twitch Detection – Basic Method

As mentioned earlier, registration provides the means to correct for motions but the need for a practical means of identifying when motions occur has been comparatively ignored. What is developed in the following chapter is that: a way of detecting discontinuities in the scan data that does not require any additional equipment or steps in the imaging protocol.

1. Introduction

1.1 Motivation

Recall that within the wider set of types of motions, one can select subsets of motion based on certain characteristics. Motions caused by breathing or cardiac cycle, for instance, are continuous and, to some degree, periodic. Further, the motions can be linked to another easily observable process: in the case of the cardiac cycle it is the ECG trace, for breathing the motion of the chest wall can be measured and used as a proxy for a phase of motion cycle. Such periodic and easily, externally, observable motions have already received much attention in PET imaging, and are tackled using gating.

As noted in §3.1.1 in Chapter I a distinct class of motions are the abrupt transition changes position perhaps best called ***twitches***. These can be thought of as distinct and specific events during which the subject changes from one comparatively stable pose to another, also comparatively stable, pose. In practice, twitches occur quite frequently during imaging studies: often as a result of a neurological condition that is the primary motivation for the study, or as random muscle spasms (quite common as anaesthetics wear off), or many people twitch while drifting off into sleep, etc.

The impact of the anaesthetics on the physiology of interest is also a concern in pre-clinical practice and so there is a desire to perform imaging with awake animals (for example, see [88]), or at least minimise the sedation – which would make twitches more likely. For FDG several anaesthetics are investigated [85], with PET images and autoradiography results compared in [49]; both show that there is a variation between different anaesthetics. Moreover the effect of the anaesthetics was found to be minimised by allowing the tracer distribution to reach steady state before administering the anaesthetic – for the general case of dynamic studies this is not possible; it is the evolution of the tracer distribution that is of interest.

Twitches are, by their nature, unpredictable and result in sections of the scan that do not match the configuration of the subject used for the attenuation correction scan. It seems wasteful – in terms of time, cost and patient stress – to perform a scan (potentially several hours for a dynamic study) just to discard the data because of a very short motion event during the scan.

Indeed, given that between twitches the subject remains in a relatively stable position, it is not unreasonable to suppose that if the scan were to be split up into sections before and after the times when the twitches occurred it would be possible to estimate the motion and then correct for it. As discussed in the previous chapter, registration is a fairly well established field within medical imaging. The objective, therefore, is to devise a practical, robust and accurate method of identifying an abrupt motion.

1.2 Existing Solutions

Existing solutions are essentially based on observing the subject during the scan. At its most basic, this involves a technician watching the subject for the duration of the scan and manually noting the times at which twitches occur. This has the obvious limitation of requiring the close attention, over a long period of time, of the technician. Not only could the technician's time be put to better use, but also, it is quite likely that the observer will miss twitches – they are rare and the scans can take a long time, boredom and distraction will set in. Termed 'vigilance', performance in such tasks has been studied in industrial and military research, and continued monotonous observation tasks leads to poor performance, especially as the duration of the task increases, see [89]. Further, such tasks place stress on those expected to perform them [90].

To improve on this, several schemes have been suggested along the lines of automated observation. In most cases some form of video camera rig is installed to observe the subject of a scan and track motions. Some of these systems require specific targets to

be affixed to the subject, as in [27], [28], [29], [35], [69], and [82]. In these cases the motion detection system also provides the means to estimate the motion, and thus their focus has been on how to use that information to transform the data. The method in [78] is different in that it does not require a marker – the camera systems track the surface of the patient. Conversely, in [56] the target tracked is a small point source of radiation in the scanner's field of view and so no additional cameras are needed.

There are a number of drawbacks to these approaches. The inconvenience involved in setting up such equipment, particularly if it involves a new target for the tracking that needs to be affixed to a patient, means that it reduces the throughput of the scanner. Each scan takes a certain amount of time, including the data acquisition and also the setup and the take-down time. Increasing the setup time decreases the number of scans that can be done each day — therefore the scanner is not being used to its full potential. Also a serious concern is the stress on the patient caused by methods that require either more time or more hassle. Although motion tracking methods potentially have the added benefit of providing additional information about the motions that do occur, they also face problems with reliability.

A method that inferred the times of twitches directly from the PET data would be far more convenient and better able to distinguish changes that are relevant to the PET imaging than a method based on infra-red of visible light imaging of the outside of the subject.

A means of gating from list mode data is presented in [9]. This is interesting in the context of this project because it too detects motions from list mode data without any reconstruction. It infers a gating signal from the rate varying rate of coincidences caused by the uneven sensitivity of scanner across the field of view and from the 'centre of mass' of counts along the axis of the scanned. It is driven by the movement of the heart (which has a high concentration of FDG) caused by breathing.

1.3 Method Overview

At the core of the idea is the fact that the bin counts on a sinogram are random variables which follow a Poisson distribution. If there is no change in the underlying tracer distribution (i.e. if there is no motion at that time) then the Poisson parameters for all the bins should be the same before and after that time. Therefore, by computing the joint likelihood of observing two sections of data (one before and one after a given time), under the null hypothesis that there is a common set of rate parameters, one can estimate the likelihood of observing the data of both sections. Repeating this calculation at regular intervals through the duration of the scan one can look for times with comparatively low likelihoods which, given the null hypothesis, implies that there was some abrupt change around that time. A key point is that this is a statistical measure of a property of the data — firstly, this means that the statistical properties must be maintained and, therefore, this relies on performing the twitch detection on data that has not been corrected for scatter, randoms, or attenuation (or, worst of all, iteratively reconstructed), and secondly, it is to a large degree an intrinsic property of the observed data (the method is comparatively free of a priori choices that influence the results).

Using several techniques discussed later in this chapter and the next, this method has the potential to be implemented very quickly — not least the fact that it is performed on the raw data without reconstruction significantly reduces the computational cost. Initial implementations have, indeed, exceeded real-time performance on comparatively low-end hardware and, as such, it is entirely reasonable to propose that this analysis be performed online (during data acquisition), which would open up a range of possibilities such as acquiring data for as long as it takes to achieve sufficient twitch-free data.

2. Method

2.1 Basic Method

In order to describe the method in sufficient detail, the following will describe the calculation of the likelihood under the null hypothesis (no twitch) at some given time t. For the purposes of this description, the first step is to form two sinograms. As described in §2.6.1 of Chapter I, this requires converting the events recorded by the scanner hardware (which are indexed in terms of the crystals within the scanner) to a histogram in the projective space. For the purposes of this description, the two-dimensional formulation will be used, although as will become obvious, the only changes that the 3-D formulation requires is an increase in the number of indices used to form the sinograms. However, since the method does not use the indices of the bins other than for iterating through them, the difference between 2-D and 3-D is completely irrelevant.

The one parameter of the method is a time scale and this is used to fix the sections of data used to create the two sinograms — one sinogram contains data from before the specified time t, the other from after — the length of time before and after being determined by the time scale parameter.

Conceptually, this step is quite simple, but computationally it is quite involved because the histogramming requires quite a lot of memory access and therefore runs quite slowly. The histogram is stored in memory as some form of array and there are typically hundreds of thousands of bins. As there is no way to predict which bin the next event will fall into, there is no way to efficiently cache the sinogram – the histogramming is constricted by the so-called von Neuman bottleneck: the computation is delayed by the time needed to read and write to a main memory, which nowadays runs much more slowly than the CPU.

As will be discussed later, the implementation can be modified to enable much faster processing; but, for now, it is easier to think of twitch detection for a specific point in time, with histograms of the data before and after that time being readily available, even though this would entail re-histogramming the two windows for every point at which the likelihood value is to be computed.

The number of counts in each bin on each of the sinograms follows a Poisson distribution. The number of events counted in a bin follows from the decay of the tracer along the corresponding line of response and so the Poisson statistics are a direct consequence of the radioactive decay of the tracer (which is one of the best examples of a Poisson process). Each Poisson random variable has an associated mean, μ. *The term Poisson Parameter is also used, interchangeably, with the term 'mean'.* Given that parameter, one can thus find the probability of observing the number of counts in that bin as follows. In which s_b represents the number of counts observed in the chosen bin from the 'before' sinogram, μ_b the corresponding Poisson parameter, and s_a, μ_a the same for the 'after' sinogram.

$$\begin{aligned} \mathrm{P}\left(s_b \;\&\; s_a\right) &= \mathcal{P}\left(s_b \mid \mu_b\right) \times \mathcal{P}\left(s_a \mid \mu_a\right) \\ &= \frac{\mu^{s_b} e^{-\mu_b}}{s_b!} \times \frac{\mu^{s_a} e^{-\mu_a}}{s_a!} \end{aligned} \tag{18}$$

If there is no change in the underlying tracer distribution (i.e. if there is no motion at that time) then the bin's Poisson parameters will be the same in both the before and after windows. So with $\mu_a = \mu_b = \mu$, (18) becomes:

$$\mathrm{P}\left(s_b \;\&\; s_a\right) = \frac{\mu^{s_b+s_a}\; e^{-2\mu}}{s_b!\; s_a!} \tag{19}$$

It follows that the overall joint likelihood of observing all the data in both the before and the after windows is the product of the probabilities for each bin within each window. (In equation (20) the product is over ι, which is some linear index of the sinogram bins,

so that the expression is same for sinograms in the 2D and 3D cases. The upper case S_b and S_a represent the whole 'before' and 'after' sinograms.)

$$\mathrm{P}\left(S_b \;\&\; S_a\right) = \prod_{\iota} \left(\frac{\mu[\iota]^{\left(S_b[\iota]+S_a[\iota]\right)}\; e^{-2\,\mu[\iota]}}{S_b[\iota]!\; S_a[\iota]!} \right) \qquad \textbf{(20)}$$

Note that there is an assumption made here, one which is commonly made in the field, that each bin is independent of the others.

In the case of empty bins: as the factorial of zero is one, the denominator term is still valid, but if both bins are zero then μ will also be zero and this means that the whole expression for that value of ι is zero – and, thanks to the product, one such bin forces the whole likelihood to zero. Therefore bins with no counts in either window must be ignored. *There is a further discussion on this theme later, in §*2.3.2.1*, once more of the method is explained at the conceptual level.*

With this calculation repeated at different times over the duration of the scan, a 'likelihood plot' (joint likelihood plotted against time) can be used to show the variation over the course of the scan. Near times when the underlying pattern of emissions does change, e.g. because of a motion, the pattern of the counts recorded in the sinograms will become different from each other, causing the log-joint- likelihood to fall. The scan could then be split into sections based on these local minima.

2.1.1 Finding μ

The key step in being able to calculate the probabilities is choosing the Poisson parameter μ. The null hypothesis (that there was no motion, and therefore a very similar distribution of the tracer) implies that there should be a common μ for each bin. For each of the bins on the histogram there are two measurements — one from the before

and one from the after windows. Therefore, μ is chosen by selecting the value that makes observing both measurements most likely.

Formally, for any specific bin, μ is chosen to be the maximizer of the joint likelihood of observing the counts in that bin in the before window and the after window.

$$\mu = \arg\max_{\mu} \left(\frac{\mu^{s_b+s_a}\ e^{-2\mu}}{s_b!\ s_a!} \right) \tag{21}$$

Conveniently the maximizer happens to be the mean of the observed counts in the before and after data. This can be shown in the usual way, taking the derivative and identifying the zero crossing point. (The sign of the second derivative at this point will confirm that this is a maximum.)

$$\begin{aligned} 0 &= \frac{\mathrm{d}}{\mathrm{d}\mu} \left(\frac{\mu^{s_b+s_a}\ e^{-2\mu}}{s_b!\ s_a!} \right) \\ &= e^{-2\mu} \left((s_b + s_a)\, \mu^{(s_b+s_a-1)} - 2\mu^{(s_b+s_a)} \right) \\ &\quad \ldots \text{ignoring } \mu = \infty \text{ solution} \ldots \\ &= (s_b + s_a)\, \mu^{(s_b+s_a-1)} - 2\mu^{(s_b+s_a)} \\ \mu &= \frac{s_b + s_a}{2} \end{aligned} \tag{22}$$

2.1.2 Log Likelihood

Real sinograms, even in the 2D case, have many thousands of bins. This implies many thousands of multiplications. Given that the individual probabilities will be much less than unity the result of all these multiplications risks reaching the limits of numerical computation.

2.1.2.1 The Numerical Problem

For example, consider the number of bins with 2 counts in each of the windows required to fall below what is possible to compute. The bin-wise joint likelihood in this scenario is:

$$\begin{aligned} \mathrm{P}\left(s_b = 2 \;\&\; s_a = 2\right) &= \frac{2^4\, e^{-4}}{2!\; 2!} \\ &= \frac{16 \times 0.01831563\ldots}{4} \\ &= 0.0732625\ldots \end{aligned} \tag{23}$$

This means that the sinogram-wise joint likelihood, for a sinogram of N such bins, is:

$$\begin{aligned} \mathrm{P}\left(S_b \;\&\; S_a\right) &= \prod_{\iota=0}^{N-1} \left(\frac{16\; e^{-4}}{4}\right) = \left(0.0732625\ldots\right)^N \\ &= 4^N\; e^{-4N} \end{aligned} \tag{24}$$

Floating point arithmetic is a notoriously complicated problem, but the two key concepts are ***precision*** and ***accuracy***. Accuracy refers to the ability to reliably record a value that is close to some 'true' value. Precision, on the other hand, is the amount of information captured. Integer arithmetic, to provide the counter example, is completely accurate (having very simple rules about what the 'true' results are) but lacks precision, being unable to capture any non-integer information in the result.

The floating point representations of real numbers (see Figure 36) have much higher precision. However the need to round off at some point limits the accuracy: it is not possible to construct rules that guarantee that performing an inverse calculation will return the original numbers. Indeed, the rules of floating point arithmetic, which cover such topics as how exactly to represent the numbers, when and how to round off etc., are much more complicated than for integers.

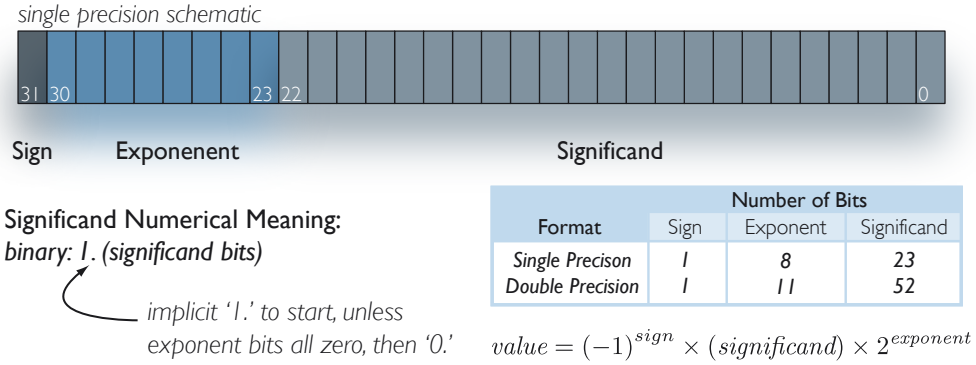

Figure 36 Summary of IEEE 754-1985 floating point representation. Note that in the expression defining the value of the number the ***sign***, ***significand*** and the ***exponent*** all refer to the numerical meaning of the bits stored in the corresponding parts of the overall floating point number. The exponent is stored as an ***offset binary number***, meaning that the first half of the range is given over to negative numbers counting up to zero, then up to the maximum positive number (-126 to 127 for single precision, with exponent bits set to hex 00 and FF reserved for special purposes). N.B. the significant is used to represent a fractional value starting with binary '1.' unless the exponent bits are all zero – in such cases the number is called ***sub-normal*** and the value is computed with the significand starting '0.' and the exponent with decimal value -126 (in the single precision case).

In the case of IEEE 754 single precision floating point numbers, the significand is able to express binary numbers to the equivalent of 7.22 decimal digits. For double precision, the significand's precision is equivalent to 15.95 decimal digits.

These digits can be used to express numbers far smaller than one by using a negative exponent to scale them down – by 2^{-126} for single precision (this scaling adds, in decimal terms, the equivalent of 38.23 decimal places) and 2^{-1022} for double precision (adding the equivalent of 307.95 decimal places).

Therefore the smallest number that can be represented is on the order of 45 decimal places for single precision floats and 324 decimal places for double precision floats.

There is a trade off between precision and range. For non-zero exponents the minimum increment of the significand represents an increasingly large jump in value.

Being generous and rounding the bin-wise joint likelihood up to 0.1, this means the numerical precision will be exhausted for $N=46$ in single precision, or $N=325$ in double

precision. Taking an easy example, one 2D slice of a sinogram with 128 projection angles and 128 bins per angle implies N=16,384, well in excess of the maximum available.

2.1.2.2 Solution Using Logarithms

As is well known the joint likelihood can be approximated by taking logarithms, thus converting the multiplications into a sum. This trick is used, for instance, in statistical reconstruction algorithms. Crucially, as the logarithm converts the product into a sum the exponential approach to zero does not occur.

$$\begin{aligned}\log\left(\mathrm{P}\left(S_b \;\&\; S_a\right)\right) &= \log\left(\prod_{\iota}\left(\frac{\mu[\iota]^{\left(S_b[\iota]+S_a[\iota]\right)}\; e^{-2\,\mu[\iota]}}{S_b[\iota]!\; S_a[\iota]!}\right)\right) \\ &= \sum_{\iota}\left(\log\left(\frac{\mu[\iota]^{\left(S_b[\iota]+S_a[\iota]\right)}\; e^{-2\,\mu[\iota]}}{S_b[\iota]!\; S_a[\iota]!}\right)\right) \\ &= \sum_{\iota}\left(\left(S_b[\iota]+S_a[\iota]\right)\log\left(\mu[\iota]\right) - 2\,\mu[\iota]\right. \\ &\qquad \left. - \log\left(S_b[\iota]!\right) - \log\left(S_a[\iota]!\right)\right)\end{aligned} \tag{25}$$

As the logarithm is monotonic, there is a strict relationship between the variation of the true joint likelihood with that of the log joint likelihood — if the former rises (or falls) then so must the latter. Therefore, the intuition that a motion would cause a fall in the joint likelihood carries across and the log joint likelihood would also exhibit a drop in value around the time of a motion.

2.1.3 Definitions

Henceforth, for simplicity, the term ***likelihood plot*** will be used to refer the log joint likelihood plotted against time. Also, in terms of notation, let the log joint likelihood for one bin be:

$$\mathrm{L}_{W^*}\left(t\right) = \sum_{\iota}\left(l_{W^*[\iota]}(t)\right) \tag{26}$$

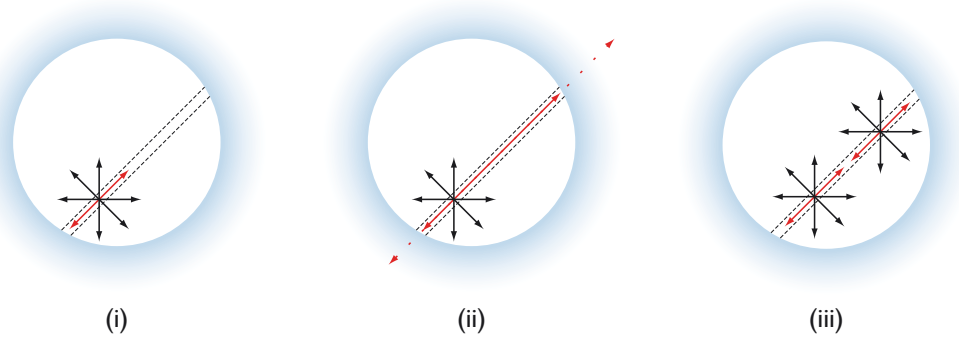

Figure 37 Effects of geometrical and detector efficiencies on the Poisson statistics of the decay process. (i) Only some fraction of the decays produce photon pairs in the right direction. (ii) Some fraction of those do not interact with the crystals and thus are not detected. (iii) Photon pairs from decays at multiple positions get summed together.

where the window width over which the sinograms are histogrammed is determined by W^*. Further, from (25):

$$\begin{aligned} \mathrm{l}_{W^*[\iota]}(t) = (S_b[\iota] + S_a[\iota]) \log(\mu[\iota]) - 2\,\mu[\iota] \\ - \log(S_b[\iota]!) - \log(S_a[\iota]!) \end{aligned} \tag{27}$$

2.1.4 Validity of the Poisson model

The validity the method depends on the suitability of the Poisson model. Attenuation and scatter correction cannot be applied to the raw data because both rely on the attenuation correction CT scan and its relationship to the emission data, which is potentially erroneous.

The tracer decay and positron production events are known to be Poisson in nature. Some fraction of the pairs of photons, produced by the positron annihilations, that travel in a given range of directions are counted in the corresponding bin of the sinogram. The thinning of the emissions by the geometrical factors and the detector efficiency does not change the nature of the distribution; rather it just scales the parameter, so the parameters of the variables representing the sinogram bin counts are the rates of decays producing emissions in the relevant directions (and 'predestined' to be detected). There is also a summation along the line of response corresponding to the sinogram

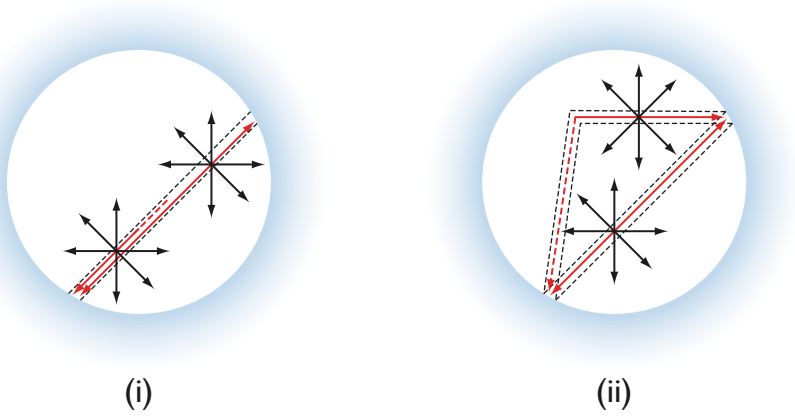

Figure 38 Effects of attenuation and scatter on the Poisson statistics of the decay process. (i) Attenuation is very much like the detector efficiency: only some of the decays produce two photons that will reach the detectors to be detected as a coincidence. (ii) Scatter, in effect, enables new routes that join the two detectors together, so the observed events are the result of some (weighted) summation over all possible paths.

element, so the bin count's parameter is the rate of emissions (in the right direction) from any part of that line of response. See Figure 37.

– Scatter and Attenuation

Events that get scattered or attenuated still conform to the Poisson model: attenuated photons do not reach the detectors, just as only some decay events cause photons in the right direction; scatter just provides alternate routes to the detector pair from various parts of the object, just as all the points along the line of response contribute to the counts in the bin. For these reasons neither phenomenon disrupts the proposed method, and it is deemed better to use uncorrected data so that the Poisson model fits the data as well as possible, rather than risking unknown effects from the correction. See Figure 38.

– Random Coincidences

Random coincidences do not necessarily follow the model in the same way that attenuation and scatter do. However, as randoms correction is normally done by subtracting a delayed window of data, the correction certainly does not fit the Poisson model. Therefore, no correction is applied to the data for twitch detection.

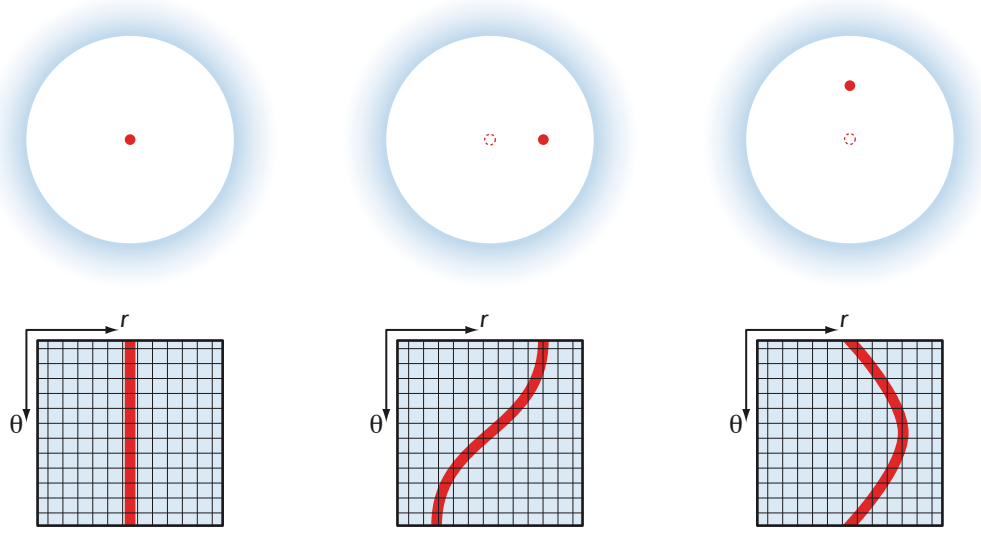

Figure 39 Schematic showing the sinograms corresponding to a point source in three different positions, and, therefore, different bins (and different numbers of bins) on the sinograms. Further the number of counts falling into the bins changes – consider the fraction of each grid square that is covered by the sinusoid.

2.1.5 Appearance of Motions on the Likelihood Plot

The effect of the two windows rolling past an idealised twitch (i.e. instantaneous change of position) is the 'V' shaped minimum as would be expected from the assumption of a constant emission pattern not being valid. However, the baseline likelihood in the two poses may not be the same: the re-distribution of counts between different sinogram bins that occurs as a result of the twitch can mean the 'baseline' joint likelihood is different thereafter.

Consider a point source at the centre of the scanner's field of view. All (true) counts will be recorded in one column (radial offset co-ordinate) of the sinogram. Once the point moves off centre the counts will be observed in many columns (although which one depends on the sinogram row — projection angle). The result is that the same number of counts is spread over many bins and so the Poisson distribution for each bin is different, as Figure 39 illustrates.

It follows that the 'V' shape from the two windows rolling past a change is confounded with a progressive change in the baseline level, centred on the twitch time, as more of the new data with its new baseline level is considered by the windows. The relative magnitudes of the baseline change and the minimum control the overall behaviour, ranging from a pure 'V' to an (almost) perfect slope, with the minimum becoming progressively more asymmetric between the two. Midway between the extremes, the minimum disappears: one side of 'V' is completely cancelled by the change in baseline level, this results in a two part slope between the two levels (one part is where one side of the 'V' reduces the slope from the baseline change; the other reinforces the other part). However, the known width of the windows means that there is some prior knowledge of the shape of the 'minimum' that will prove useful in identifying the features of the likelihood plot corresponding to motions. See Figure 40, which shows the range of features typically encountered during the development of the method with simulated data.

2.2 Decay Correction

One of the factors that exhibits a strong link to the joint log likelihood computed is the average number of counts in the bins. The overall decay of the tracer during the scan means that the rate of emissions, and therefore observed counts, falls as the scan progresses. This means that, for a fixed window width, the expected number of counts in the windows will fall as the scan progresses. This results in a systematic drift of the likelihood upwards with time.

To correct this requires ensuring that there is a constant expected number of counts in the before and after sinograms. This can be achieved by adjusting the widths of the windows based on the time within the scan. Specifically, by lengthening the windows, the number of counts will increase and thus compensate for the falling count rate. If the width of the window is chosen correctly, the net result will be a constant expected number of counts.

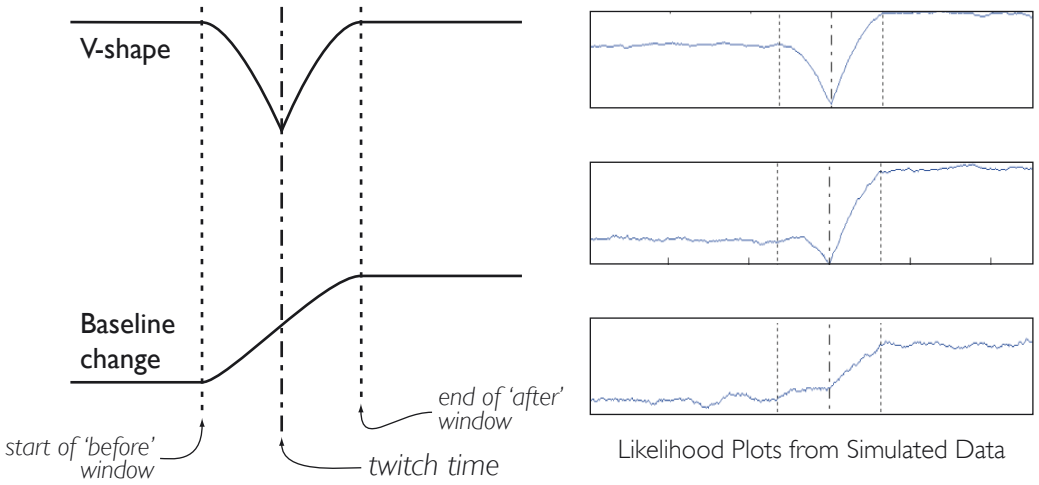

Figure 40 The appearance of twitches on the likelihood plot depends on the mixture of the 'V' shaped minimum from the violation of the null hypothesis of a common set of Poisson parameters and the progressive baseline change as the likelihood adjusts to the new position of the subject in the scanner. By using simulated data to generate the examples on the right the change of position can be made instantaneous – all the variation in shape is due to the different mixes of the two basic effect, determined by the positions before and after the twitch.

For tracers with short half-lives (short relative to the window width) there will also be a need to correct the decay between the before and after windows. The strategy for this correction is exactly the same, the only difference being that the widths of the 'before' and 'after' windows are both chosen adaptivity rather than picking a single average width for both windows. Given that it is simple, it makes sense to do both corrections anyway.

Radioactive decay means that the count rate will fall exponentially, so it is easy to choose the window widths for some given time. First consider the activity at time t in terms of N_0 the inital activity (at $t=0$) and the isotope's half life (λ is the reciprocal of the half life).

$$\mathrm{N}(t) = N_0 e^{-\lambda t} \tag{28}$$

Thus if the window widths are set by some function of the time the expected number of counts in each of the windows can be found thus:

$$\begin{aligned} \mathrm{C_b}(t) &= \int_{t-\mathrm{w_b}(t)}^{t} \mathrm{N}(\tau)\,\mathrm{d}\,\tau \\ &= \frac{N_0}{\lambda}\left(e^{-\lambda(t-\mathrm{w_b}(t))} - e^{-\lambda t}\right) \end{aligned} \tag{29}$$

$$\begin{aligned} \mathrm{C_a}(t) &= \int_t^{t+\mathrm{w_a}(t)} \mathrm{N}(\tau)\,\mathrm{d}\,\tau \\ &= \frac{N_0}{\lambda}\left(e^{-\lambda t} - e^{-\lambda(t+\mathrm{w_a}(t))}\right) \end{aligned} \tag{30}$$

By setting a target number of counts and rearranging (29) and (30), the following equations gives the width of the windows required to achieve that number of counts.

$$\mathrm{w_b}(t) = \frac{1}{\lambda}\ln\left(\frac{\lambda C_*}{N_0}e^{\lambda t} + 1\right) \tag{31}$$

$$\mathrm{w_a}(t) = \frac{-1}{\lambda}\ln\left(1 - \frac{\lambda C_*}{N_0}e^{\lambda t}\right) \tag{32}$$

For convenience, one can specify the target number of counts in each window in terms of a duration and the initial count rate. That is to say, instead of choosing N counts per window one can specify a hypothetical window width in seconds which corresponds to having windows of that width, assuming that the count rate were to be constant, matching the average count rate over the duration of the scan.

The average count rate can be deduced from (28), provided the initial count rate and the duration of the scan are known:

$$\begin{aligned} N_* &= \frac{1}{T}\int_0^T N_0 e^{-\lambda\tau}\,\mathrm{d}\,\tau \\ &= \frac{N_0}{\lambda}\left(1 - e^{-\lambda T}\right) \end{aligned} \tag{33}$$

Using this, the expression for the target counts per window in terms of a hypothetical constant window width is easily found:

$$\begin{aligned} C_* &= W_* N_* \\ &= W_* \frac{N_0}{\lambda}\left(1 - e^{-\lambda T}\right) \end{aligned} \tag{34}$$

Combining the above together into two unified expressions for window width gives the two required functions, which conveniently do not need the initial count rate to be

measured (the half life is known from the tracer, but the activity depends on the age of the tracer, amount injected, et al.):

$$\mathrm{w_b}(t) = \frac{1}{\lambda} \ln\left(W_* \left(1 - e^{-\lambda T}\right) e^{\lambda t} + 1\right) \quad (35)$$

$$\mathrm{w_a}(t) = \frac{1}{\lambda} \ln\left(1 - W_* \left(1 - e^{-\lambda T}\right) e^{\lambda t}\right) \quad (36)$$

2.2.1 Alternative

The alternative to dilating windows, as described above, is to decimate the data so as to standardise the count rate throughout the scan. If the above strategy is considered as modifying 'time' to ensure a constant count rate, then this strategy would be considered as changing the data to achieve the equivalent result. This has the obvious disadvantage of requiring data to be discarded from the early sections of the scan in order to reduce the count rate to match the minimum count rate encountered at the end of a scan.

This drawback is particularly acute for the early sections of dynamic studies, when the pharmacokinetics of the tracer are fast, and the exact variation is very important to the study. This is exactly the time when the need to discard data would be greatest.

Dilating the windows has an equivalent sacrifice in that the effective timescale is increased as the scan progresses.

2.3 Eventwise Implementation

PET imaging is a fundamentally discrete process – the image is formed from data made up of the records of when coincident scintillation occur in the scanner as a result of individual molecules of the tracer decaying. This key attribute of the data, combined with the fact that the histogramming step needed to form the sinograms is very slow, suggests that there is a better way of performing the analysis than that outlined earlier

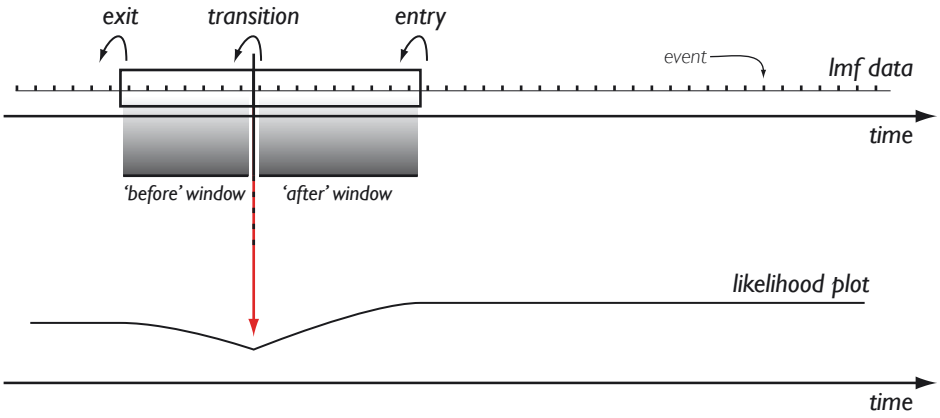

Figure 41 Definition of the three deltas and their relationship to the 'before' and after windows used thus far in calculating the value of the likelihood plot.

– namely driving the method by these events rather than extrinsically sweeping the windows through time.

2.3.1 Principal

Each prompt coincidence (called, for short, an 'event') will affect the likelihood plot 3 times – with reference to Figure 41:

Entry: when the beginning of the after window passes that event, and therefore that event contributes to the histogram for the 'after' window.

Transition: when the border between the two windows passes the event, at which point the event transitions from counting in the 'after' window to counting in the 'before' window.

Exit: when the trailing edge of the 'before' window passes the event, at which point the event drops out of consideration.

Given that the effects of each event are largely independent it is possible to consider the three individual effects of each event separately and then combine these effects, termed

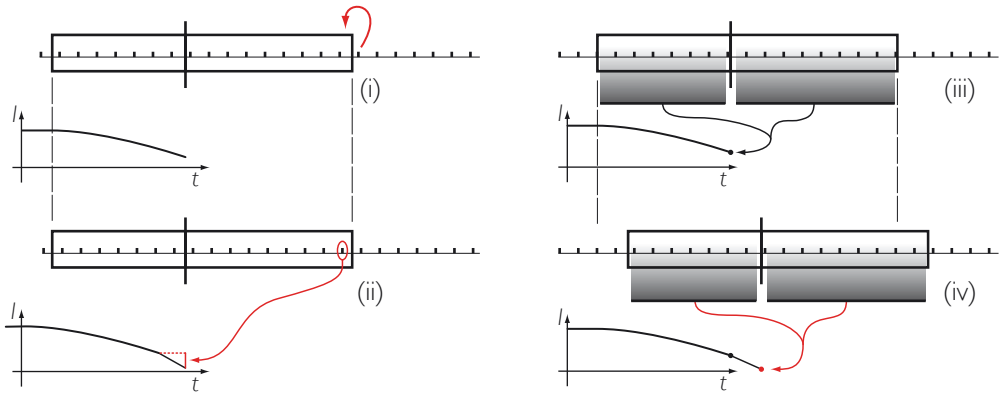

Figure 42 Comparison between the event-wise implementation (left) and the window-wise, explicit histogramming approach used to introduce the twitch detection method earlier (right). From (i) to (ii) an event has 'entered', this implies a change to the likelihood value because of that event. In contrast, at (iii) the two sinograms are used to calculate a value for the likelihood plot. The windows are move a pre-specified increment to (iv), re-histogrammed to form two new sinograms, which are used to calculate the value of the next point on the likelihood plot.

'deltas', together to form the likelihood plot by a simple accumulation (adjusting for the temporal offset between the event time and when delta-entry and delta-exit affect the likelihood).

This formulation is driven by the actual events as recorded in the list mode data. It can be thought of most clearly as the list mode stream flowing through two static windows rather than the two windows being externally slid through the data source. In addition to reflecting the fundamentally discrete nature of list mode data this makes the twitch detection have an arbitrary temporal resolution— driving the calculations the events recorded means the likelihood plot inherently has the same temporal resolution as the data. Now the only parameter that one must choose when performing twitch detection is the width of the windows rather than also having to chose how frequently to rehistogram the data and calculate new likelihood values.

Although the process still requires sinograms for the two windows they can be updated continuously — as events enter, transition, or exit from the windows — and thus the time consuming step of repeatedly re-histogramming the windows (see Figure 42) can be avoided.

2.3.2 Calculation of the Deltas

Each event only affects its own bin on the sinogram. Therefore, the effect of that event on the overall log joint likelihood (the change to the log joint likelihood caused by that event) is identical to the effect on the bin-wise log joint likelihood (i.e. the probability of observing the counts from before and after in that specific and individual bin).

$$\begin{aligned}\delta &= \mathrm{L}_{W^*}[t] - \mathrm{L}_{W^*}[t-1] \\ &= \sum_{\iota} \left(l_{W^*[\iota]}[t]\right) - \sum_{\iota} \left(l_{W^*[\iota]}[t-1]\right) \\ &\quad \text{...if the event at } t \text{ goes into bin } \iota' \text{ ...} \\ &= l_{W^*[\iota']}[t] - l_{W^*[\iota']}[t-1]\end{aligned} \tag{37}$$

This can be thought of as starting in some 'state zero', and the effect of the prompt (be it an entry transition or exit from the windows) causes the state to change from state zero to state one, therefore the delta is the difference of likelihood between state one and state zero.

Entry: For the case of the entry effect: the entry causes the encounter in the 'after' window to increase by one and the mean (Poisson parameter) to increase by one half.

$$\begin{aligned}\delta_{entry} = (s_b + s_a + 1)\log\left(\mu + \frac{1}{2}\right) \\ - (s_b + s_a)\log(\mu) - 1 - \log(s_a + 1)\end{aligned} \tag{38}$$

Transition: For the transition effect there is no change to the mean but the bin count in the 'after' window is decremented by one, and in the 'before' window incremented by one.

$$\delta_{trans} = \log\left(\frac{s_a}{s_b + 1}\right) \tag{39}$$

Exit: For the exit effect the mean falls by one half and the bin count in the 'before' window is decremented by one.

$$\delta_{exit} = (s_b - 1 + s_a) \log\left(\mu - \frac{1}{2}\right) - (s_b + s_a) \log(\mu) + 1 + \log(s_b + 1) \quad \textbf{(40)}$$

These delta values can be precomputed and the values looked up during online twitch detection. Pre-computation results in a three-dimensional array indexed by the 'before' bin count, the 'after' bin count, and then, in the third direction, the three possible effects (entry, transition, or exit).

In summary: for each prompt coincidence, calculate (or look up) the delta for the relevant effect, accumulate the deltas to form the overall likelihood plot, and finally update the histograms for the two windows (which are required to know the bin counts when computing δ – the Poisson parameter μ is implied by the two bin counts).

Note that the times at which the three different deltas apply are different relative to the time of the event: a coincidence at time *t* will affect the likelihood plot at *t* minus the width of the 'after' window as it enters, at *t* when it transitions, and at *t* plus the width of the 'before' window for its exit.

Note that the fully simplified forms of expression for the deltas are not necessarily the best for online computation. Full simplification produces a neater expression for analysis but increases the number of floating point operations required for their computation.

Figure 43 shows three plots, one for each class of delta, showing the value of the deltas for various before and after bin counts.

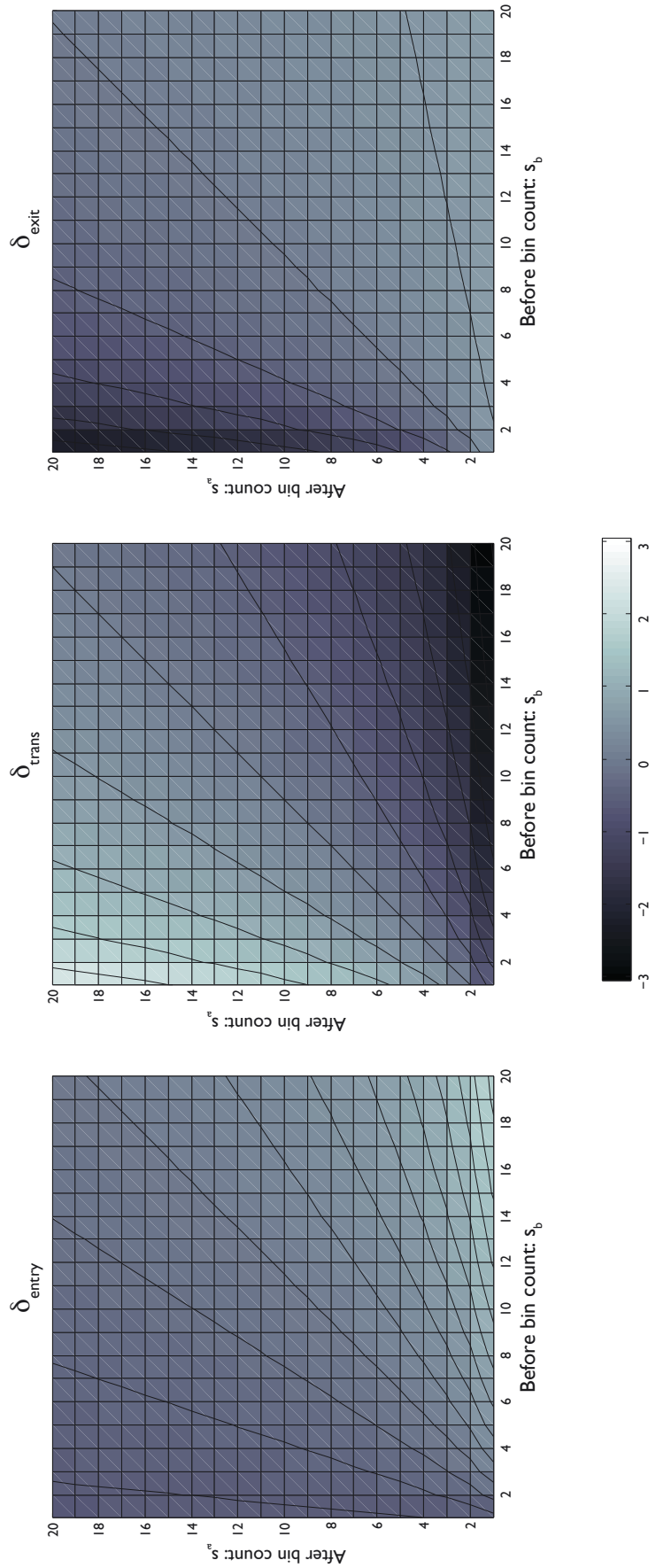

Figure 43 Individual delta values for the three classes of delta. The x-axis gives the current number of counts in the bin on the 'before' sinogram, the y-axis the same for the 'after' sinogram. The shading gives the value of the delta. Also shown on each plot are a selection of isocontours, i.e. lines along which the delta values are all equal.

2.3.2.1 Conditions for Delta to be Defined

Looking at the expressions for the three deltas it can be seen that they are not always defined. In some cases the reasons are obvious and the cases cannot occur – trying to exit a nonexistent count for instance. The other cases involve the $(s_b + s_a) \log(\mu)$ terms, which can be written as $2\mu \log(\mu)$, as μ tends to zero.

In the continuous case the limit of above expression is found using L'Hôpital's rule, albeit once rearranged to the form $\log(\mu) / (2\mu)^{-1}$. Taking the derivatives of numerator and denominator separately as per the rule, yields:

$$\begin{aligned} \lim_{\mu \to 0} \quad 2\mu \log(\mu) &= \lim_{\mu \to 0} \quad \frac{1/\mu}{-2(2\mu)^{-2}} \\ &= \lim_{\mu \to 0} \quad \frac{-2\mu}{1} \\ &= 0 \end{aligned} \tag{41}$$

However, μ is not continuous, it does not 'approach' zero: instead it steps from one to a half and then becomes zero. The derivatives used by L'Hôpital's rule do not make sense.

In the discrete case the consensus is to define 0^0 as 1 (and therefore $0 \log(0)$ as 0), because, as argued on page 162 of [31], otherwise the binomial theorem (amongst other things) would require a 'special case'. Furthermore, as the Poisson distribution can be considered as the limiting case of the Binomial distribution (as the number of trials grows large and the probability of events remains small), the use of the Poisson distribution tacitly implies that 0^0 should be defined as 1, precisely because that makes the Binomial distribution behave elegantly in the cases where there are few, rare events in many trials.

> *This section was the root of the problem with the method found during the viva. Specifically* $0.\log(0)$ *terms were considered*

undefined and such cases were ignored, introducing a bias into the likelihood plot around motions, when such cases more likely.

This caused the motions not to cause a 'V' shaped minimum in the likelihood plot, but rather an 'M' shaped feature. Further, it introduced a dependance on the order in which entries, transitions, and exits were processed. As many deltas need to be calculated within each 200 microsecond period that is the limit of the scanner's temporal resolution, the order with which they were done could influence which of them had to be ignored and thus the accumulated likelihood plot value.

Chapter V: Twitch Detection – Implementation

Having presented the method, there remains the task of implementing it before experimental work could proceed. This proved to be quite involved and so is described here.

In addition, the task of automatically locating the effects of a twitch on the likelihood plot are discussed. This is an important tool for performing experimental work, and so had to be implemented to facilitate that work, but is not presented as a major contribution of this thesis. Indeed, feature detection is a well developed field in its own right and this task is an area for future improvements.

1. Introduction

Much of the previous chapter has supposed that PET data consists of a series of discrete events in continuous time. In fact the data consists of both discrete events and discretely sampled times. Further, the events are not recorded in the same projective space that is used for the analysis.

1.1 Nature of LMF Data

List mode data consists of a stream of packets. There are many different types of packets which contain different sorts of data. These include such things as the operating parameters of the scanner hardware, prompt coincidences detected by the scanner, delayed window coincidences, timing markers, total number of counts so far in the scan, etc. Of all the different sorts of packets, just two are of interest to the method presented here. These are the timing marks and the prompt coincidences. The timing marks count the number of time increments since the start of the scan, in the case of the data used for this work, the time increments are to the nearest 200 μs. The prompt coincidences are recorded in a packet containing the hardware addresses of the two crystals that detected scintillations coming from a pair of photons emitted as a result of a positron annihilation.

The key point with real list mode data is that time is only known to the nearest timing marker; that is to say, time is only known to the nearest 200 μs increment. There may be many or few or no prompt coincidences between any pair of timing markers, depending entirely on whether or not there were any decays detected during that part of the scan. The discrete and sparse nature of real LMF data reinforces the idea of driving the twitch detection algorithm on an event by event the basis introduced in the previous chapter. (Also, note that the need to convert from crystal addresses to the sinogram coordinates further motivates avoiding the histogramming step required by the window by window approach first used to introduce the method.)

Driving the algorithm by the events with only a 200 μs resolution raises three issues:

Uneven temporal sampling: That there may be times without any events, and a variable number of events sharing the same time, means that one cannot assume that every time has events nor that any event has a unique time. This was a major motivation for developing the marginal twitch detection algorithm and it must be remembered that there is not a clear, unequivocal, 'next' time, or 'next' event.

Synchronisation of the deltas: The deltas (or changes to the likelihood plot value) are generated as events enter and leave windows. It is important that the windows are up-to-date as these events are calculated, because the value of each delta depends on the state of the two windows at the time. Therefore, it is vital that the three sources of deltas (entry into the after window, transition between the windows, and exiting from the before window) proceed in a synchronous manner. This is complicated by the potential scenario that there are no counts in either or both of the windows.

Delta arrival order: Although the deltas are generated at three different points in time, because the window edges correspond to different points in time, their effect is accumulated at a single time (called here the 'reference time'). Therefore, not only must the deltas be generated in synchrony, they must also arrive in the correct order for accumulation to form the likelihood plot. Although it would be possible, in theory, to receive a late delta and go back over the likelihood values that have been generated since that delta's time and correct the likelihood plot this is far from ideal. Thus, it is preferable to ensure that once the accumulation has moved on to a later time in the likelihood plot no further changes are needed to earlier times.

2. Implementation Strategies

2.1 Single Cursor / Virtual Time

To describe the scheme, consider following one of the many events through the twitch detection process. As discussed in the previous chapter, each prompt coincidence produces deltas, or changes to the likelihood plot value, at three points: entry, when the event starts to count in the 'after' window; transition, as the event ceases to factor into the 'after' window that starts account in the 'before' window; and exit, when the event is no longer relevant to the 'before' window. The process of producing each of these deltas will be described in turn, following the chosen prompt.

2.1.1 Entry

As the list mode file is processed packet by packet, eventually the chosen event is encountered; the current time can be defined as the most recent timestamp packet encountered (Figure 44). To begin processing this event, the hardware and addresses of two crystals must first be converted to the sinogram domain coordinates. Next, it is possible to calculate the delta and accumulate that change to the likelihood plot at the relevant reference time (the reference time corresponding to the boundary between two windows, and thus offset from the current time by the width of the 'after' window – also on Figure 44).

Next, the count in the relevant sinogram bin for the 'after' window can be incremented by one, to include the chosen event in the 'after' window's sinogram.

Lastly, the event is packaged in a token storing the sinogram coordinates and a timestamp for that event. At this point, the term 'virtual time' is relevant: the timestamp applied to the token is modified from the current time with an offset to account for the window width; the offset is chosen such that it pushes the token into the future so that it

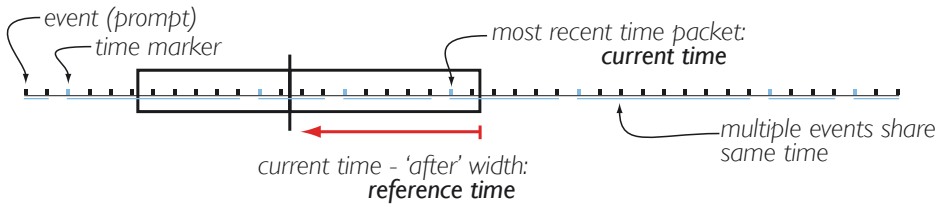

Figure 44 Definition of current and reference times.

will sit alongside other events in the list mode file (if there are any at that time) that will, themselves, be entering the 'after' window as the event being followed is due to transition from the 'after' window to the 'before' window. Therefore, the token for this event will be placed in time such that its transition delta occurs in sync with the entry deltas of later events. This achieves the key challenge of maintaining synchronisation. Figure 45 summarizes the key steps in the process.

2.1.2 Transition

The token representing the event now sits in a queue of similar tokens until it is 'pending'. When the timestamp matches the most recent timing packets (or, if there is no time marker that exactly matches, as soon as the timestamp applied to the token is earlier than the most recent timing marker in the list mode file) the event being followed is due for its transition between the two windows. Performing the transition requires the following steps:

First, the delta is calculated using the existing counts in two sinograms, and this delta is then applied to the likelihood plot at the relevant reference time (which, for the transition events, matches that timestamp of the token).

With calculation completed, the sinograms must be updated. This requires incrementing the bin count in the 'before' window and decremented in the 'after' – having stored the sinogram coordinates in the token, this is a trivial step.

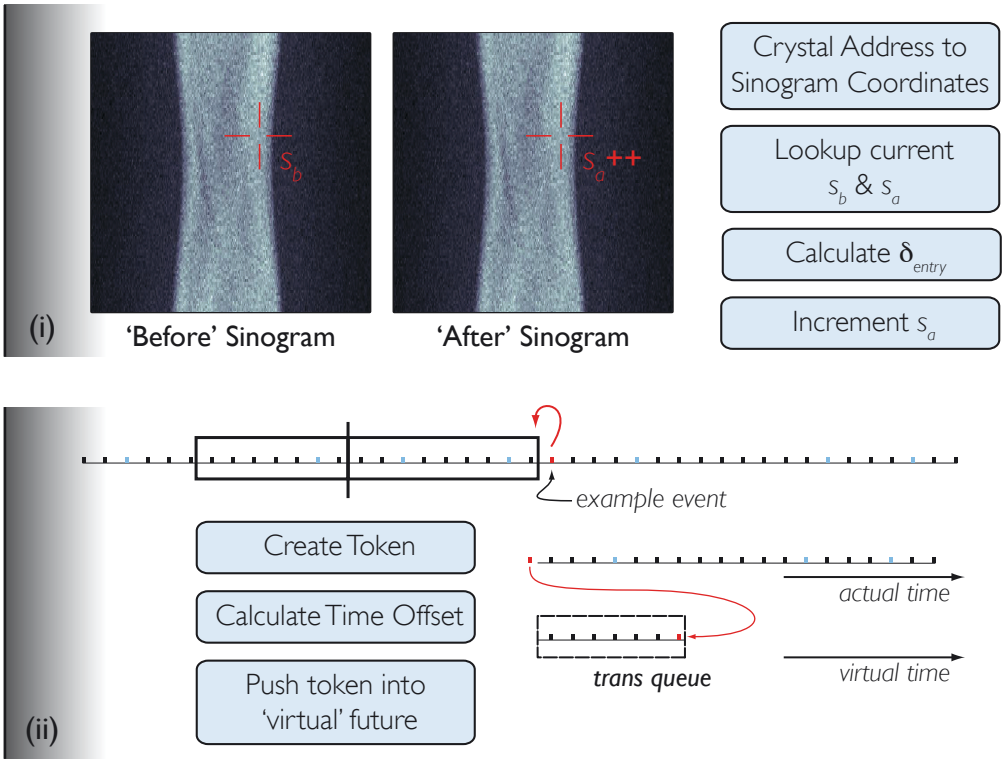

Figure 45 Entry processing schematic. Top row (i) shows the steps to calculate the change to the likelihood plot and keep the sinograms up to date. Not shown is the application of the delta to the likelihood plot at the reference time. Bottom row (ii) shows the steps to prepare the event for its next effect by pushing a token representing the event into a queue of other tokens with modified timestamps, such that it sits alongside future events in the list mode data that will be due to enter as this event is due to transition.

Lastly the token is pushed onto a queue to wait until it is due to exit from the 'before' window. As before, to maintain synchronisation the timestamp is modified before the token is pushed onto the queue. In this case the timestamp is set such that the token will sit alongside events in the list mode file that will be entering at the same point in the processing as when this event will be exiting. Figure 46 summarizes the key steps in the process.

2.1.3 Exit

The token now sits, waiting, in a queue of similar tokens until it is again pending. When it is, the exit delta can be calculated and applied to the likelihood plot at the reference time and then the sinogram for the 'before' window can be updated by decrementing the bin count at the relevant coordinates. The token can be discarded as it is no longer

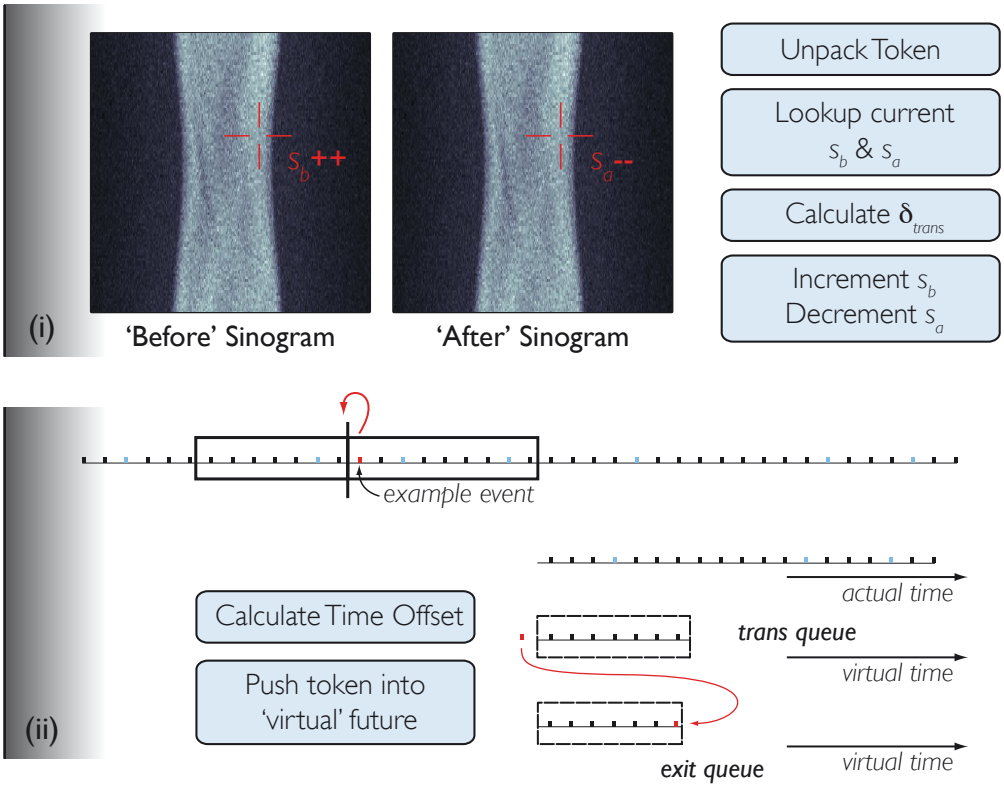

Figure 46 Transition processing schematic. Top row (i) shows the steps to calculate the change to the likelihood plot and keep the sinograms up to date. Not shown is the application of the delta to the likelihood plot at the reference time. Bottom row (ii) shows the steps to prepare the event for its next effect by pushing a token representing the event into a queue of other tokens with modified timestamps, such that it sits alongside future events in the list mode data that will be due to enter as this event is due to exit.

needed – the event has passed through both windows and has no further effect. Figure 47 summarizes the key steps in the process.

2.1.4 Discussion

This scheme is complicated upon first reading, but, as Figure 48 demonstrates, it is really a reorganisation of the windows in time such that any events before the current time should be dealt with next.

Without decay correction it is very simple because all the offsets (including both from timestamps to the reference time and the modifications to the timestamps in tokens needed to synchronise the deltas) are constant, matching the fixed window width. The

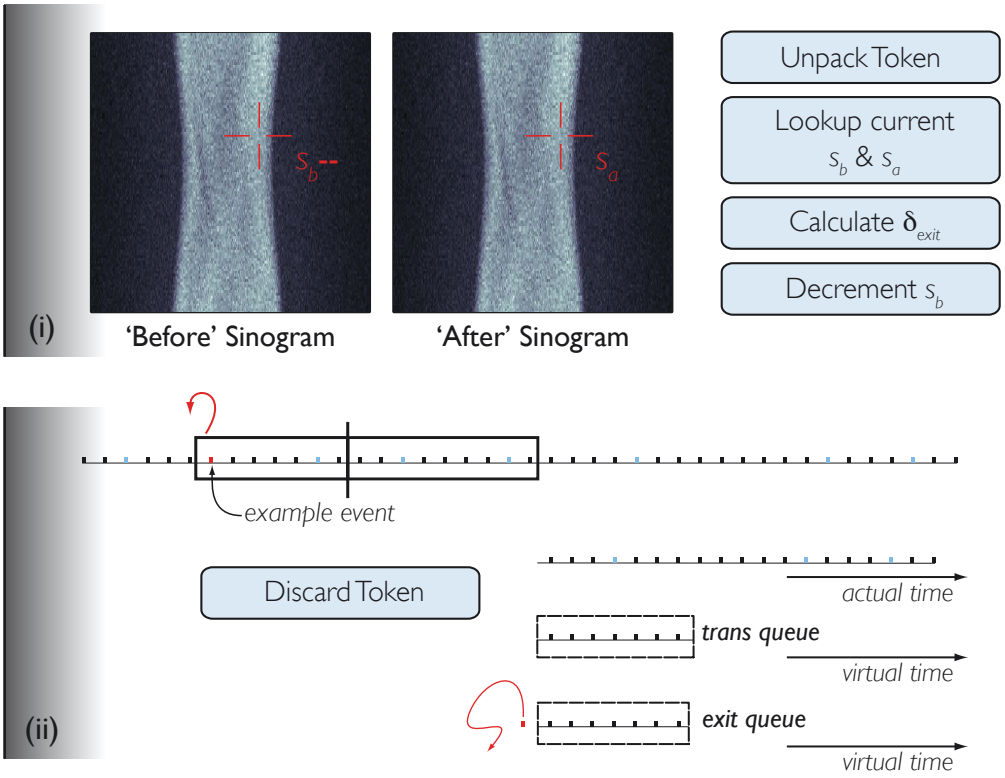

Figure 47 Exit processing schematic. Top row (i) shows the steps to calculate the change to the likelihood plot and keep the sinograms up to date. Not shown is the application of the delta to the likelihood plot at the reference time. Bottom row (ii) shows the discarding of the token, this event no longer being relevant to the method.

single cursor, i.e. the comparison of events or tokens against a single time (from the most recent timestamp from the LMF) to see if they are pending, is elegant as it ensures that the deltas are in sync and the correct order – if the earliest timestamp is always done first then the histograms are kept up-to-date and the deltas always appear in the correct order. Admittedly, it is common to find ties – two events sharing the same timestamp or, for events in the list mode file about to enter multiple events between two time mark packets – in these cases some rules must be applied to break the ties and choose the order of processing; for example performing all the exits first then the entries and then the transitioning events.

When implemented in C++ the standard template library's double ended queues serve as suitable containers to store the tokens and the list mode file is wrapped in a class providing accessor methods. With these data structures in place it is easy to peek at the timestamp of the tokens next in each queue and compare them with the current time

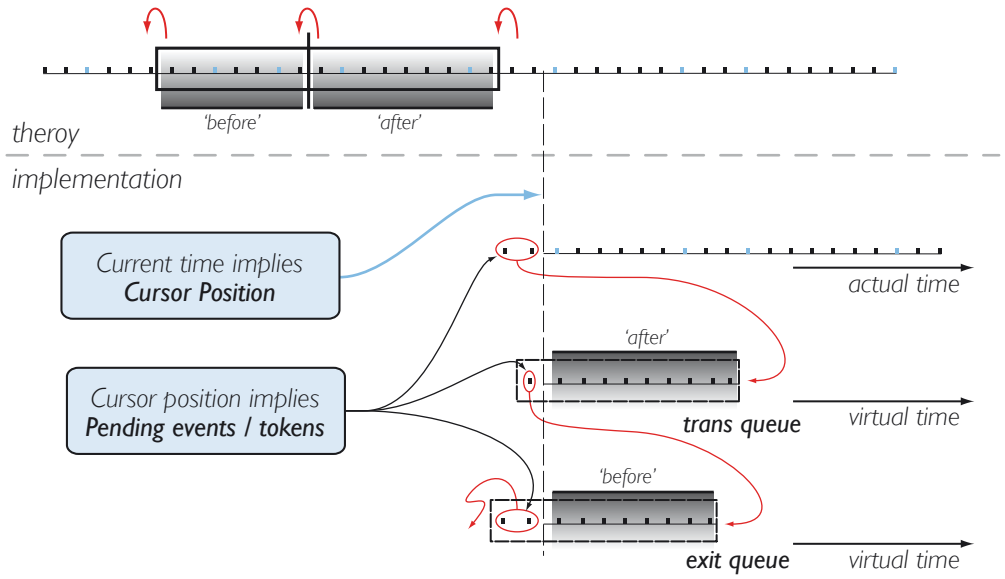

Figure 48 Summary illustration of the single cursor implementation scheme, showing how the windows used in the theoretical discussion of the method relate to queues in virtual time, and the current time is used as a single cursor, reading all the queues simultaneously.

defined by the most recent time mark packet in the list mode file, thereby deciding which event process next, at which point that token may be popped off its queue, or, in the case of an entry, the relevant packet extracted from the list mode data. This peek, choose, pop, routine can be repeated until all three potential sources of deltas are exhausted – and if one of the queues happens to be empty at some point during analysis it does not matter.

2.2 Three Cursor / Actual Time

Again, to understand the strategy consider the following one chosen event as it produces the three changes to the likelihood plot that it will.

2.2.1 Entry

The first delta, from entry, occurs when the event's time (based on the most recent time packet in the list mode file) is equal to the entry time – how this is defined will be discussed shortly. The practical process of performing the event's entry is very similar to

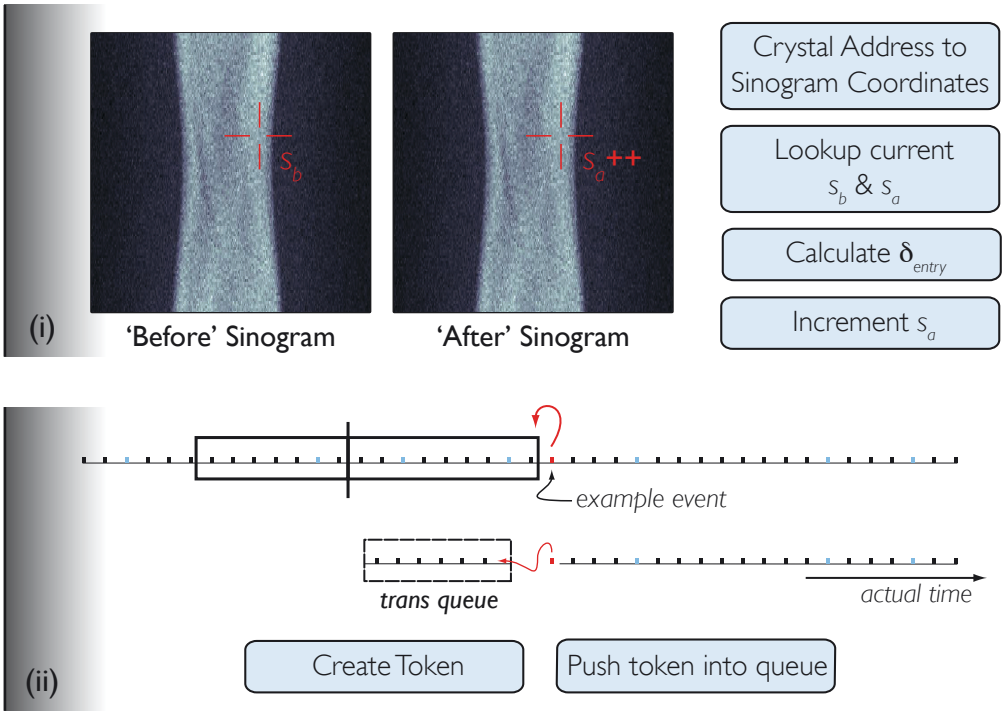

Figure 49 Entry processing schematic. Top row (i) shows the steps to calculate the change to the likelihood plot and keep the sinograms up to date. Not shown is the application of the delta to the likelihood plot at the reference time. Bottom row (ii) shows the steps to prepare the event for its next effect by pushing a token representing the event into a queue of other tokens waiting to transition.

the single cursor case. First, the crystal addresses are translated to sinogram coordinates, then the actual delta can be calculated. This delta is applied at the reference time (the offset between the event's time and reference time coming from the window width as set by the decay correction). Then the sinogram for the 'after' window can be updated by incrementing the relevant bin. As before, a token is used to store the event and is pushed into a queue of tokens for other events waiting for the transition delta to be due. The difference in this strategy is that the token holds the actual timestamp – there is no modification made so the token records the time as defined by the most recent time packet in the list mode data. See Figure 49.

2.2.2 Transition

The second effect, the transition delta, occurs when the event's stored timestamp matches the transition time (again the definition of the transition time will be discussed

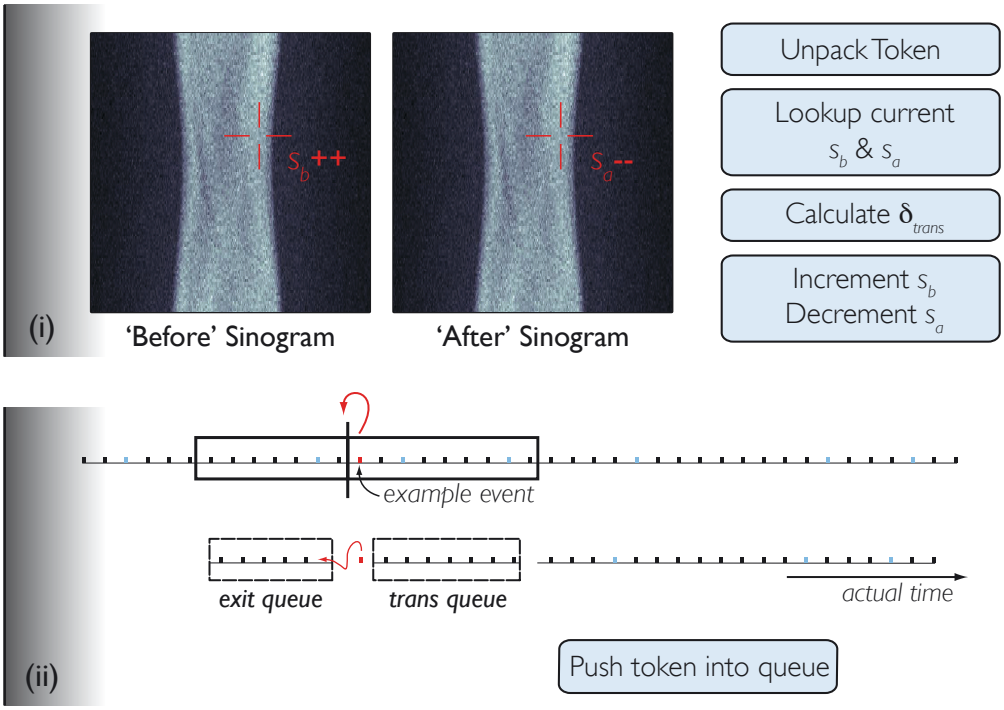

Figure 50 Transition processing schematic. Top row (i) shows the steps to calculate the change to the likelihood plot and keep the sinograms up to date. Not shown is the application of the delta to the likelihood plot at the reference time. Bottom row (ii) shows the steps to prepare the event for its next effect by pushing a token representing the event into a queue of other tokens waiting to exit.

shortly). Practically, the transition delta is calculated, then applied at the reference time (of course, the reference time and transition time are identical). Both the before and after sinograms are updated by incrementing or decrementing the relevant bin and then the token is pushed on to a queue of events waiting to exit. In the case of the three cursor scheme no modification is made to timestamp in the token. See Figure 50.

2.2.3 Exit

The third, and final, effect is the exit delta, which occurs when the events timestamp stored in the token matches the exit time. Practically, the delta is calculated, then applied at the reference time (again, the decay correction equations give the relationship between the exit time – explained later – and reference time). The 'before' window's sinogram is updated by decrementing the relevant bin, and the token is discarded. See Figure 51.

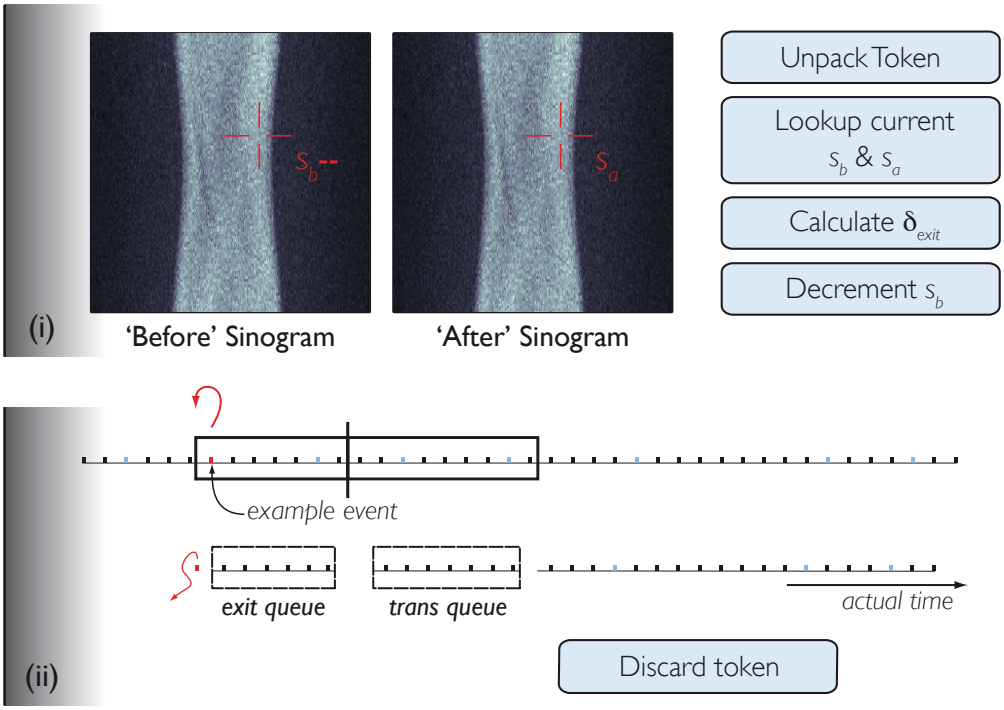

Figure 51 Entry processing schematic. Top row (i) shows the steps to calculate the change to the likelihood plot and keep the sinograms up to date. Not shown is the application of the delta to the likelihood plot at the reference time. Bottom row (ii) shows the token being discard, as the event it represents is no longer relevant.

2.2.4 Discussion

As is clear from the above, the key point is the definition of the entry time, transition time, and exit time. These have to move in lock step to ensure that the deltas arrive in sync. First though, consider Figure 52 (the equivalent schematic to that presented for the single cursor scheme – Figure 48) and the differences between the strategies.

2.2.4.1 Summary of the Differences to the Single Cursor Strategy

- Events now maintain their actual time throughout the process.

- Deltas are triggered when an event's time matches one of the cursors (i.e. the entry time, transition time, and exit time).

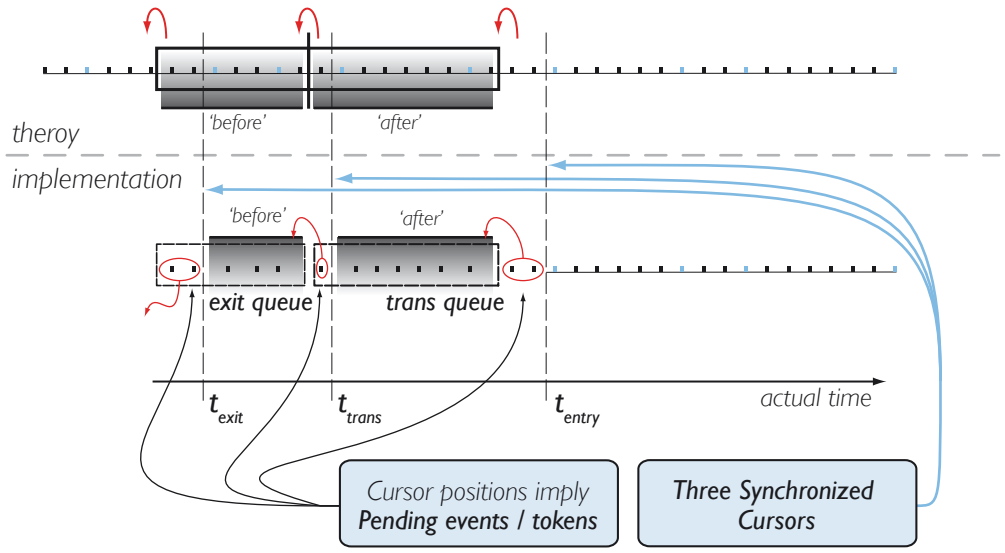

Figure 52 Summary illustration of the three cursor implementation scheme, showing how the windows used in the theoretical discussion of the method relate to queues, with three cursors used to identify events that need to be processed.

- The cursors now must be kept in sync and, remember, the updates to the cursors must be driven by the events in the queues.

2.2.5 Cursor Updating & Synchronisation

The decay correction equations (in §2.2 of Chapter IV) imply a three-way relationship between the cursors, and all can be related to the reference time.

By comparing the times of the events at the back of the queues it is possible to identify which cursor is lagging behind the others using the above relationships: the reference times suggested by all three should match up, and if one event implies an earlier reference time that the others so then that event should be processed in order to close the 'reality gap' – Figure 53. This way the cursors move when the events that they point at are behind, relative to the events pointed at by the other cursors – thus the method is self-synchronising.

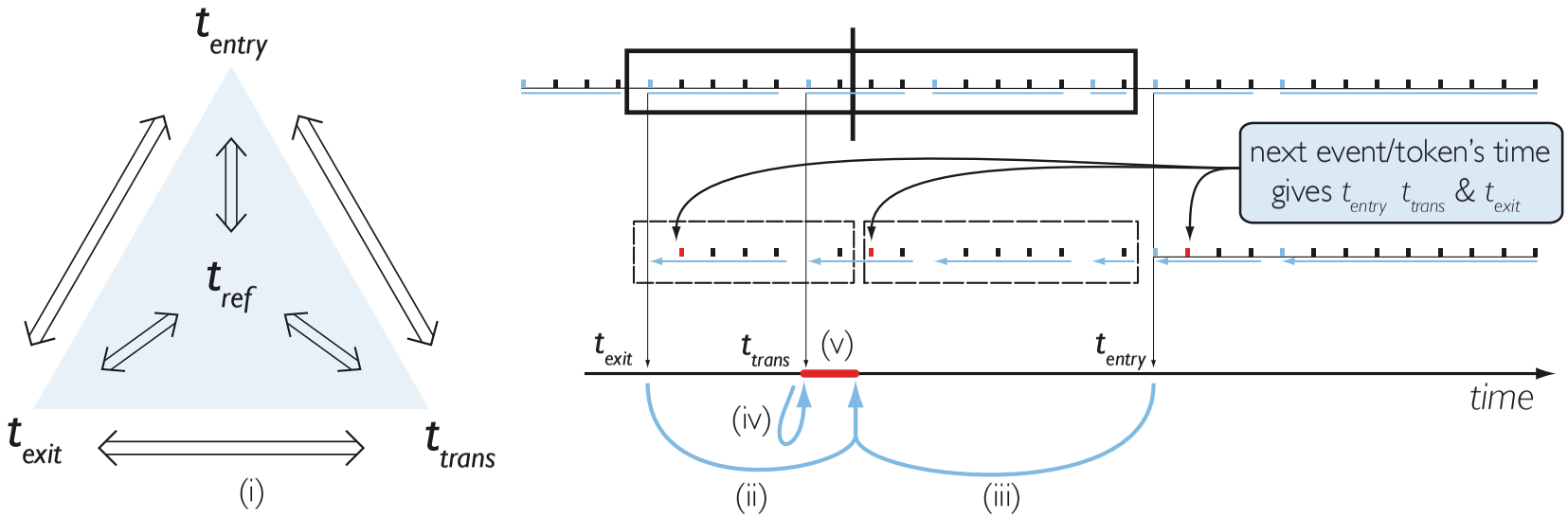

Figure 53 Cursor synchronization for the three cursor strategy. The window widths implied by the decay correction equations form the basis of a set (i) of relationships between t_{entry}, t_{trans}, t_{exit} and t_{ref} – the reference time, at which the deltas will be accumulated to update the likelihood plot. t_{entry}, t_{trans}, t_{exit} are all defined by the time of the next event to enter, transition, or exit, respectively. In the example on the right t_{entry} and t_{exit} both imply the same reference time: (ii) and (iii). However t_{trans} implies a different time (iv) and as this is 'lagging' (v) behind the other two it dictates that one or more tokens should be processed from the transition queue in order to bring that cursor back into sync with the others.

Of course, it is still possible to have ties and, as with the single cursor case, some rule for tie breaking must be used.

2.3 Choice of Strategy

Admittedly, at first glance, the single cursor approach seems much simpler. However, with decay correction the time offsets used to modify the timestamps applied to the tokens as they enter the transition and exit queues must be computed on-the-fly to deal with the variable window widths required. Thus the multiple cursor approach is actually about the same in terms of complexity: the calculations needed to keep the cursors in sync are no more complicated (but perhaps easy to understand… and debug…) than those needed to calculate how far to push the tokens into the future. Indeed the single cursor approach was used before decay correction was added to the code. The multiple cursor approach was implemented when adding the decay correction functionality, largely because the 'reality gap' concept made it easier to understand what calculations needed to be done when.

> *The discussion of the programming for this implementation is presented in Appendix A. It has been move to an appendix because, though an interesting discussion (especially the contrast with a latter programming project), it does not fit into the narrative of the thesis at this juncture.*

3. Twitch Localisation

The first figure in Chapter VI (experimental work on twitch detection) will demonstrate the concept of twitch detection with a clear feature on the likelihood plot corresponding to a transient motion event. However, to go further than an initial demonstration and start exploring the performance of the method requires a systematic, and preferably automatic, way to identify the feature on the likelihood plot. It needs a means of converting the graphs of likelihood values into specific twitch times and this process will be called twitch localisation.

Why is it so desirable to formalise the process of interpreting the likelihood plot? There are four reasons: objectivity, repeatability, numeracy, and automation.

Objectivity Human interpretation, especially in the absence of an explicit procedure, is vulnerable to interference from other factors and is therefore subjective. For instance, knowing when the twitches took place will influence the ability to identify the location of the feature on the likelihood plot. Expectations, for instance that larger motions should be easier to find, may influence the amount of effort put into finding a feature.

Repeatability Having a specified algorithm, especially one implemented by a computer, will mean that the same likelihood plot will always produce the same estimates of twitch times. This means that the algorithm cannot itself be a source of variation when considering the effects of various factors on the twitch detection. For example, if the timescale of the twitch detection method were to be varied in order to find which is most accurate but the likelihood plots were interpreted in an ad hoc manner the knowledge of previous results would aid later experiments, biasing the result in favour of whichever timescales were tested later.

Numeracy For the purposes of investigating trends in the performance of twitch detection it is necessary to have a specific time for the identified twitch, not just a qualitative description that there is some effect visible on the likelihood plot.

Automation Both in the context of experimental work and eventually in real world use there is a need to be able to extract the twitch time automatically, ideally with no human input.

One point to stress here is that the output sought is a list of one or more twitch times. It is not enough to simply process the likelihood plot to make features clearer, there must be some decision making step in the algorithm – if only a thresholding step. There is a need for some inference, a point at which a conclusion is drawn from the data. This will make the process non-linear, with all the attendant complications in understanding its performance.

Furthermore, adding this step introduces a complication in assessing performance: the results can limited by the twitch detection method or by the twitch localisation algorithm, there may be some coupling between the two (for instance the optimal choice of timescale in the twitch detection may depend on what localisation method is used.

The rest of this section describes a method developed to perform twitch localisation for the purposes of the experimental work in this thesis. The localisation stage is important and initial investigations of a variety of options proved very interesting: it leads into many areas of signal processing and inference. Further discussion of other approaches considered is in Appendix B, but they are outside the main focus of this thesis.

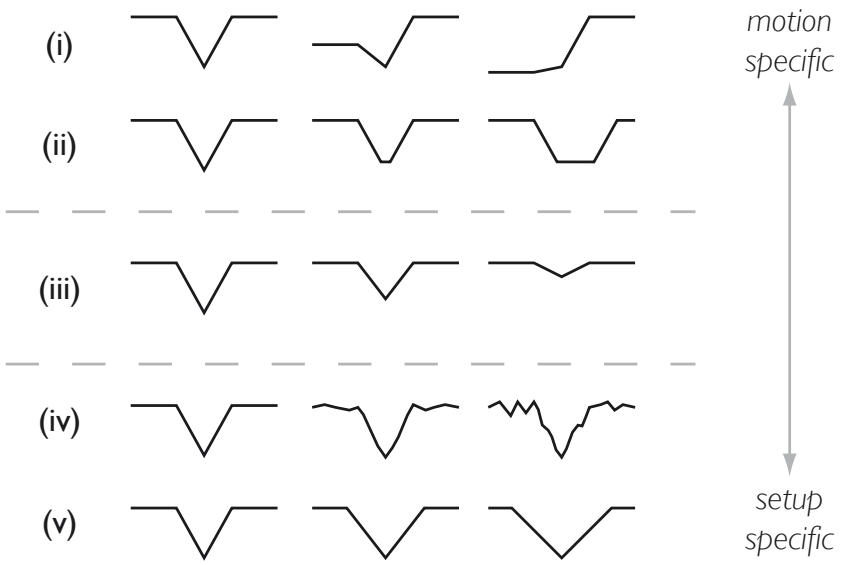

Figure 54 Variations of how twitches can manifest on the likelihood plot. Some types of variation are principle determined by the specific motion. The mix between the 'V' and the slope (i) and the effect of non instant motion (ii) are two examples. The depth of the 'V' (iii) is another, but it is also influenced by the setup of the twitch detector's timescale. The amount of noise (iv), width of the twitch feature (v) are less variable, being determined by the setup of the scan and the twitch detector.

3.1 Challenge

The way that twitches manifest on the likelihood plot is highly variable. As in Figure 54 the variations can be split in two: some are a result of the specific motion, some are the result of the scenario such as the scan and the timescale parameter.

In the former category is the mix between the 'V' feature and the slope between baseline levels shown in Figure 59 on page 164. Also there is the range of durations of the twitches. Lastly there is the size of both individual effects, which in the case of the depth of the 'V' is also partly determined by the timescale of the twitch detector and so straddles both categories.

The variations in the second category are mainly determined by the setup of the scan (e.g. the amount of tracer) and the timescale of the twitch detector. The variation of

signal to noise ratio of the likelihood plot and the duration of an instant twitch's effect fall into this category.

3.1.1 Commonality

What is common to all of these is the disruption around the time of the twitch over a section of the likelihood plot roughly corresponding to twice the timescale used to form the likelihood plot. This size of section will be referred to as a ***block***.

3.2 2-stage Strategy

The strategy initially used in this thesis consists of two stages. The first is a biased, but very robust, approximate localisation. The second stage incorporates more information from the model of how twitches manifest on the likelihood plot in order to refine the estimate.

3.2.1 Stage 1: Local Variance

Common to all the manifestations of twitches on the likelihood plot is a disruption of approximately known duration. As a proxy for 'disruption' local variance is used: compared to a section of likelihood plot without motion a section with the 'V' shape or a slope between baseline levels will have a higher variance. Given that the disruption is expected to last about twice the timescale of the twitch detector it makes sense to define 'local' as such, computing the variance for sections of likelihood plot of that duration – Figure 55.

Consider a local section centred on a twitch. The likelihood plot can be modelled as having a baseline level, plus some mix of a 'V' and a slope, with noise. Of course the

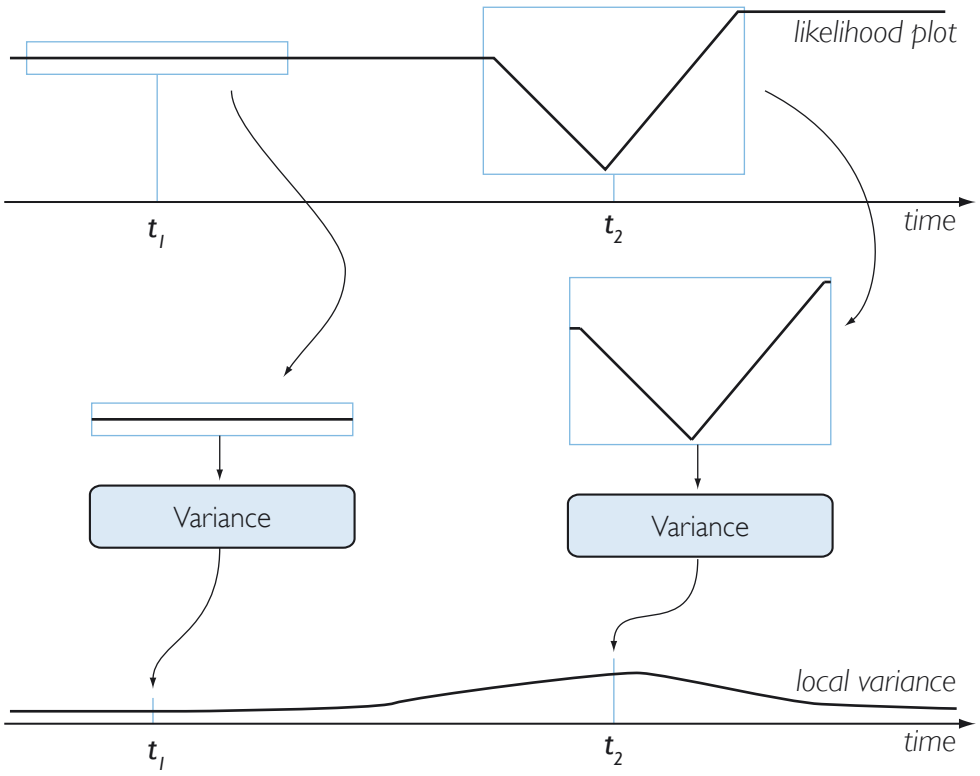

Figure 55 Computation of local variance. At each time the local variance is the variance of a block of the likelihood plot centred at that time.

slope is, in fact, a change to a new baseline level, but for a block it is observed only as a slope.

$$block_{twitch} = baseline + \alpha V + \beta slope + \gamma noise \qquad (42)$$

α and β represent the relative contributions of the slope and 'V' components of the twitch feature. They are, therefore, characteristics of the twitch. γ represents the amount of random variation on the likelihood plot and is, therefore, a property of the scan (and timescale used). Taking the variance of this section of likelihood plot (being a constant, the baseline has no variance):

$$\begin{aligned} \mathrm{Var}(block_{twitch}) = \; & \mathrm{Var}(baseline) + \alpha^2 \mathrm{Var}(V) \\ & + \beta^2 \mathrm{Var}(slope) + \gamma^2 \mathrm{Var}(noise) \end{aligned} \qquad (43)$$

Setting α and β to zero removes the effects of the twitch, the result is the corresponding model for the section without a twitch:

$$block_{no\text{-}twitch} = baseline + \gamma noise \qquad (44)$$

The local variance in this case is:

$$\mathrm{Var}(block_{twitch}) = \mathrm{Var}(baseline) + \gamma^2 \mathrm{Var}(noise) \quad (45)$$

With the term representing the noise being a property of the scan and the twitch detection method it is the same in (43) and (45). Whatever the variance of the slope and 'V' are, or whatever mix between the two there is, the variance with a twitch will be higher than in the non-twitch case (because none of the terms can be negative).

3.2.1.1 Off-Centre Blocks

It was found early in the use of this method that the local variance in the twitch case above is not necessarily the maximum – there are cases when the local section does not include all of the twitch features that produce higher local variance values.

The reason for this is that the slope will, to some degree, cancel out one side of the 'V', making it appear flatter. See Figure 56. Combined with this is that the symmetry about the point of the 'V' may reduce the variance created by that section than the kink from a steep slope to flat – the slope and the other side of the 'V' reinforce each other and so appear steeper. The result is that the local section that contains the most variance may be biased towards the 'high' side of the twitch (the side with the higher baseline level) as this is the side of the 'V' that is reinforced rather than reduced. Not surprisingly, the greater the change in baseline, the greater the bias – until the slope starts to exceed the point past which one side of the 'V' is cancelled out and starts to create a feature with two slopes, both upwards. At that point, both sides of the feature again contribute variance and so the bias will fall.

The maximum bias that would be expected is equal to the width of one of the windows of the twitch detection. With reference to Figure 57, in the extreme case that one half of the 'V' is completely suppressed by the slope the twitch will manifest only as a slope

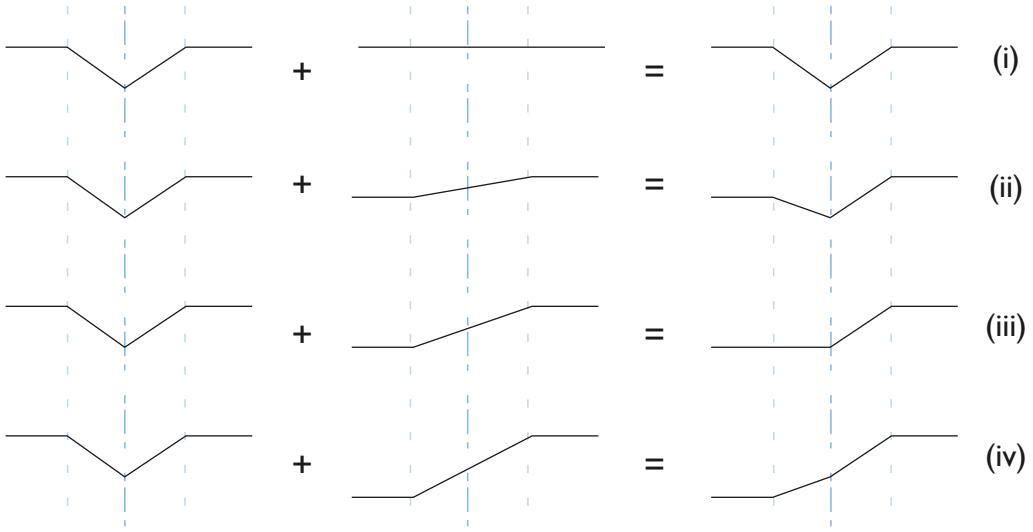

Figure 56 Illustration of how the slope component of a twitches manifestation interacts with the 'V' shape. In case (i) the change of position does not impact the baseline level of the likelihood plot, so there is no slope, and the manifestation in pure 'V'. The local variance will greatest when centred on the twitch time. Increasing the slope component leads to (ii) – note that one side of the 'V' is steeper that the other, and so the variance of that section is greater. Moving to (iii) one side of the 'V' is completely cancelled out, so the local variance will be centred on that side. A greater slope results in (iv), and the bias will be reduced.

on one side of the twitch time. In this case it might be conceivable that the maximum local variance is from a region a full window with off from the true twitch time, but not more, because moving further would remove some of the slope in order to replace it with more consistent, no-twitch points. All of this only holds in general because it is based on the measured variance using data produced by a stochastic process; the noise, other effects on the likelihood plot (e.g. from the ongoing cardiac motion) could cause unusual outcomes. *The solution to the bias will be addressed shortly.*

3.2.1.2 Size of Local

Given that the duration of the motion is not known the size of the section that will be disrupted is not known either. The minimum duration is twice the window width of the twitch detection method – but with decay correction this varies over the duration of the scan and likelihood plot. (It would be possible to implement a local variance calculation that takes this into consideration however.) The question, therefore, is whether it is wise to allow extra leeway over this minimum.

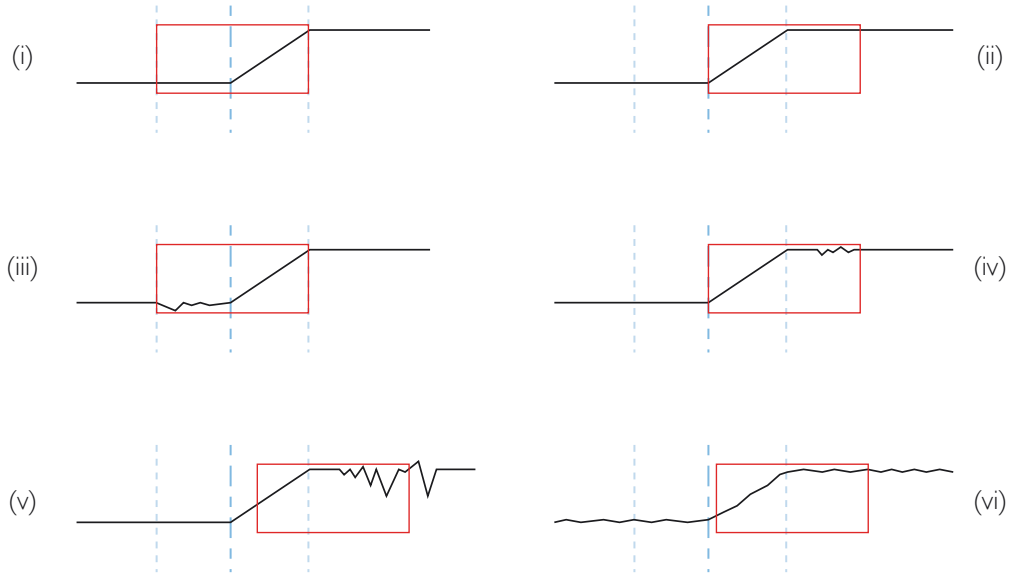

Figure 57 Discussion of various extreme cases of bias in the local variance. As (i) and (ii) show in the extreme – and highly idealised – case that the slope exactly cancels out one half of the 'V' there is nothing to differentiate between the block centred on the actual twitch time and when there is a bias of one window width. Adding a burst of strong noise in (iii) and (iv) pulls the block corresponding to maximum local variance to one side of the other. If the burst of noise were to continue for longer (through some coincidence) then, as in (v), the bias could exceed the window width. In (vi) the superposition from other motivation effect, say from breathing or the cardiac cycle, can pull the bias off one way or another.

In the case where the actual disruption is longer than the sections used to compute local variance there will be more chance of multiple maxima in the local variance. Though it is possible for there to be multiple maxima from a single twitch when the local section is longer than the twitch's effect, it is less likely. This is because the longer the section the more likely the data within it will be representative, so the less likely it is that a change of a small number of point will significantly perturb its properties, variance included – as in (iv) and (v) of Figure 57 – to get multiple maxima with a section longer than the twitch effect requires unusual coincidences for the noise, the longer the section the more such noise point will be needed to have the same effect, and this is less likely (assuming the noise at each point is independent).

The case of a sub-minimum duration section moving within the twitch's effect fits this explanation also – inside the twitch effect there are different populations and shifting the section will, therefore, be able to move into regions with different properties – because

the properties change there is no need for an unlikely coincidence to produce enough effect.

Thus, in general, it is better to err on the side of a longer duration for defining 'local'. It will produce less clear peaks – after all, the disruption of the twitch is within the local section for longer – but it is less likely to cause problems with multiple maxima.

3.2.1.3 Multiple- and No-Twitch Cases

In addition to the problem of the bias there is also the problem of how to identify multiple twitches, and its converse, how to identify the absence of twitches. Both are a result of seeking the maximum local variance to locate the twitch. The general strategy would be to identify some threshold of local variance above which it is considered significant, and then find a maximum for each region above that level of significant.

However, this is not needed to satisfy the requirements of the experimental work in this thesis; for that it would be acceptable to ensure that there is only one twitch in the likelihood plot. This does limit the ability to assess the false positive performance of twitch detection to a qualitative assessment of the likelihood plot, but an attempt to quantitatively characterise the sensitivity and specificity of the method would need to consider both twitch detection and twitch localisation, along with their interaction.

3.2.2 Stage 2: Model Based Refinement

The local variance approach outlined above suffers from a bias that can be significant. However, it does provide an estimate that is approximately correct and works across the spectrum of effects that a twitch would be expected to have on the likelihood plot. It is, therefore, a good starting point to find which part of the likelihood plot warrants further attention.

Beyond noting that both the slope and 'V' increase variance over the otherwise flat likelihood plot, the model of a twitch's effect has not been used so far – despite the fact that it explains the bias. Therefore it seems reasonable that the model could be used to correct the bias.

Having selected a region to focus on, the slope component is estimated from the baseline level before and after the disruption. These levels must be an average of many points in order to combat the noise, and the points must be sufficiently far from the maximum of local variance to fall outside the disruption caused by the twitch, allowing for the bias. However, if selected from too far away they will be more likely to be unrepresentative of the baseline level either side of the slope. In early experiments with simulated data, two window widths away from the maximum was found to be sufficient to cover all observed biases – as would be expected as by that point the local variance would be being computed for the no twitch case. Note that for these experiments ample time was left between motion to avoid 'collisions' and this issue should be addressed in future work.

With the estimated slope found it can be removed from the likelihood plot to recover the 'V' and the twitch time could from the minimum of the 'V' (give or take any effect from noise). However: the slope is of limited duration and centred on the twitch time – of which there is only a biased estimate available. It might be possible to estimate the amount of the bias by trying to estimate the relative magnitude of the slope and 'V', but this is not done as it risks becoming over reliant on the model and ignoring the fact that it is only an approximation of the real manifestations of twitches on the likelihood plot.

Figure 58 shows annotated plots of the 2-stage localisation algorithm running on preclinical data used in the next chapter.

There is one further piece of information that can be incorporated: the direction of the bias is known. As discussed, it will always be to the 'high' side of the true twitch time,

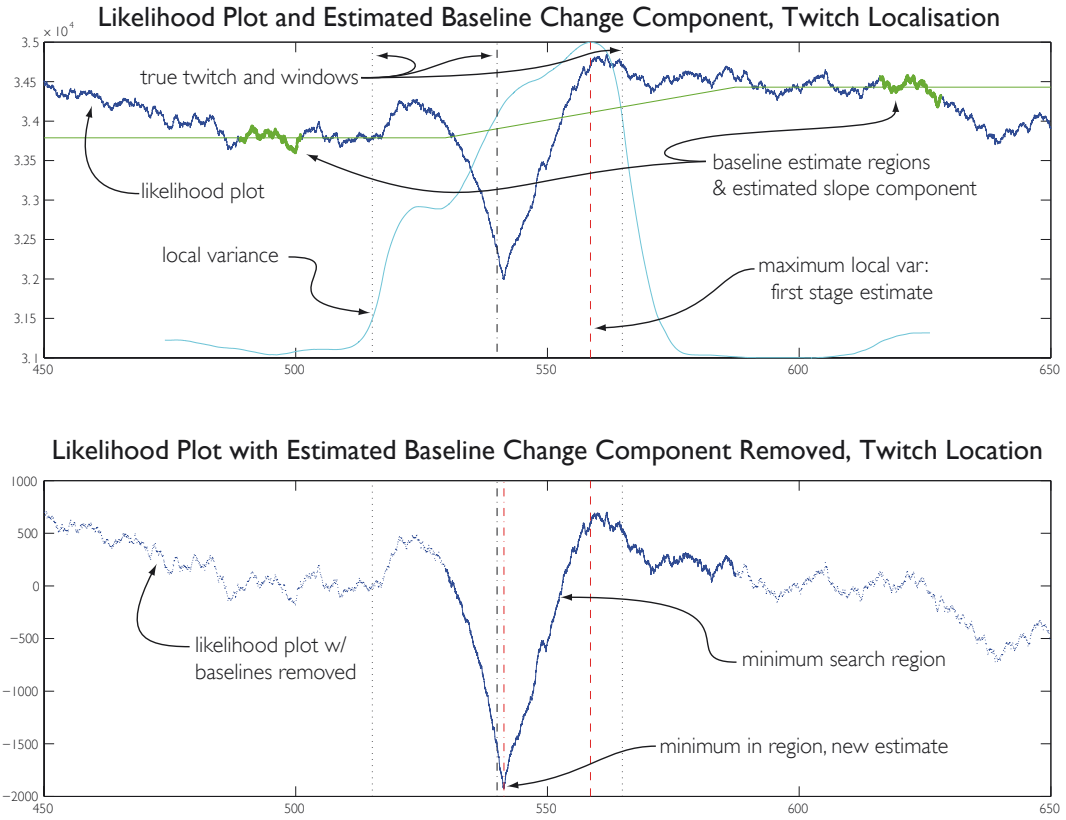

Figure 58 Annotated plots from the 2-stage local variance twitch localisation method. See text for details.

so the search for the minimum of the 'V' is only on the 'low' side from the maximum of the local variance.

3.2.2.1 Non-Instant Twitches

The chief weakness of the slope and 'V' model is that it ignores the non-instant twitches, which would be expected to produce a far more flat bottomed manifestation on the likelihood plot. This would obviously cause problems when trying to identify the minimum of the 'V' as the time of the twitch. Two solutions are proposed: conditioning and preemption.

Conditioning would entail filtering the likelihood plot before trying to find the minimum of the 'V'. Using a box-car (or rolling mean) filter would mix some of the sloping section into the flat region – provided the filter is longer than the duration of

the flat section. The result would be a broad minimum, but not a flat section. In the case of an instant motion (or one much shorter than the box-car) the result would be a broader minimum. In all cases noise would also be smoothed out. The filtering can bias the minimum however: if the slope of the sides is asymmetric then the minimum after the box car will be biased towards the side of lesser slope. This is, therefore, not considered a good approach.

Preemption seeks to put the interaction of the twitch detector and the localisation to good use, by recognising that it is the duration of the motion relative to the timescale of the twitch detector that is the difference between twitches that appear instantaneous and those that do not. However there is a trade off in that the longer the timescale of the twitch detection the rarer twitches must be – too close together and their effect will start to overlap.

Neither strategy attempts to estimate the duration of the motion. This would be an obvious third strategy, again to be investigated in future.

3.3 Multi-scale Localisation

During the process of making corrections to the thesis and repeating the experimental work this approach to localising the twitches was developed. It proved sufficiently successful to include it in the revised manuscript, not least because it overcomes a key limitation of the above method, the assumption that there was only one twitch.

The timescale parameter has been suggesting a multi-scale twitch localisation throughout the project. First, consider the evolution of the likelihood plot as the scale varies. Figure 59 shows the likelihood plots as a colour map, after normalising each scale's likelihood plot by removing its mean and scaling it to a 0–1 range. The obvious problem with this

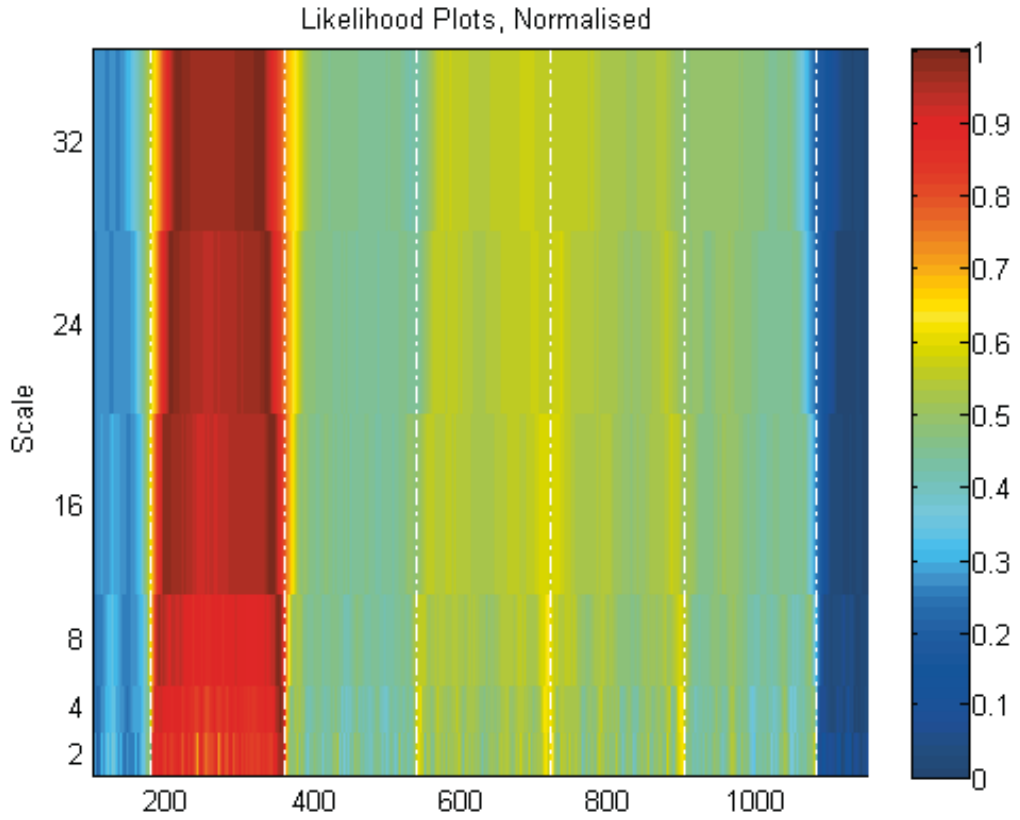

Figure 59 Likelihood plots, after normalisation by removing mean and scaling to the 0–1 interval. for some real data (see next chapter). Twitch times marked by white chain-dashed lines.

view is the range of possible twitch manifestations – they can form 'cliffs' or 'trenches' across the scale-space.

3.3.1 Local Variance... Again

What is required, therefore is a means to highlight the full range of twitch manifestations. Of course, this is exactly the same problem as the first stage of the 2-stage method, and the local variance is again a valid tool to do so. Figure 60 and Figure 61 show the local variance (of the ***un-normalised*** likelihood plots) before and after normalisation –here normalisation again removes the mean and scales the results to the 0–1 range, but the values are converted to a log-space to make the variations of the local variance more visible (but preserving maxima and minima).

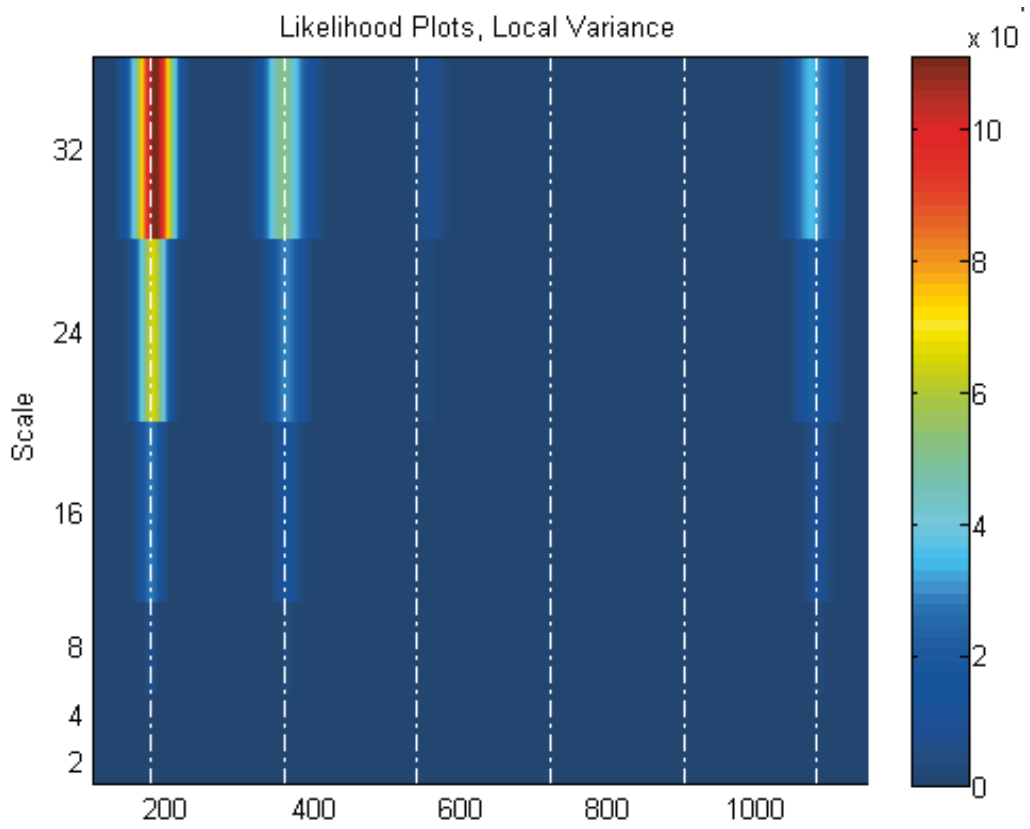

Figure 60 Local-variance of the un-normalised likelihood plots of Figure 59.

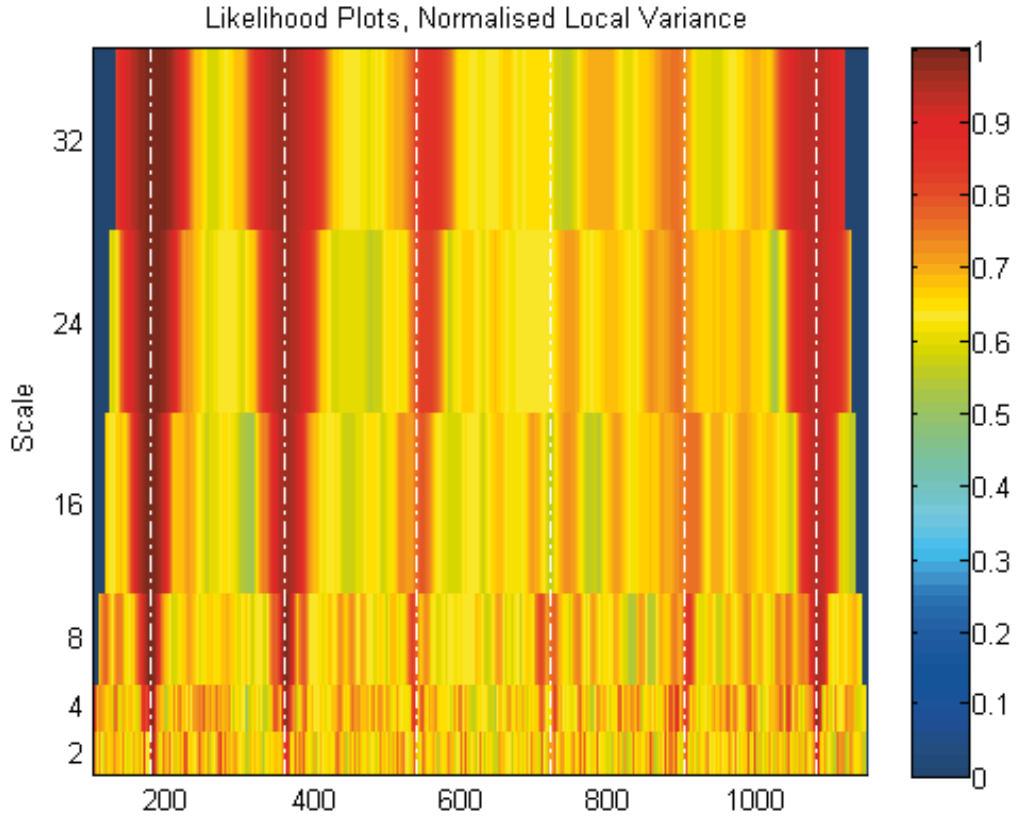

Figure 61 As Figure 60 but normalised, removing the mean and scaling the results to the 0–1 range, but the values are also converted to a log-space to make the variations of the local variance more visible.

The local variance is known to be a potentially biased detector for the twitches: recall that the more the slope and V components of the twitch manifestation cancel each other out the greater the bias will be, but is roughly limited to be less than the timescale. In light of this, the tracking of the maxima through scale space must be aware of the diminishing bias.

3.3.1.1 Peak Tracking

Consider a maximum at the 32 second timescale: if the twitch has manifested on the likelihood plot in a manner that tends to cause a highly biased maximum in the local variance then the true twitch time is probably about 32 seconds before or after the maximum. Further, the maximum in local variance from the 24 second timescale will probably be biased by a similar amount in the same direction – so the maximum should be about 8 seconds from the one already found. If, however, the manifestation produces no bias then both maxima will be at about the same time. Therefore the maximum at the 24 second scale should be within about ±8 seconds of the one found at the 32 second scale. After the first two scales have been processed they give an indication of the direction of the bias, which can also be incorporated when moving to finer timescales.

In practice, it is sensible to allow some extra leeway: 60% of the change of scale in both directions for the results shown in this work. Note that this does allow the bias to change direction as well.

If no maximum is found in the allowable region the tracking is abandoned and the next maximum in the highest timescale is considered. At the moment, maxima at finer timescales can be used several times, i.e. deemed to be related to two or more maxima at a coarser timescale. Figure 62 shows peak tracking using these rules.

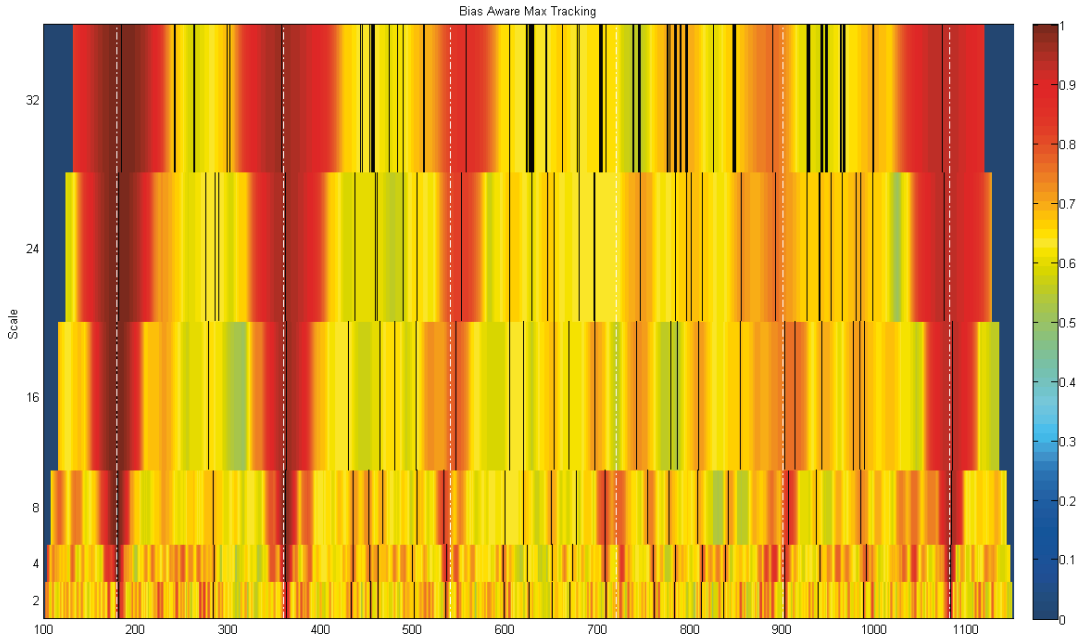

Figure 62 Tracking the local-maxima through scale space. The identified maxima are shown with the black lines. See text for the details of how the tracking is performed.

3.3.1.2 Pre-thinning

The algorithm as discussed so far was very promising, but produced numerous false positives(although the sharing of maxima helps `funnel down' with scale and reduces the number of false positives). There is another piece of information, not yet used, that could be incorporated to reduce the number of spurious maxima: it should be more difficult to detect two twitches close to each other, relative to the timescale used to generate the likelihood plot, because within twice the timescale the manifestations will overlap and interact.

Two approaches to performing the thinning will be demonstrated in the experimental work, see Chapter VI . The first involves requiring the maxima, for all timescales, to be at least one and a half timescales apart from one another during their detection. The second uses a morphological closing of each timescale's local variance (after normalisation as described to suppress small variations before detecting local maxima. The closing is done with a shape element of a flat rod of length equal to one and a half timescales,

and is only applied to the two coarsest timescales – this combination was settled upon through a process of trial and error and is not necessarily optimal.

> *Any choice between these two methods is a somewhat moot point: the twitch localisation in this thesis is only presented as a tool to facilitate experimental work, and is acknowledged as one of the more pressing areas for continued work in future. A variety of others options that were explored are discussed in Appendix B.*

Chapter VI: Twitch Detection – Experiments

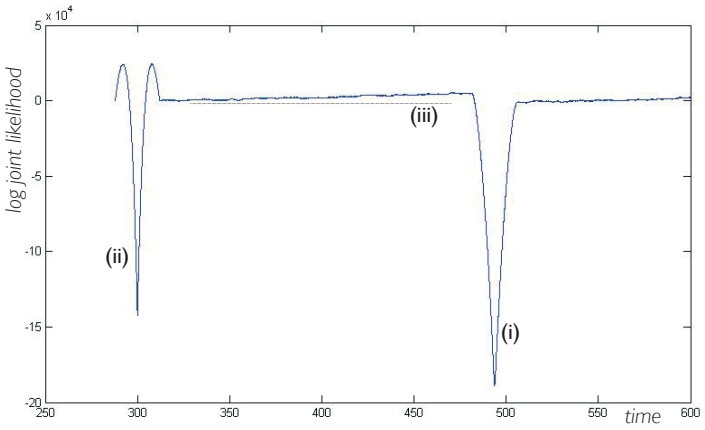

Figure 63 The output from the first successful run of the event-wise implementation of the twitch detector, using preclinical data, showing the manifestation of a twitch (i). Still visible is a bug to do with initialization of the process (ii) and the drift in likelihood due to decay (iii) – decay correction was yet to be implemented.

1. Introduction

Given that twitch detection from raw scanner data and the concept of the likelihood plot are novel propositions the first experimental task is to demonstrate the potential method. Perhaps Figure 63 has already done that?

The content of this chapter was significantly altered during the corrections made after the viva. One of the surprising effects of the change to method was the patterns in the results became much more apparent, leading to more material for discussion.

1.1 Purpose

The experiments presented in this chapter seek to begin to characterise how the method performs. For example, understanding the way twitches manifest on the likelihood plot such that they can be located automatically. In addition, the influence of motion size and speed on the manifestation of twitches on the likelihood are of interest, and the effect of varying the timescale parameter. The latter also provides a way to understand how different count rates will affect the likelihood plot: a lower dose of tracer would reduce the count rate, and therefore the number of counts in each window – the effect should be the similar to using a shorter timescale and 'rescaling' the time axis.

At present the twitch detection algorithm is set to ignore events that do not occur 'in plane'. In other words the 3D counts are ignored and the algorithm only considers a stack of 2D slices. The reason for this was to make it easier to inspect sinograms while developing and testing the code, and as the results were adequate the limitation was not removed. Including 'cross-plane' events would increase the effective count rate, but also the number of bins amongst which the counts would be distributed. The two

effects would be expected to offset each other's effect on the likelihood plot. Altering the histogramming is a topic for future work, discussed in Chapter VII.

1.2 Data

The data used in this chapter is drawn from three preclinical studies conducted specifically for this work. The reasons for using preclinical studies have been discussed in (46). The simulated data used during the development of the method has not been used for these experiments because the pseudo-LMF system needed to convert the PET-SORETO simulated sinograms was implemented in Matlab, whereas processing real data required the use of the C++ implementation (for performance and memory reasons). As a result the latter wound up being better developed, for instance incorporating decay correction, but the two are incompatible and the work required to add reading pLMF data to the C++ implementation was not practical.

Mouse The first study involved a mouse, moved by hand inside the bore of the scanner at predetermined intervals. This study was principally to facilitate development of the implementation – e.g. reading file formats and demonstrating a proof of concept likelihood plot. A very large dose of FDG was used, as it would be better to have more counts than too few at first. The activity at the start of the scan was measured at 21.4 MBq. The mouse was administered an overdose of anaesthetic prior to the scan after the tracer had reached a steady state distribution – this make performing the scan easier and safer; it is difficult to keep the animals in good health under anaesthetic in the restricted confines of the scanner and so it is more humane to put the animal down before the scan. In this study the motions were across the bore, and were 'free-hand', sliding the animal strapped to a small pallet over the main bed.

Rat 1 The second study used a rat. The larger animal should provide more detailed spatial information. However, only a section of the animal is inside the field of

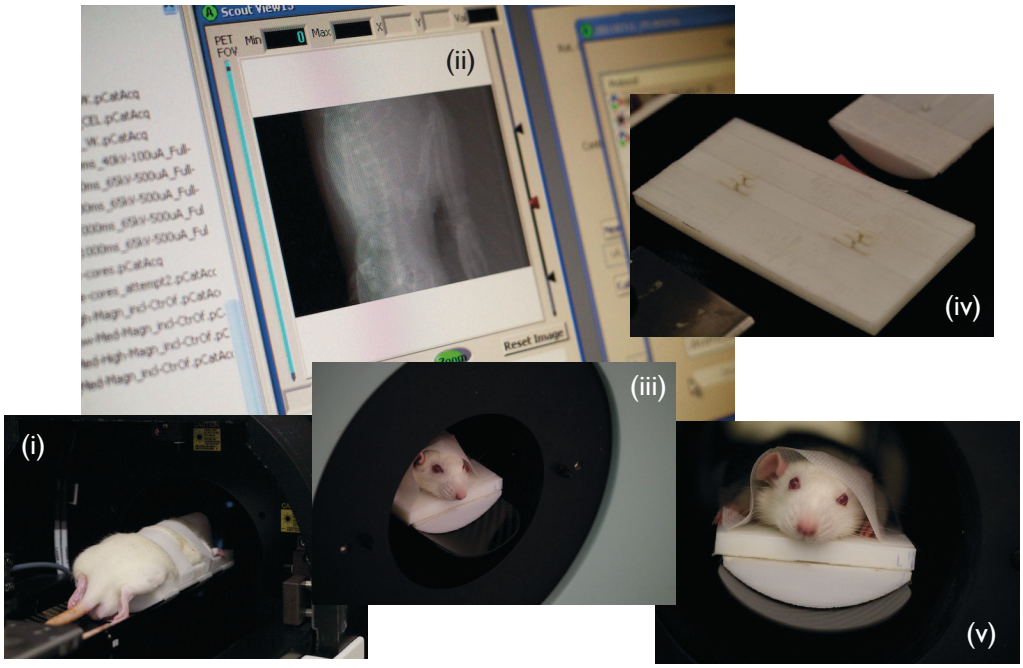

Figure 64 A montage of photographs taken during the 'Rat 2' imaging session. (i) shows the rat entering the scanner, passing into the CT section. (ii) shows the real time preview of the attenuation correction CT, indicating the section of the animal that will be scanned. (iii) shows the animal in position for the PET acquisition, with access out the back of the scanner. (iv) shows the underside of the top half of the motion jig, with guides that define the sizes of the motions – small pins in the lower half slot into these grooves. (v) shows the access at the rear of the scanner through which the jig is moves.

view during imaging (the majority of a mouse). A section through the thorax was used. This is more comparable with scans done with humans. Again the animal was euthanized before being imaged, and the dose was more in keeping with standard practice; at the start of imaging the activity in the scanner was measured as 16.4 MBq. This time the motions were more controlled: a small jig was used produce repeatable motions along and across the axis of the scanner, at varying speeds. This study provided the majority of the experimental data.

Rat 2 Late on in the process a third study was performed, this time using a live rat. This study was performed to provided data with the other motions like breathing and heart beat. A similar dose of FDG was used, and allowed to come to a stable distribution before imaging – activity at the start of imaging was recorded as 26.7 MBq. The same motion jig was used.

Name	Date	Activity	Motions
Mouse	12/08/2010	21.4 MBq	Freehand Translation 5 & 10mm Trans-axial
Rat 1	31/03/2011	16.4 MBq	4.5 & 9mm Pre-measured motion Axial and Trans-axial, All fast movements
Rat 2	12/07/2011	26.7 MBq	1.5, 4.5 and 9 mm Axial and Trans-axial Various speeds Individual limb motion Ongoing heart beat and breathing

Table 3 Summary of the 3 imaging studies.

Both rat studies used a male Sprague Dawley rat weighting approximately 250g. Figure 64 shows a collection of photographs taken during the Rat 2 imaging session, principally for the interest of the reader. Of note, however, is the one including the real time preview of the attenuation correction CT scan showing a projection of the approximate field of view. During the CT acquisition this shows each projection, and the motion of the diagram with breathing was immediately apparent, meaning that there should be a significant contribution from this motion.

Note that in all cases the times of the motion are only known approximately. The animal was moved when the scanner's software indicated that the relevant time in the acquisition had arrived. However, it appears that this clock displays the timecode received from the scanner, and is necessarily 'behind' – and not insignificantly so, given the need to partially decode the list mode data to extract the time and displaying this time is, understandably, not the top priority for the software. Further variability is introduced by the fact that the motion is done by hand in response to the time being displayed. ***As a guide, an error of a second, and generally a delay, would not be unexpected.***

After the twitch detection was completed using the eventwise formulation the likelihood plots were resampled to a uniform 100 Hz sampling to make handling the data easier.

2. Results

The figures on the following pages show the likelihood plot for the two rat studies for six timescales, with the intended time of twitches and corresponding window widths before and after, accounting for decay correction, indicated by shading. Figure 65 through Figure 70 are for the Rat1 study, ad the Rat2 results are showing in Figure 71 through Figure 76. Various aspects of these figures will be discussed by the following subsections.

Note that the progressive offset in the start time on the plots is caused by the need to 'preload' the before and after sinograms, and the queues of events waiting to transition and exit, before the marginal effect of any individual counts can be computed. At present the implementation does not handle this properly, so the queues have to fill through before the likelihood plot is valid. These sections are cropped out.

2.1 Discussion of Rat1 Results

The times of the twitches correspond to obvious disturbances on the likelihood plot. That said, at some timescales the 4th motion is hard to distinguish by eye, at least compared to the fluctuation of the likelihood plot in the surrounding inter-twitch regions.

The axial motions have produced larger shifts in the 'baseline' likelihood than have the trans-axial motions. This is not surprising, as an axial motion changes the content of the field of view, whereas a trans-axial motion only alters the position of the same content. (Recall that by changing the position within the field of view the number of bins on the sinogram that count emissions from the subject will change, and so there will still be some change in baseline likelihood – see Figure 43 on page 126.)

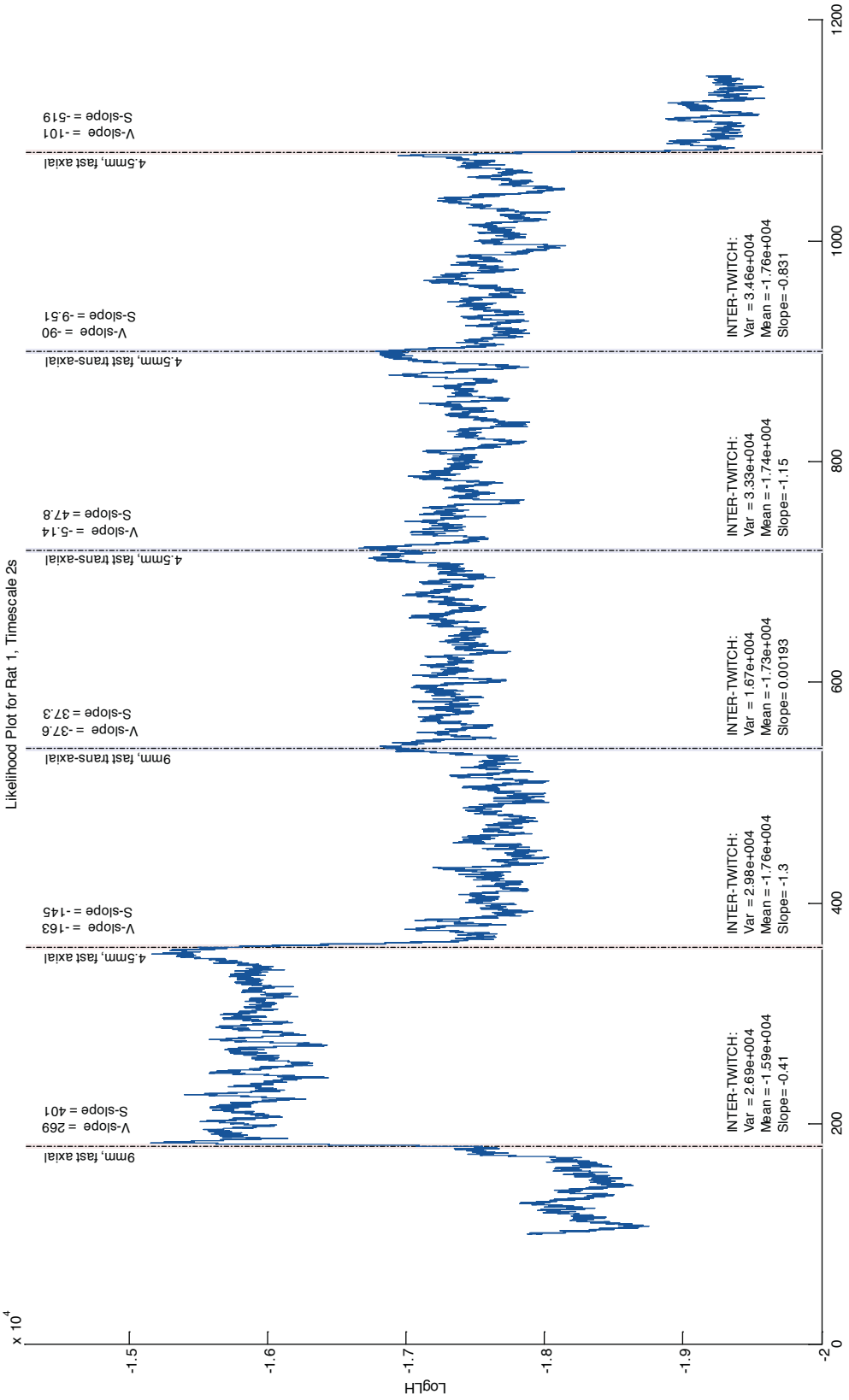

Figure 65 Likelihood plot for the Rat1 study with a timescale of 2 seconds.

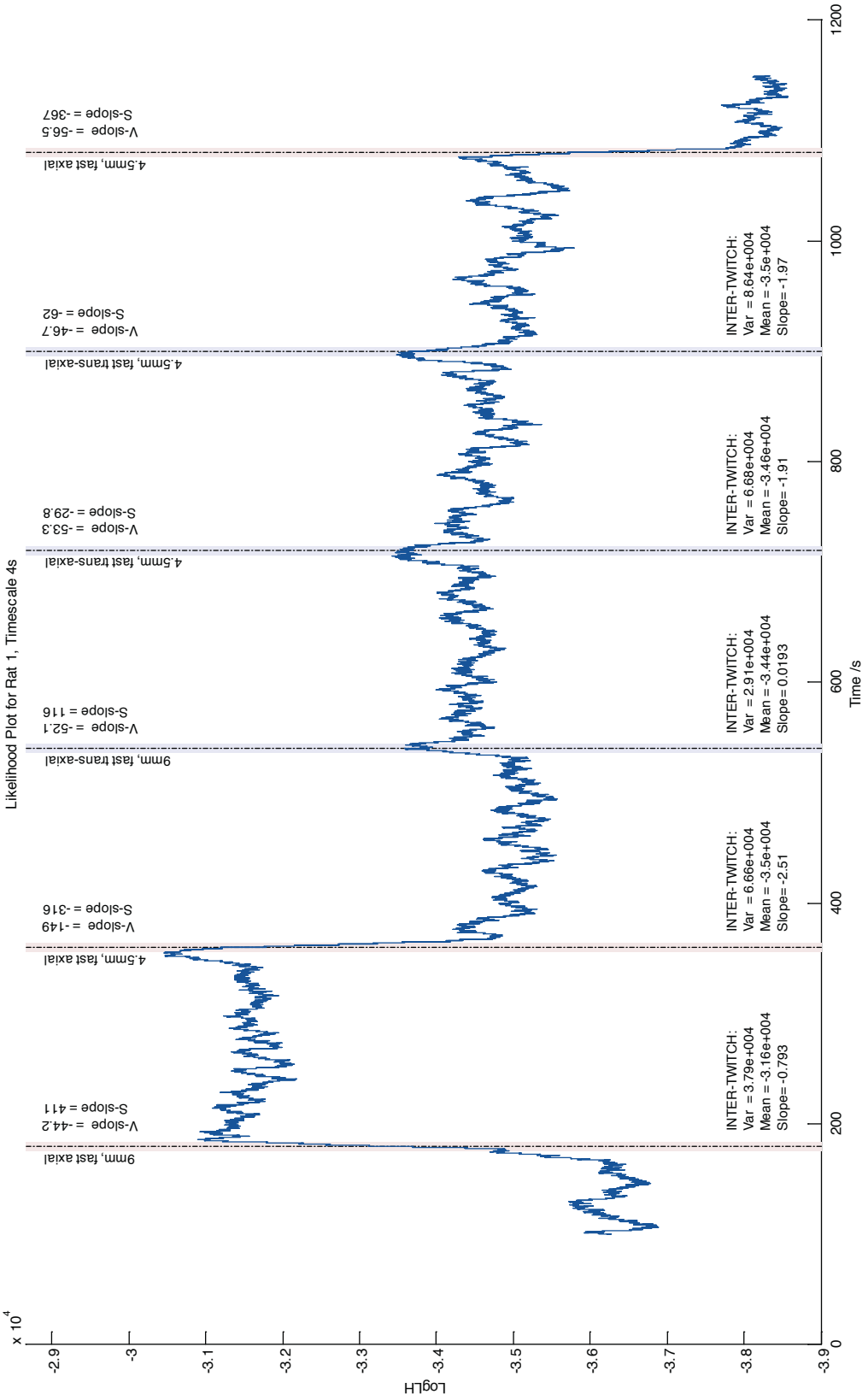

Figure 66 Likelihood plot for the Rat1 study with a timescale of 4 seconds.

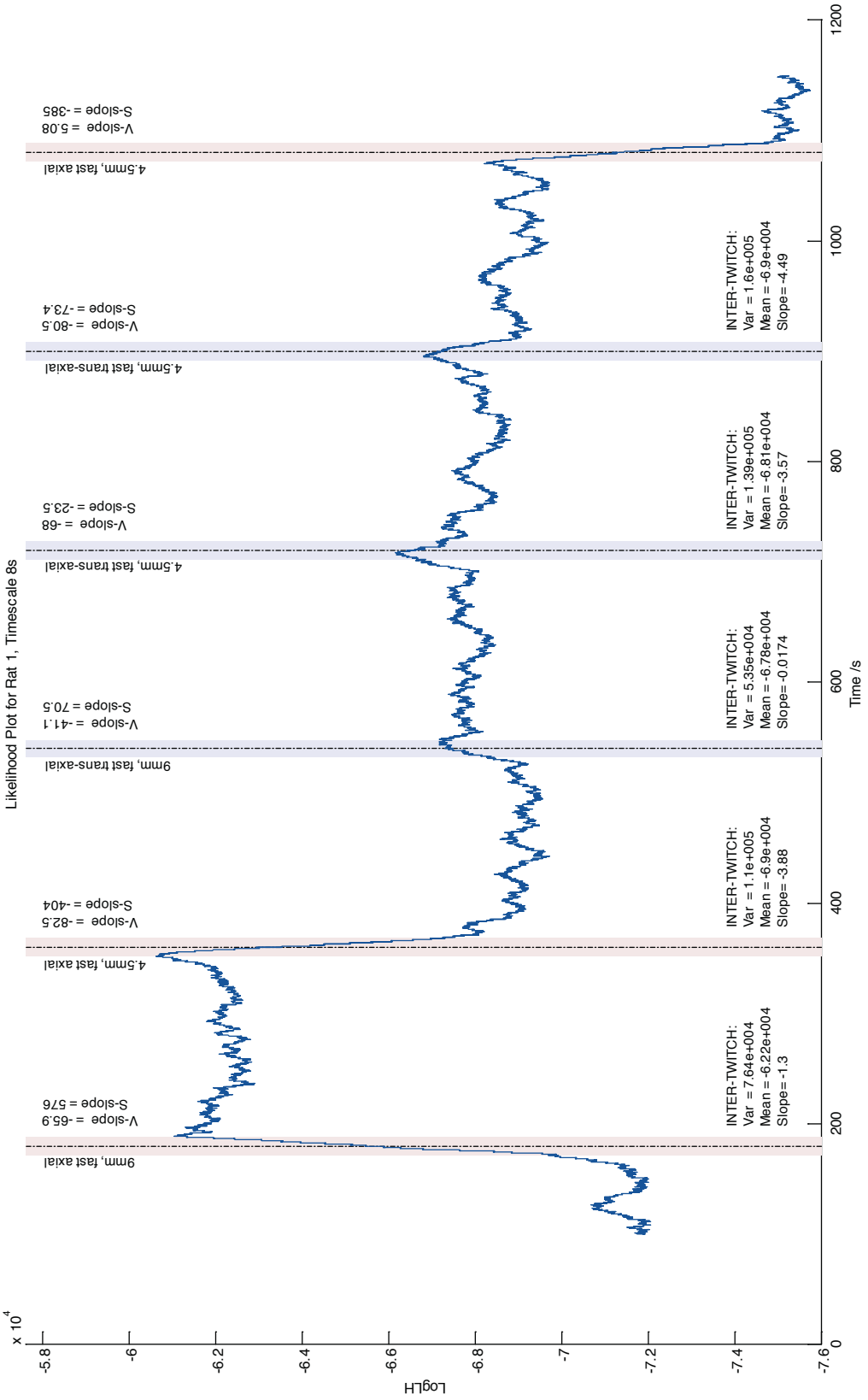

Figure 67 Likelihood plot for the Rat1 study with a timescale of 8 seconds.

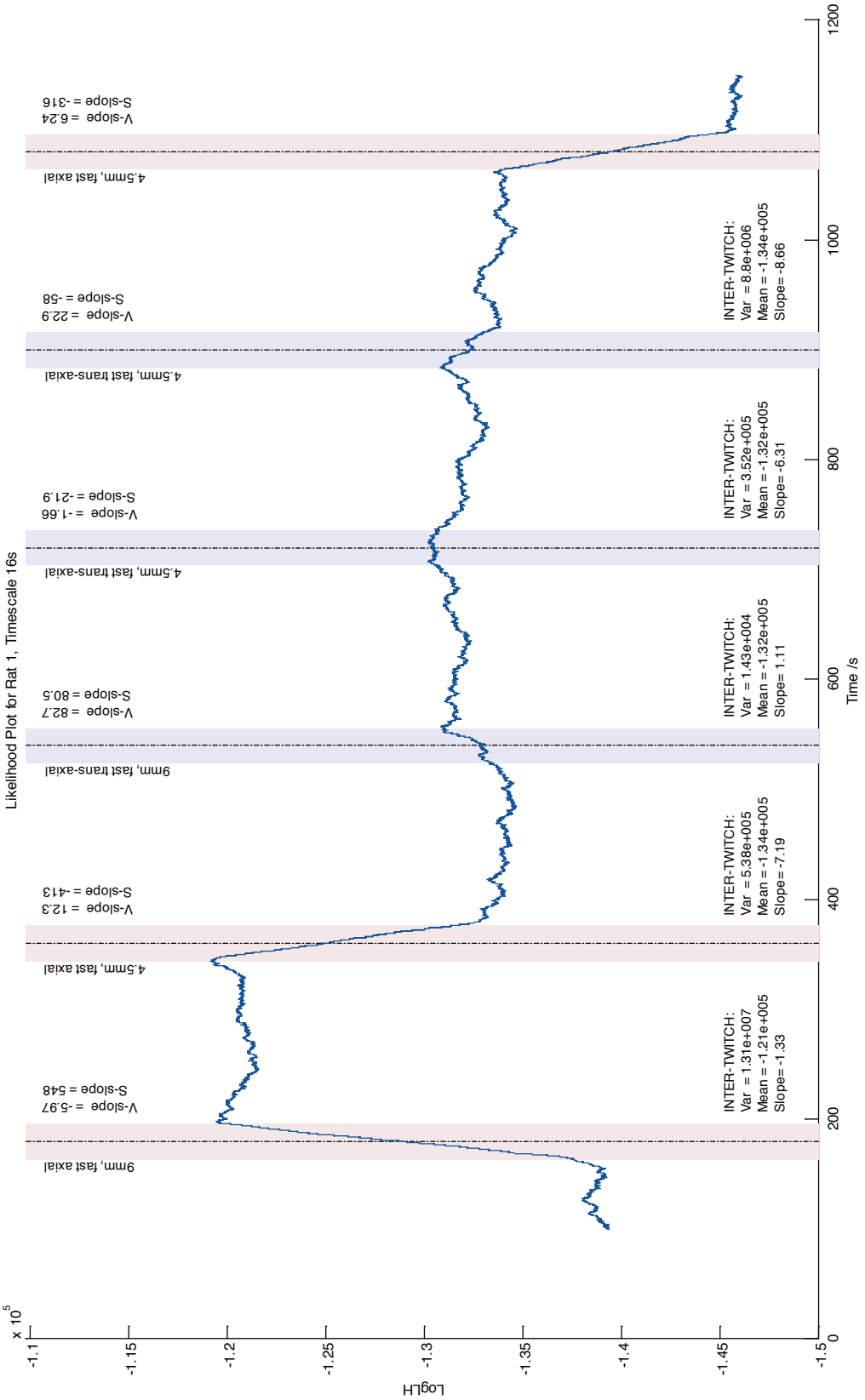

Figure 68 Likelihood plot for the Rat1 study with a timescale of 16 seconds.

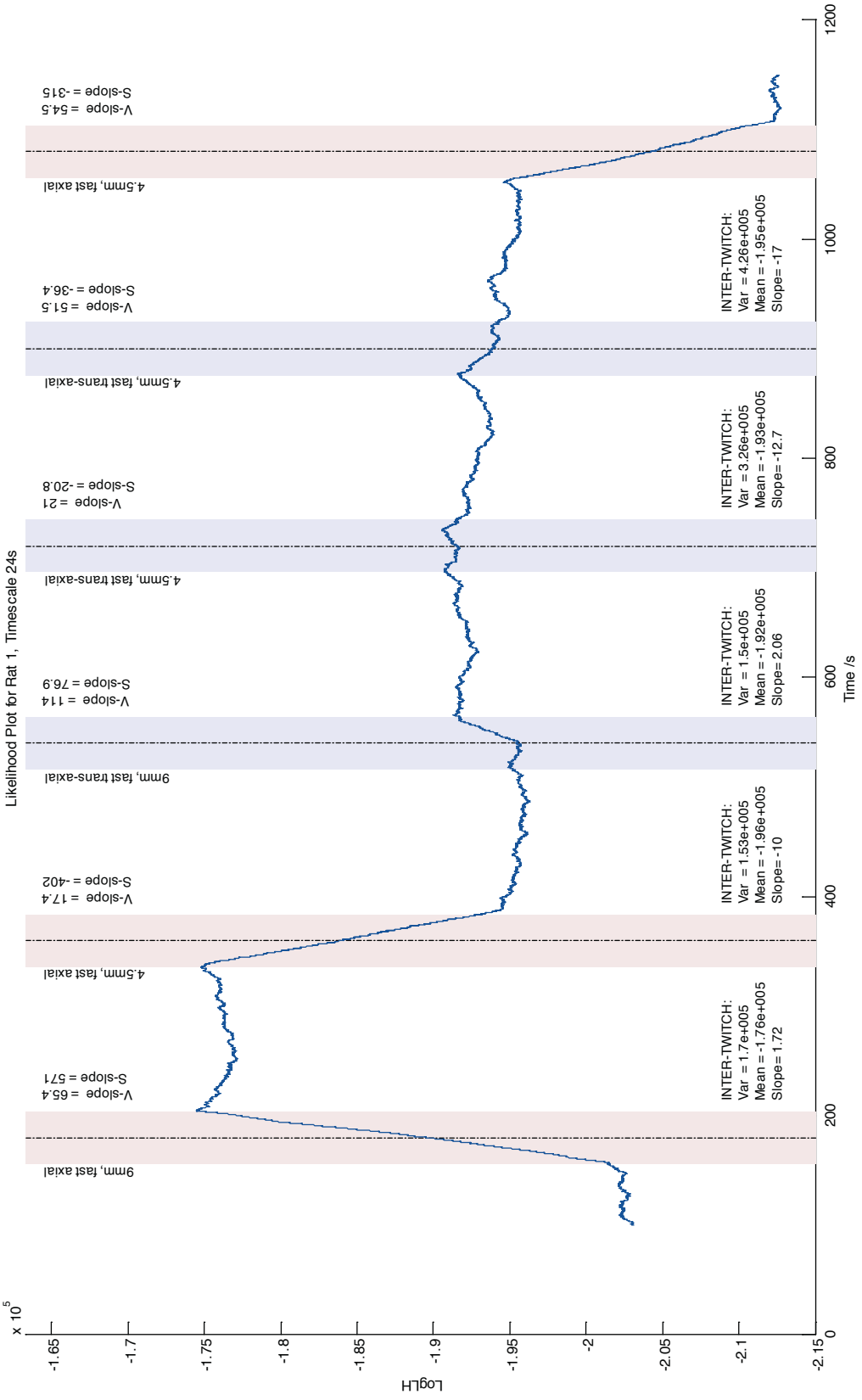

Figure 69 Likelihood plot for the Rat1 study with a timescale of 24 seconds.

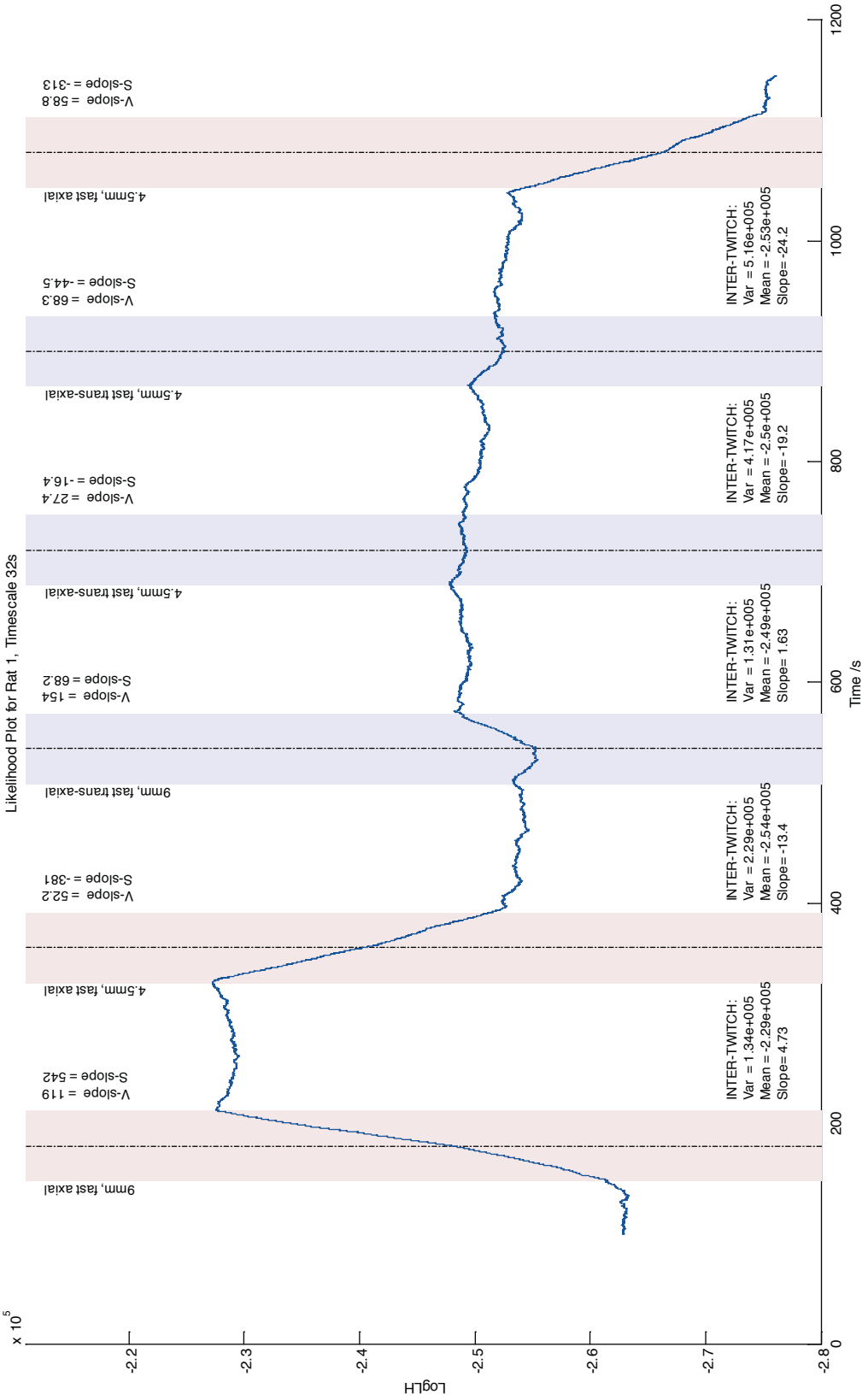

Figure 70 Likelihood plot for the Rat1 study with a timescale of 32 seconds.

The larger motions seem to have produced more pronounced effects as well, although the relationship is neither linear nor unequivocal. Rat2 contains a greater range of motions and thus provides a better test of this hypothesis.

2.2 Discussion of Rat2 Results

Again the motions produce disruptions in the likelihood plot corresponding to the slope plus `V' pattern as predicted. However, in this study, the smallest motions – 1.5mm and the pull of a leg – produce barely discernible changes. However, this is to be expected since smaller motions approach the spatial resolution of the scanner. Also, there is a clear trend for larger motions to produce bigger effects on the likelihood plot. The tendency for axial motions to produce bigger shifts in the baseline likelihood is still apparent as well. In short, there are trends in the variation of twitch manifestations that are as expected.

There does not seem to be an obvious difference between fast and slow motions. At larger timescales this is not unexpected, as the difference between a 'slow' and 'fast' motion is negligible compared to the timescale of the twitch detector. At finer timescales, the size of the twitch-related features starts to become swamped by the ongoing fluctuation of the likelihood.

The twitch at 1260 seconds is problematic as it appears to cause a spike in likelihood, especially at finer timescales. Larger timescales reveal, by making the size of the twitch features larger, that there seems to be some increase in the likelihood before the twitch. The obvious conclusion is that there was something in the way that motion was performed, specifically right before the motion, that has affected the likelihood.

> *The downward slope in these plots warrants further discussion in its own section.*

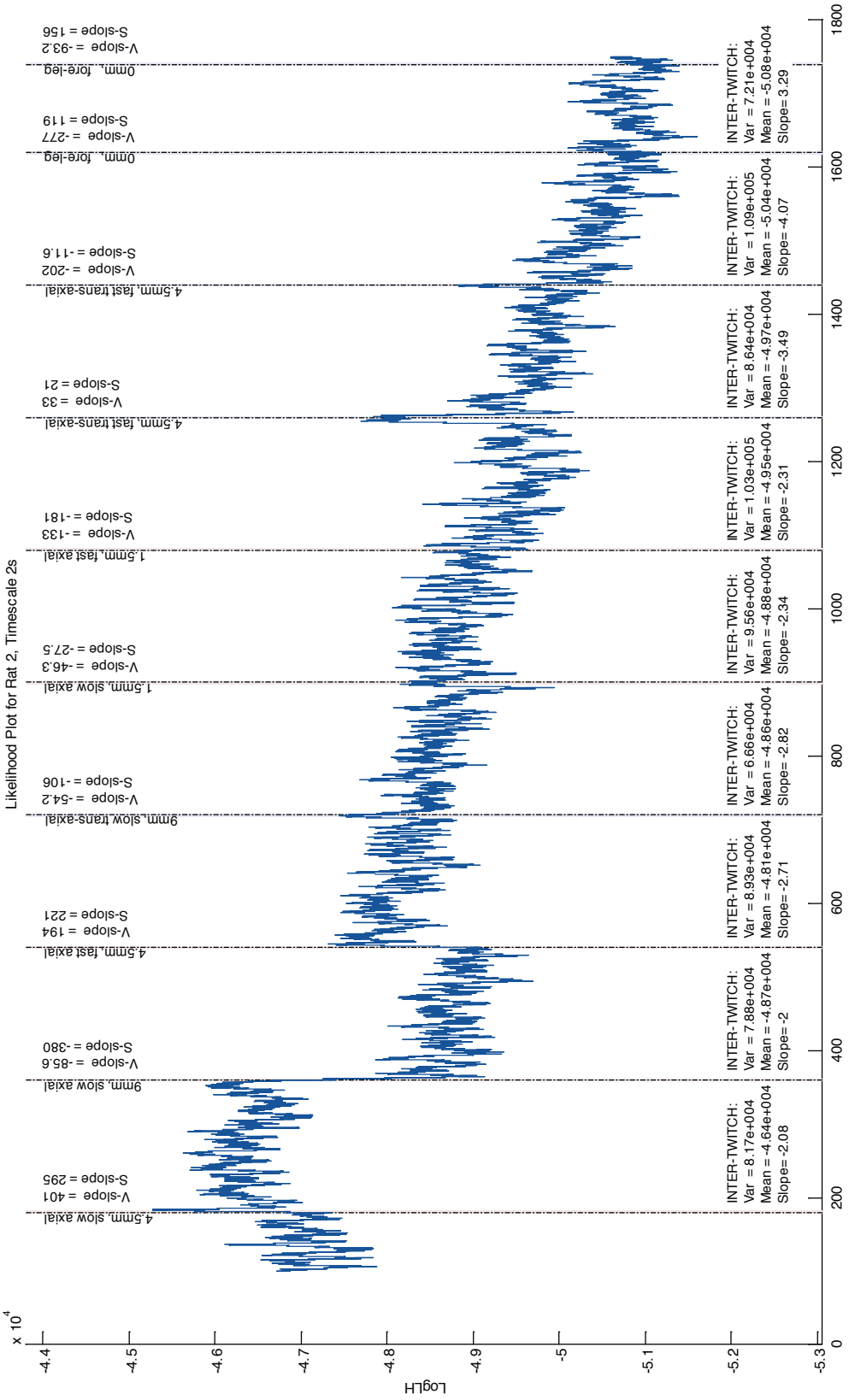

Figure 71 Likelihood plot for the Rat2 study with a timescale of 2 seconds.

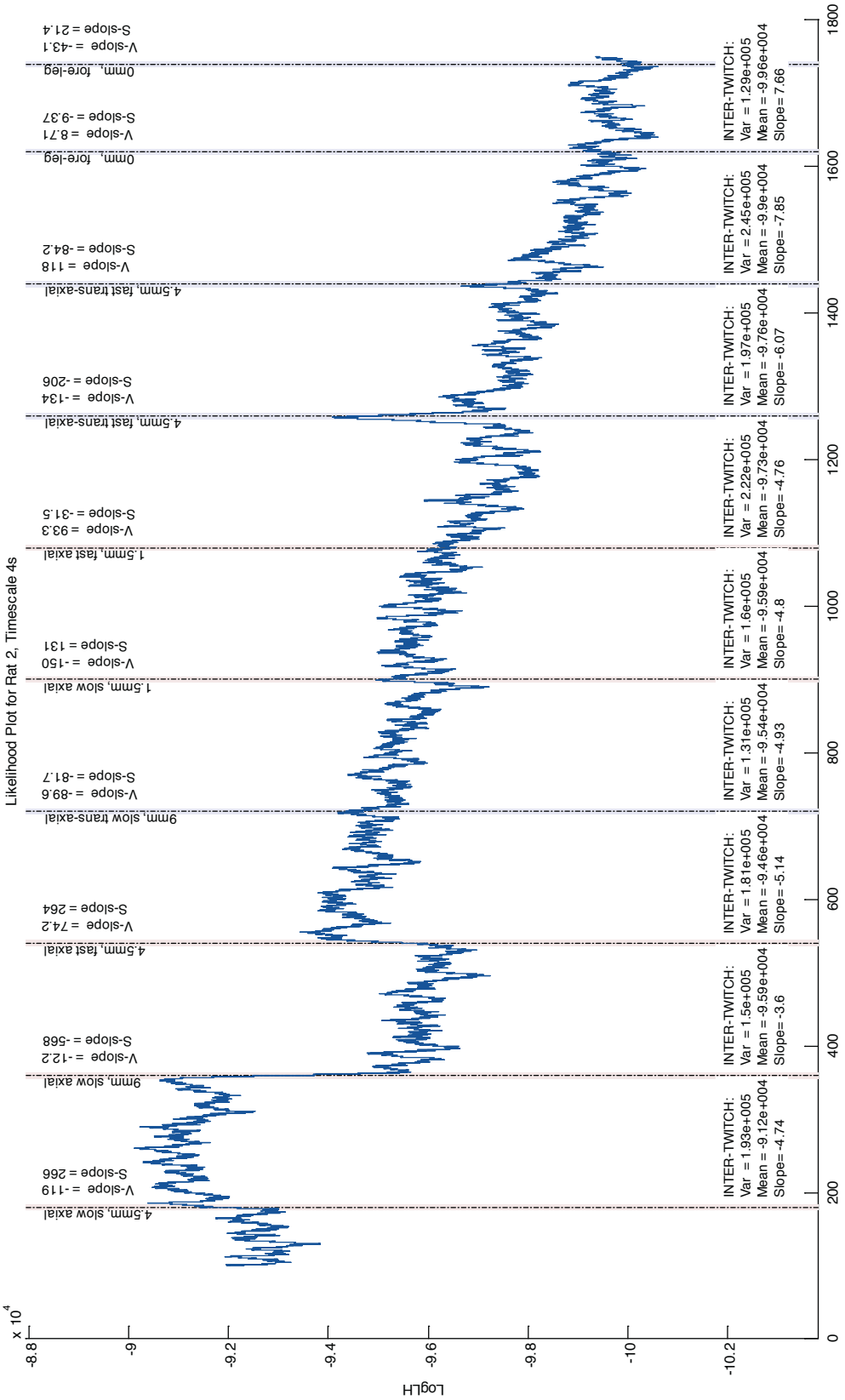

Figure 72 Likelihood plot for the Rat2 study with a timescale of 4 seconds.

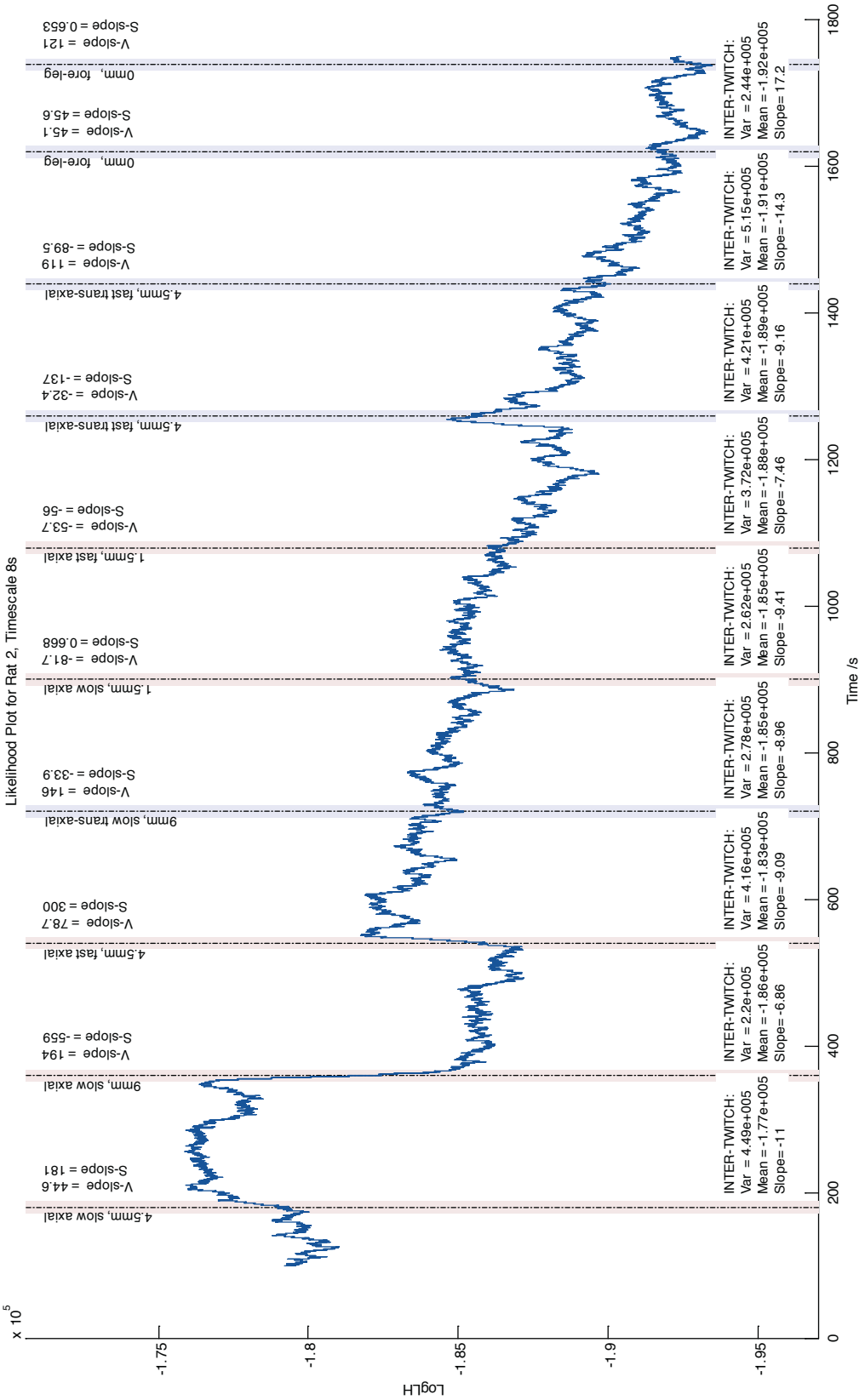

Figure 73 Likelihood plot for the Rat2 study with a timescale of 8 seconds.

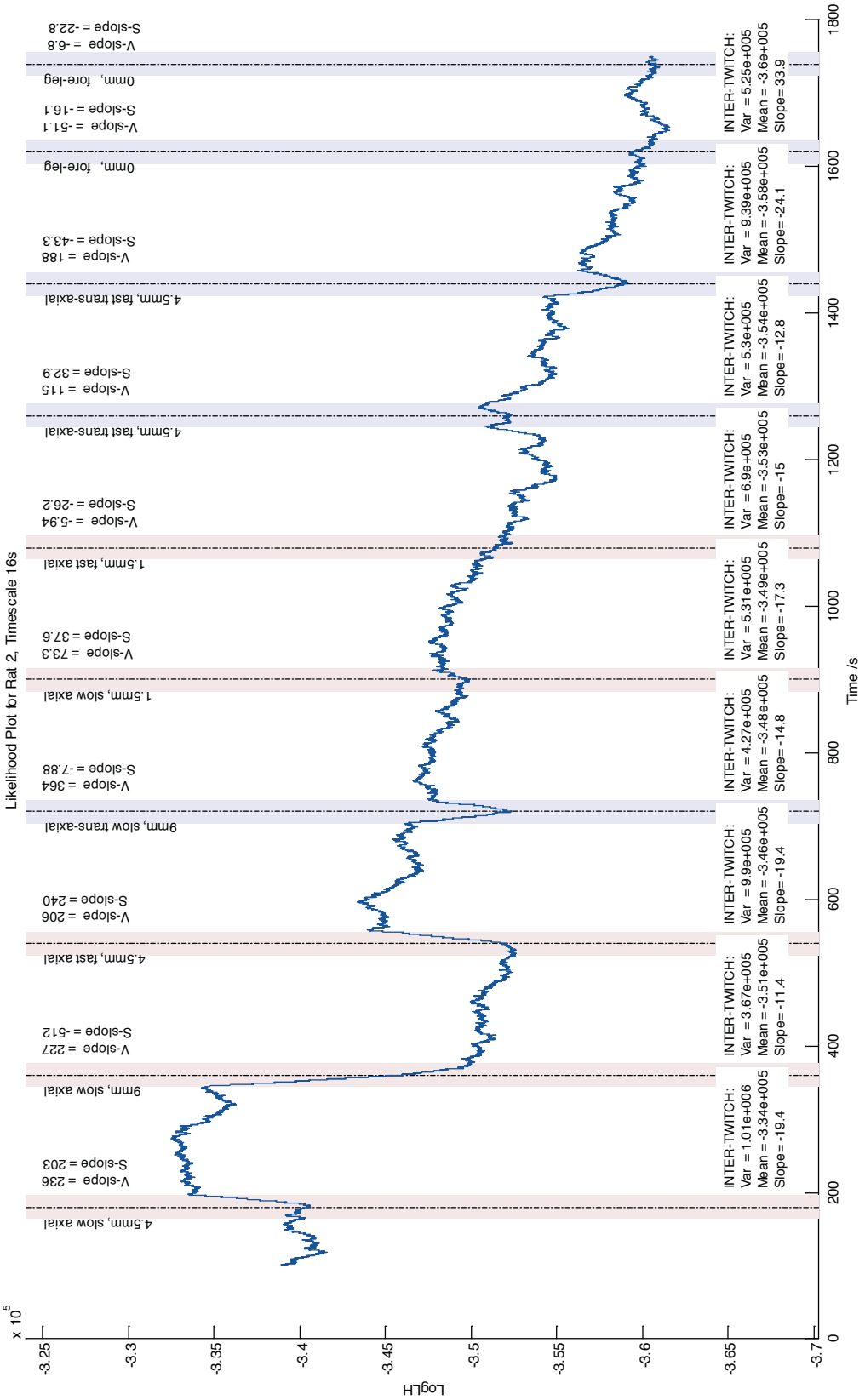

Figure 74 Likelihood plot for the Rat2 study with a timescale of 16 seconds.

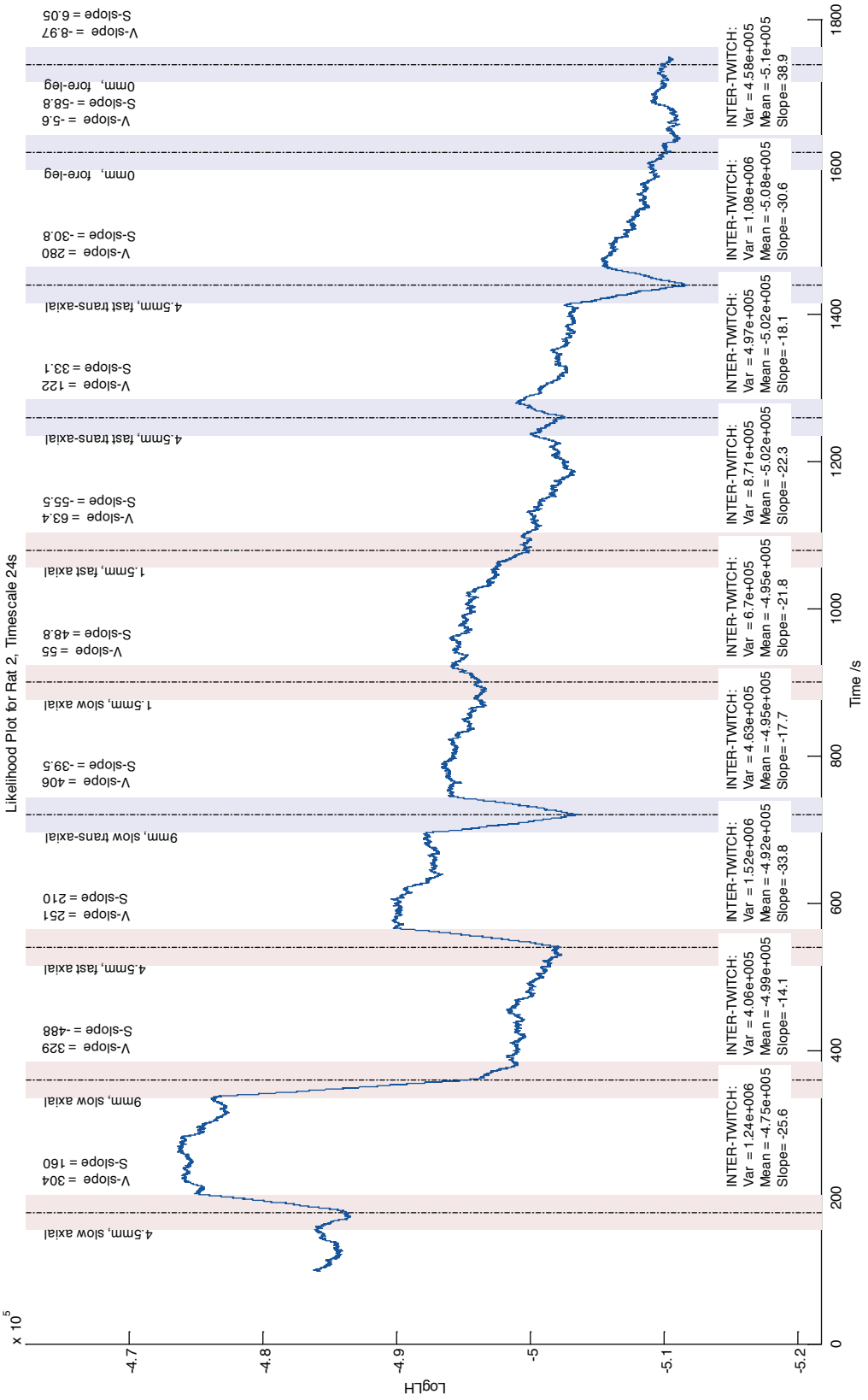

Figure 75 Likelihood plot for the Rat2 study with a timescale of 24 seconds.

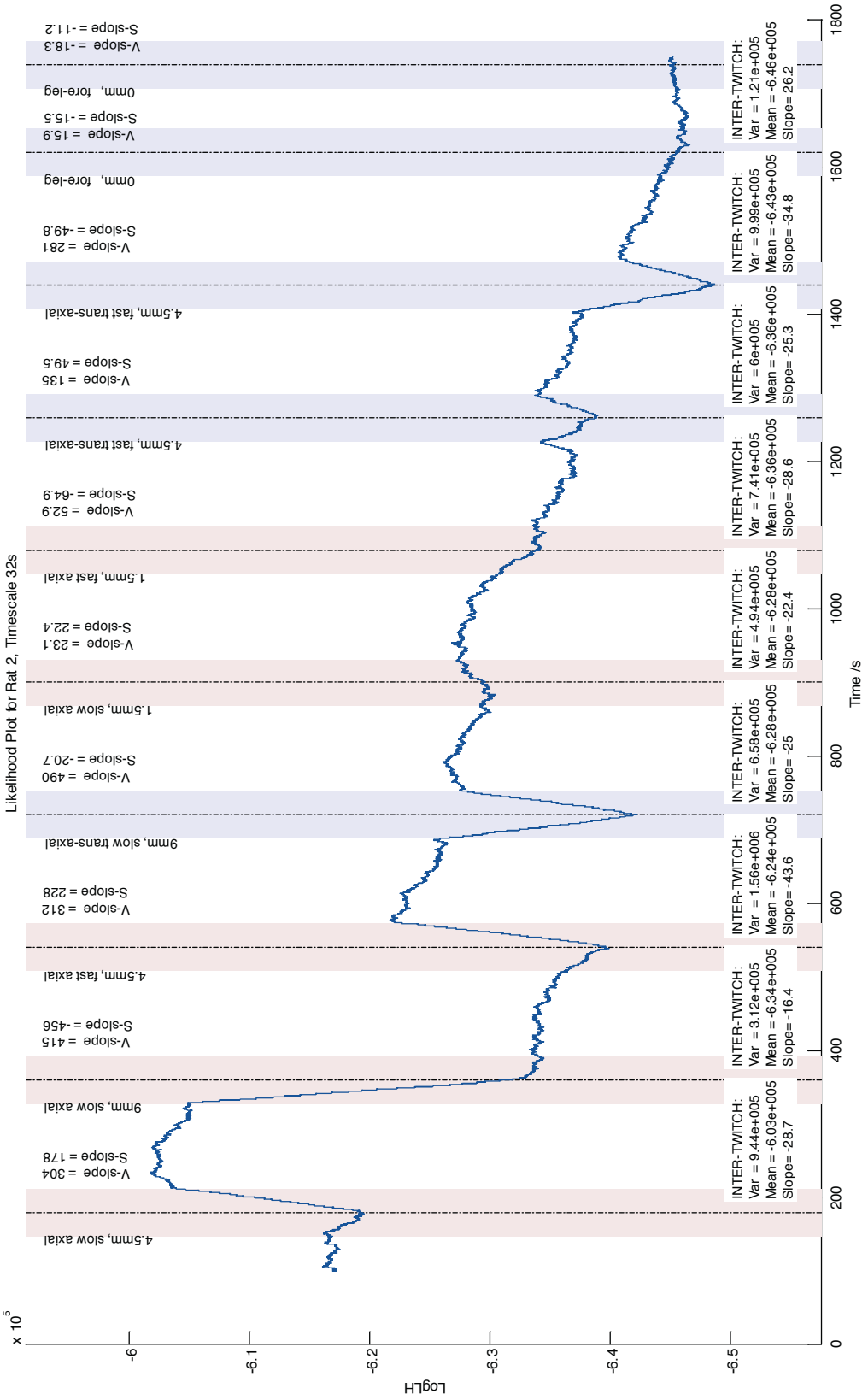

Figure 76 Likelihood plot for the Rat2 study with a timescale of 32 seconds.

2.3 Downward slope in Rat2

There is a very clear downward slope to the Rat2 study, present at all timescales. This was unexpected, and was hard to explain. The phenomenon is not present in the Rat1 study, suggesting that the effect may be a result on some ongoing physiological process. As the tracer, FDG, was administered an hour before the start of the scan, it is most likely some kind of slow 'washout' process; FDG would have reached its steady state distribution by the start of imaging.

Before decay correction was implemented, the likelihood plot exhibited an ***upwards*** drift – see Figure 63. Moreover, a drift up towards zero would be the expected result of a falling total number of counts, consider the expression for the log likelihood of each bin:

$$\log\left(\frac{\mu^{s_b+s_a}e^{-2\mu}}{s_b!s_a!}\right) \tag{46}$$

As the expected value of s_b and s_a for all bins on the sinogram falls the expected value of the likelihood plot will rise towards zero. The implication of this is that one possible cause of the slope is a migration of tracer into the field of view, perhaps to the liver to be metabolised?

2.3.1 Bulk Tracer Migration

To assess this possibility, the count rate was measured by repeatedly histogramming several short sections of data from both the Rat1 and Rat2 studies and counting the number of events recorded. This is shown in Figure 77.

Ten minutes of each study were used, from 300 to 900 seconds, sampled every 10 seconds, with nominally 10 seconds after each sample point histogrammed – although note that for the decay corrected lines the actual window width will vary (from 9.565

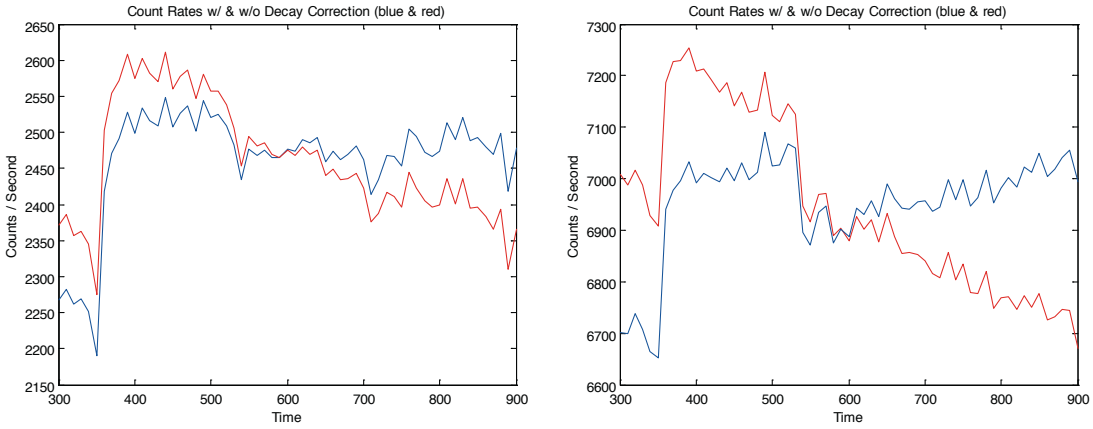

Figure 77 Measured count rates for the Rat1 (left) and Rat2 (right) studies with (blue) and without (red) decay correction. The downward slope on both uncorrected plots is expected as a result of the tracer's decay. The Rat1 plot shows that decay correction is working, as the blue line seems flat and the crossover is halfway through the period. In Rat2 there is some other process that actually causes the decay corrected count rate to increase, the focus of discussion in the text. (Sudden jumps are from motions.)

seconds to 10.462). The count rate is calculated by dividing the total number of counts in the histogram by the nominal window width (10 seconds).

Note also that there may be jumps as a result of the motions, and these will not exactly match the times of the twitches because the data histogramed at a given time is different from that implicitly histogramed for calculating the likelihood plot.

For Rat2 it seems that there is some steady net migration of activity into the field of view, after decay has been accounted for (the blue line drifts upwards). Reassuringly, there seems to be no drift in the corrected count rate for Rat1, which confirms that decay correction is working.

2.3.2 Sinogram Polarisation

Another possible explanation is that there is some other migration of tracer that changes the distribution of activity so as to continually decrease the likelihood – for example, it might be that the slow washout causes the tracer concentration to fall faster in 'hot-spots' and thus the tracer density becomes less 'polarised'. Consider a scenario in which

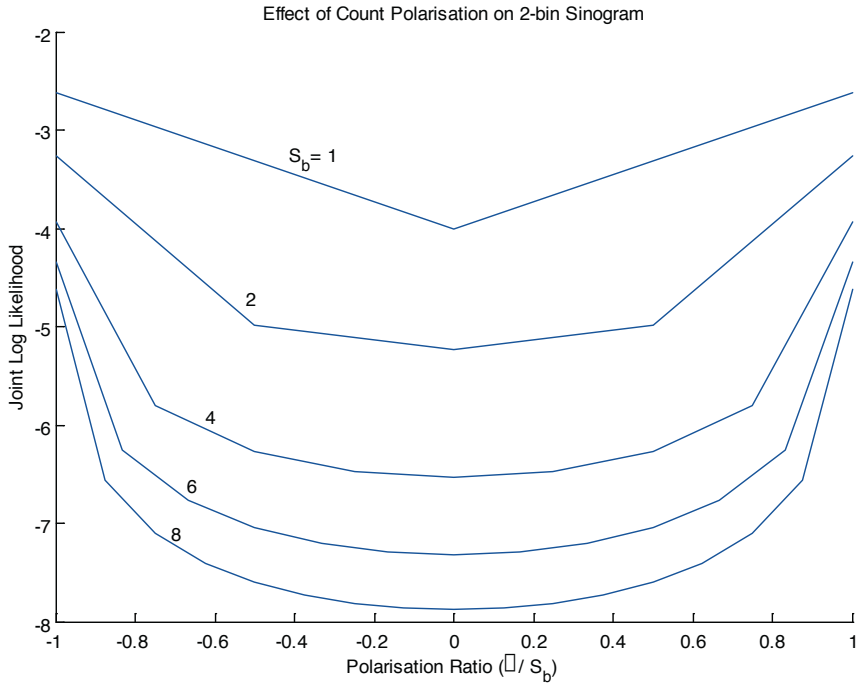

Figure 78 Likelihood plot values for a 2-bin sinogram as the ratio of counts between the two bins varies. See text for details.

the sinogram binning is such that there are only two bins. In this case the likelihood plot's value is given by:

$$\log\left(\frac{(\mu-\gamma)^{s_b-\epsilon_b+s_a-\epsilon_a}e^{-2(\mu-\gamma)}}{(s_b-\epsilon_b)!(s_a-\epsilon_a)!}\right)+\log\left(\frac{(\mu+\gamma)^{s_b+\epsilon_b+s_a\epsilon_a}e^{-2(\mu+\gamma)}}{(s_b+\epsilon_b)!(s_a+\epsilon_a)!}\right) \quad (47)$$

Where the differences between the counts between each of the two sinogram bins, for the before and after windows, is $2\varepsilon_b$ and $2\varepsilon_a$. Following from the definition of μ, it is clear that $\gamma = 1/2\ (\varepsilon_b+\varepsilon_a)$.

Restricting the discussion to one scenario (which suffices to prove the point), let $\varepsilon_b = \varepsilon_a = \gamma$, and $s_b = s_a$. In other words, there is no change in counts between the windows, and a conservation of counts as the ε terms are changed: the only change is how unevenly, or 'polarised', the distribution of counts between the two bins is.

The joint likelihood – i.e. the value of the likelihood plot – of this two-bin sinogram is plotted against the ratio of ε_b to s_b in Figure 78, for several s_a (and therefore s_a). When

the ratio is -1 all the counts are in one bin, when it is 1 all the counts are in the other bin, and 0 means there is an even split between both bins on the sinogram. Note that as the bin counts can only be integer values there are more valid ratios when the total number of counts is greater.

From the plot it is clear that the likelihood is higher in the extreme cases, when the counts are more concentrated. Therefore it is possible that a general equalisation of tracer concentration during washout might cause the likelihood value to fall as the scan progresses. (Also, the 'stacking' of the curves is a result of the fact that higher count numbers cause lower likelihoods, already known from considering the effect of decay correction, for example.)

2.3.2.1 Testing the Polarization Hypothesis

To test this phenomenon with the actual data, the number of counts in the bins have been histogrammed for the sinograms generated for the count rate plots above. This allows the change in the distribution of counts between bins with no counts in, 1 count in, 2 counts in, etc. to be inspected in Figure 79 and Figure 80.

These plots show, by the colour, the log-number of bins observing the number of counts on the x-axis, for a nominal 10 second sinogram at the time on the y-axis. The log-number of bins is used to compress the range of the values as the 0-count bins are far more common than any others. Should the rows of the figures change it would indicate that the proportions of bins with various numbers of counts in is changing over time: hot spots washing out first would reduce the number of bins with many counts in, so the value (colour) of columns to the right hand side of the plot would fall off over time (down the figure).

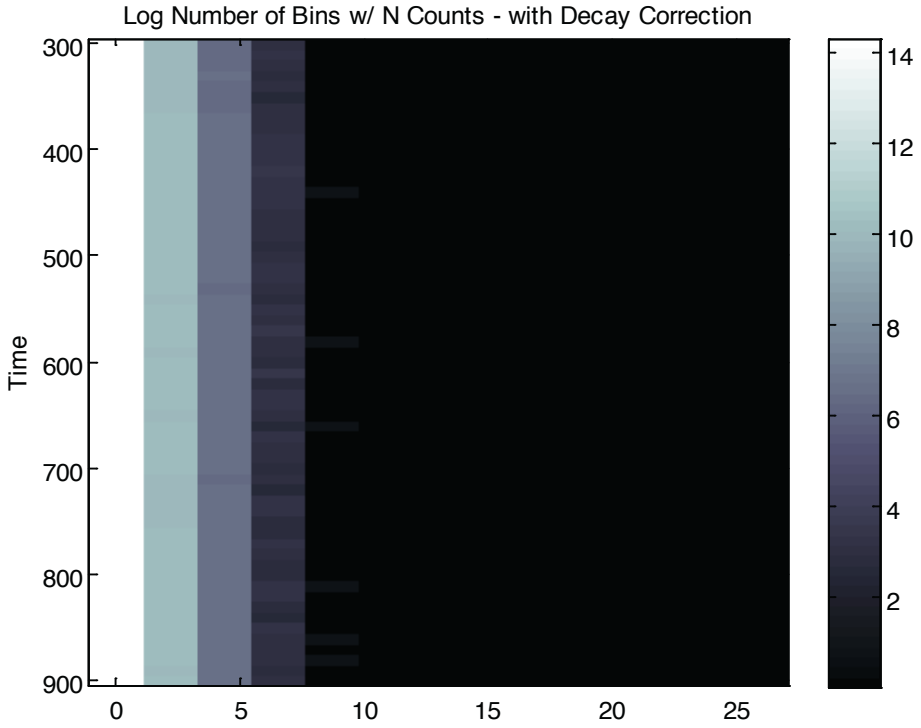

Figure 79 Variation of the log-number of bins (colour) observing a given number of counts (x-axis) for nominal 10 second sinograms – with decay correction – at various times (y-axis) for Rat1.

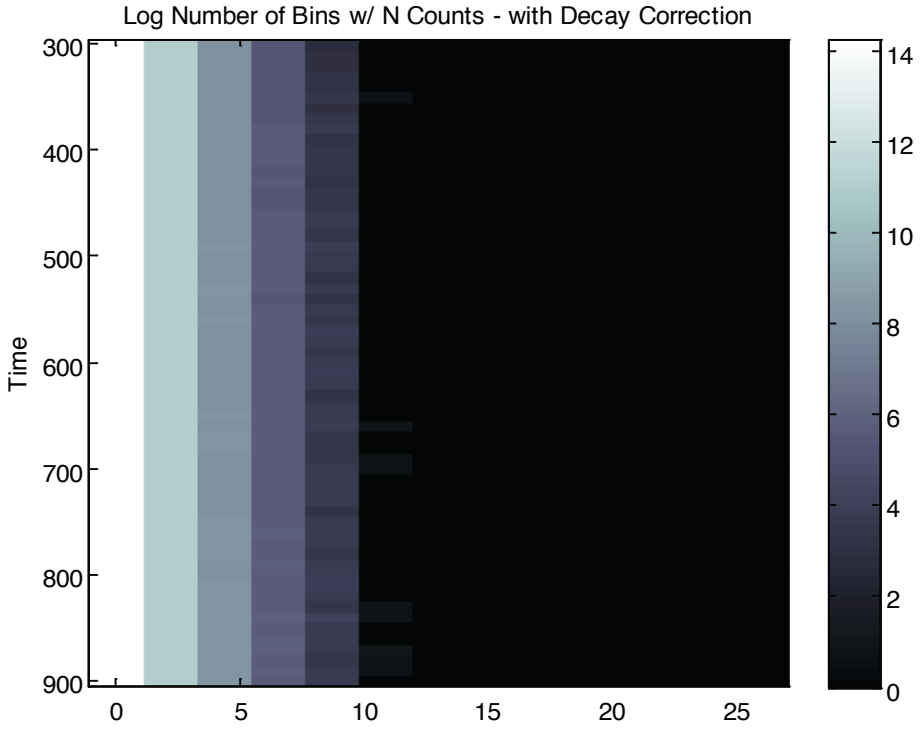

Figure 80 Variation of the log-number of bins (colour) observing a given number of counts (x-axis) for nominal 10 second sinograms – with decay correction – at various times (y-axis) for Rat2.

It appears that there is no obvious evolution in the distribution of counts over time, nor is there any obvious difference (in terms of that evolution) between Rat1 and Rat2. However, it is not clear how much of a change in the distribution would be significant, so this explanation cannot be excluded completely – note also that there is no obvious change of the histogram from the decay. (In Rat2, however, bins with more counts in do occur more frequently than in Rat1, most obviously as a result of the increase in the injected dose.)

2.3.3 Conclusions

It seems that the first explanation (migration of activity in to the field of view) is the most promising, although the second cannot be excluded based on the evidence available above. Further there may be yet other possible explanations. In terms of a plan as to how to explore this, the first step would be to try and predict the slope of the drift from the variation in count rate. This would provide a means to correct it – flattening the likelihood plot. From there it would be possible to study how much drift, if any remains to be explained by other means. Note that if the migration of tracer is physiological the slope on the likelihood plot will vary with the rate of tracer movement, which may be non-constant depending on the kinetics of the tracer in question.

2.4 Cross Validation

Figure 81 compares the likelihood plot generated using the event-wise implementation to two measures based on the reconstructions that result from explicit histogramming and reconstruction – i.e. cross validating it against equivalent image domain approaches. All approaches account for decay correction.

The image domain approaches require an explicit choice of temporal sampling, this had to be kept coarse because of the computational cost of computing each time point.

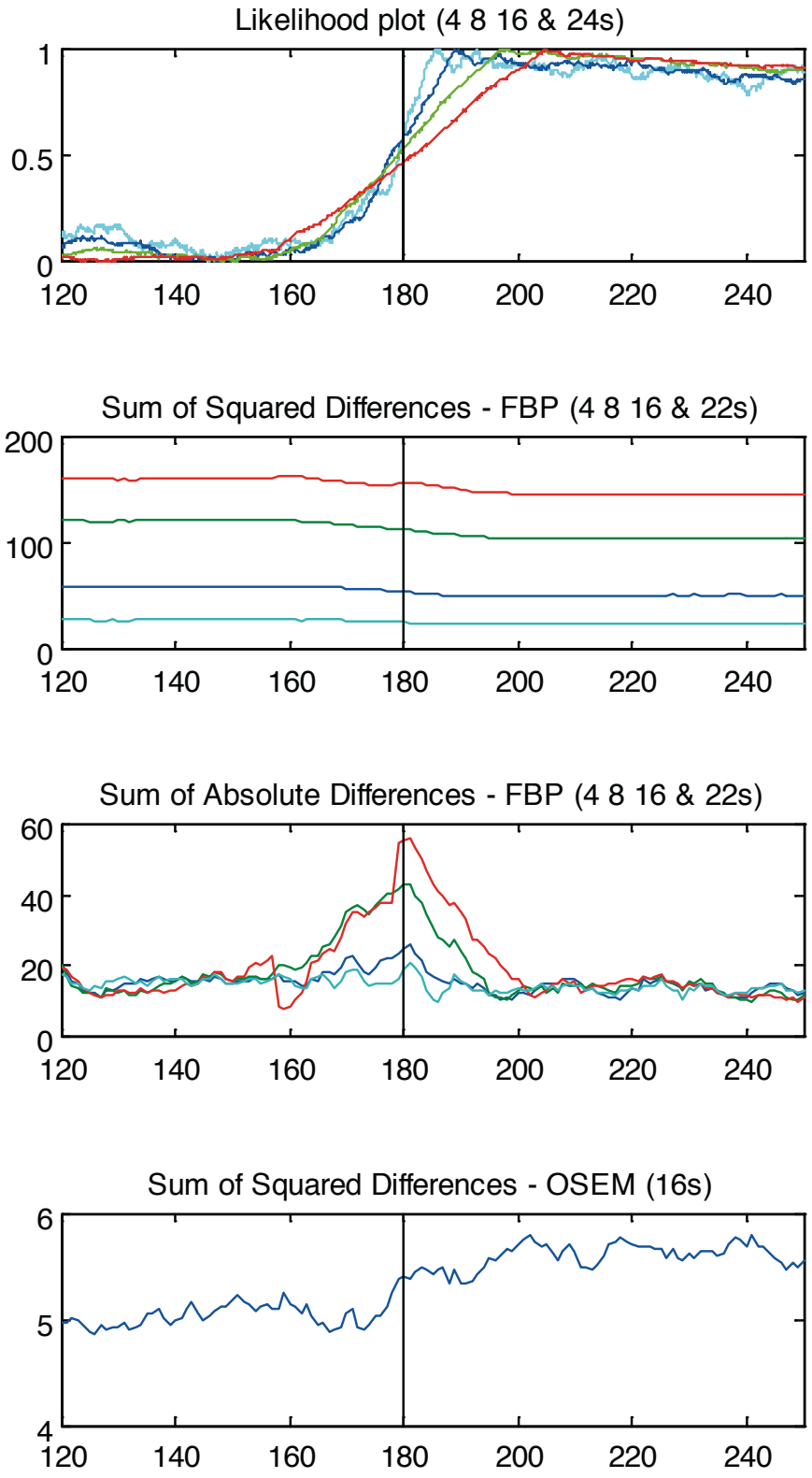

Figure 81 Cross-validation of the likelihood plot against image domain alternatives. From top: Likelihood plot, as generated by the eventwise implementation; Sum of Squared Differences between FBP reconstructions of neighbouring sections of data following explicit histogramming, computed every second; Sum of Absolute Differences, computed likewise; Sum of Squared Differences between OSEM reconstructions. All approaches account for decay correction. The timescales used are 4 seconds (cyan), 8 seconds (blue), 16 seconds (green, except for the bottom plot), and 22 seconds (red, or 24 seconds in the top plot).

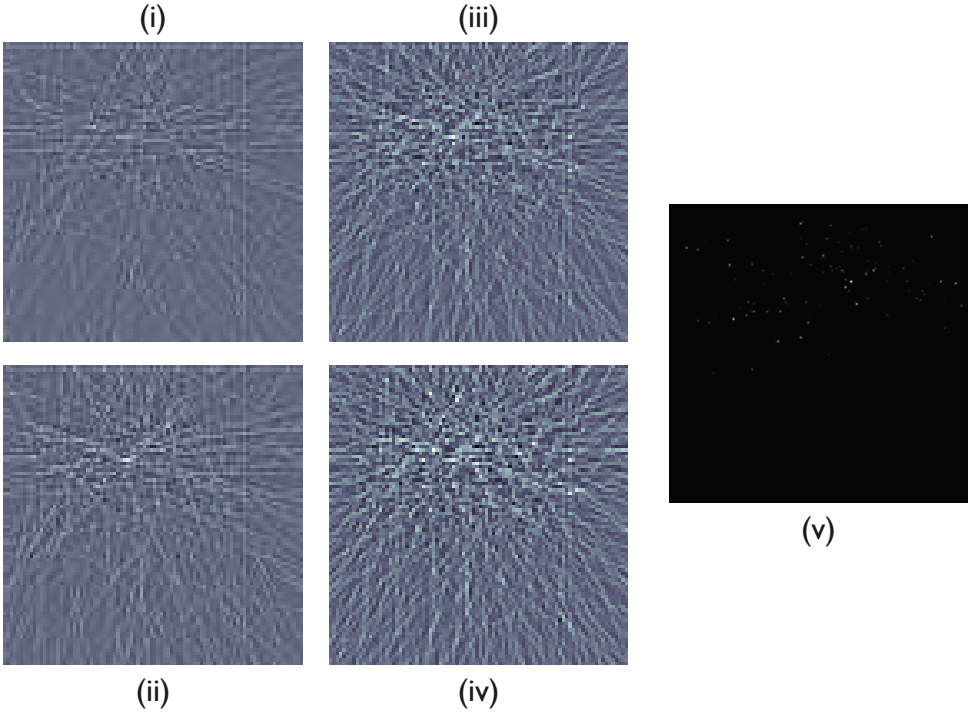

Figure 82 Sample reconstructions for the cross validation plot. (i) from 4 seconds of list mode data, by filtered back projection, (ii) 8 second, FBP, (iii) 16 seconds, FBP, (iv) 22 seconds FBP. (v) shows 16 seconds reconstructed by OSEM. There is almost no image is discernible in all of them, but in (v) the non-negativity constraint of OSEM reconstruction makes the image a handful of points. The rat should be in the upper half of the images.

In contrast, the event-wise likelihood plot is at the same temporal resolution as the list mode data.

The data was from the Rat 1 study; a fast motion of 9mm, along the bore. Reconstruction was done by filtered back projection, with the exception of one experiment that used OSEM as implemented for Matlab in [23]. Note that for the OSEM case only the central 40 cross-sections (of a total of 80) were reconstructed to try to save time. The FBP experiments used data from all 80, reconstructed as a stack of 2D images. The event-wise implementation likewise uses all the sinogram bins from all 80 slices.

Figure 82 shows sample reconstructions. OSEM (strictly the MLEM algorithm) enforces a non-negativity constraint and so produces much less noise, especially in the background. As a result the image for the OSEM reconstruction is very different from the FBP case – the 'image' is a few speckles.

Note that the sum of squared differences appears to produce a ramp feature and the sum of absolute differences an inverted 'V'. The implication of this is that the sum of squared differences is sensitive to the animal's position in the scanner (so detects the change in baseline) and the sum of absolute differences picks up on the change in position.

This could be explained by the nonlinearity of sum of squared differences, which would emphasise the amplitude of the variation due to noise more than sum of absolute differences would. The noise power is be expected to be greater when the animal is in some positions rather than others due to the nonuniform sensitivity of the scanner across the field of view. However the extent of the difference between the two is still a little surprising, and warrants further (time consuming!) investigation.

2.5 2-stage Localisation

In order to display the results of the 2-stage localisation method, the localisation has been run on for each twitch. First, the likelihood plot is re-sampled so that it has an even temporal sampling of 0.01 seconds. This is because the implementation of the localisation algorithm was much simpler with regularly spaced data. Then, the likelihood plot is cropped to a region of 70 seconds either side of the known twitch time (except, sometimes, for the first and last twitches when there was not enough extra likelihood plot). That was done because the 2-stage method is limited to finding a single twitch

2.5.1 Rat1 Results

The results are presented in Table 5 and Table 6, below. These show the estimated twitch time from the first and second stage respectively, and the error of the second stage estimate (estimate - ground truth) – remember that the 'true' twitch time is actually the ***intended*** twitch time, and a delay between the intention and the actual movement of up to a second is reasonable. The penultimate column states if the second stage, the

model fitting, helped improve the estimate. The final column classifies the localisation as a pass/fail, failure defined as either stage having an error greater than the timescale of the twitch detector plus one second, because at that point the second stage cannot be working with any feature resulting from the twitch (the plus one is a little grace to allow for the possible delay in executing the motion after the intended time is reached).

As vast tables of numbers are not, perhaps, the best way to get a feel for the results, they are also summarised visually in Figure 83.

2.5.2 Rat2 Results

Table 7, Table 8, and Figure 84 are as for the Rat1 results. Note that the final twitch (second fore-leg pull) is too close to the end of the data to run the localisation fairly, so it has not been considered.

2.5.3 Discussion

The 2-stage localisation method successfully demonstrates how the likelihood plot could be used to (semi-automatically) divide a scan up into sections corresponding to distinct poses. The caveat, semi-automatically, is because the algorithm assumes that there is only one twitch in the section of data it is presented in picking the global maximum of local variance. It could be developed by considering local maxima or implementing it iteratively.

There are some motions that are clearly harder to detect than others, and these match those deemed difficult to discern in the qualitative discussion of the likelihood plots earlier in this report – notably the fourth motion in Rat1 and the final fore-leg pulls in Rat2. Perhaps the biggest surprise is that the small 1.5mm motions in Rat2 are identified quite reliably but the 4.5mm motion at 1260 seconds seems to be a challenge. In general

Scale	Twitch			Stage 1	Stage 2	Err.	Imprv.	Result
2	9mm	axial, fast	@180s	181.29	179.66	-0.34	Yes	PASS
	4.5mm	axial, fast	@360s	362.87	362.87	2.87	No	PASS
	9mm	trans, fast	@540s	548.35	549.32	9.32	N/A	FAIL
	4.5mm	trans, fast	@720s	761.17	762.31	42.31	N/A	FAIL
	4.5mm	trans, fast	@900s	902.33	902.33	2.33	No	PASS
	4.5mm	axial, fast	@1080s	1080.95	1080.95	0.95	No	PASS
4	9mm	axial, fast	@180s	180.72	177.7	-2.3	No	PASS
	4.5mm	axial, fast	@360s	362.23	362.23	2.23	No	PASS
	9mm	trans, fast	@540s	536.1	533.97	-6.03	N/A	FAIL
	4.5mm	trans, fast	@720s	725.72	727.73	7.73	N/A	FAIL
	4.5mm	trans, fast	@900s	890.5	889.74	-10.26	N/A	FAIL
	4.5mm	axial, fast	@1080s	1080.86	1083.77	3.77	No	PASS
8	9mm	axial, fast	@180s	180.27	180.27	0.27	No	PASS
	4.5mm	axial, fast	@360s	361.95	367.08	7.08	No	PASS
	9mm	trans, fast	@540s	532.97	527.39	-12.61	N/A	FAIL
	4.5mm	trans, fast	@720s	707.83	701.04	-18.96	N/A	FAIL
	4.5mm	trans, fast	@900s	905.8	911.78	11.78	N/A	FAIL
	4.5mm	axial, fast	@1080s	1081.88	1081.88	1.88	No	PASS

Table 5 Results for the 2-stage localisation for Rat1, for timescales 2, 4, and 8 seconds. The first column gives the timescale, the next 3 describe the motions. The twitch times estimated by the first and second stages of the method are given in the next columns, followed by the error of the second stage estimate. The last two columns state if the second stage has improved the estimate or not, and if the final estimate is within the one timescale of the twitch time (classed as a pass).

Scale	Twitch			Stage 1	Stage 2	Err.	Imprv.	Result
16	9mm	axial, fast	@180s	179.98	179.98	-0.02	No	PASS
	4.5mm	axial, fast	@360s	363.47	363.47	3.47	No	PASS
	9mm	trans, fast	@540s	546.07	541.36	1.36	Yes	PASS
	4.5mm	trans, fast	@720s	741.53	754.75	34.75	N/A	FAIL
	4.5mm	trans, fast	@900s	907.55	920.68	20.68	N/A	FAIL
	4.5mm	axial, fast	@1080s	1083.66	1083.66	3.66	No	PASS
24	9mm	axial, fast	@180s	182.09	176.48	-3.52	No	PASS
	4.5mm	axial, fast	@360s	361.73	361.73	1.73	No	PASS
	9mm	trans, fast	@540s	552.48	541.56	1.56	Yes	PASS
	4.5mm	trans, fast	@720s	745.79	750.4	30.4	N/A	FAIL
	4.5mm	trans, fast	@900s	891.92	908.37	8.37	No	PASS
	4.5mm	axial, fast	@1080s	1074.12	1074.12	-5.88	No	PASS
32	9mm	axial, fast	@180s	184.94	178.56	-1.44	Yes	PASS
	4.5mm	axial, fast	@360s	358.43	362.16	2.16	No	PASS
	9mm	trans, fast	@540s	557.53	541.38	1.38	Yes	PASS
	4.5mm	trans, fast	@720s	708.88	717.85	-2.15	Yes	PASS
	4.5mm	trans, fast	@900s	889.25	899.78	-0.22	Yes	PASS
	4.5mm	axial, fast	@1080s	1072.25	1072.25	-7.75	No	PASS

Table 6 Results for the 2-stage localisation for Rat1, for timescales 16, 24, and 32 seconds. The first column gives the timescale, the next 3 describe the motions. The twitch times estimated by the first and second stages of the method are given in the next columns, followed by the error of the second stage estimate. The last two columns state if the second stage has improved the estimate or not, and if the final estimate is within the one timescale of the twitch time (classed as a pass).

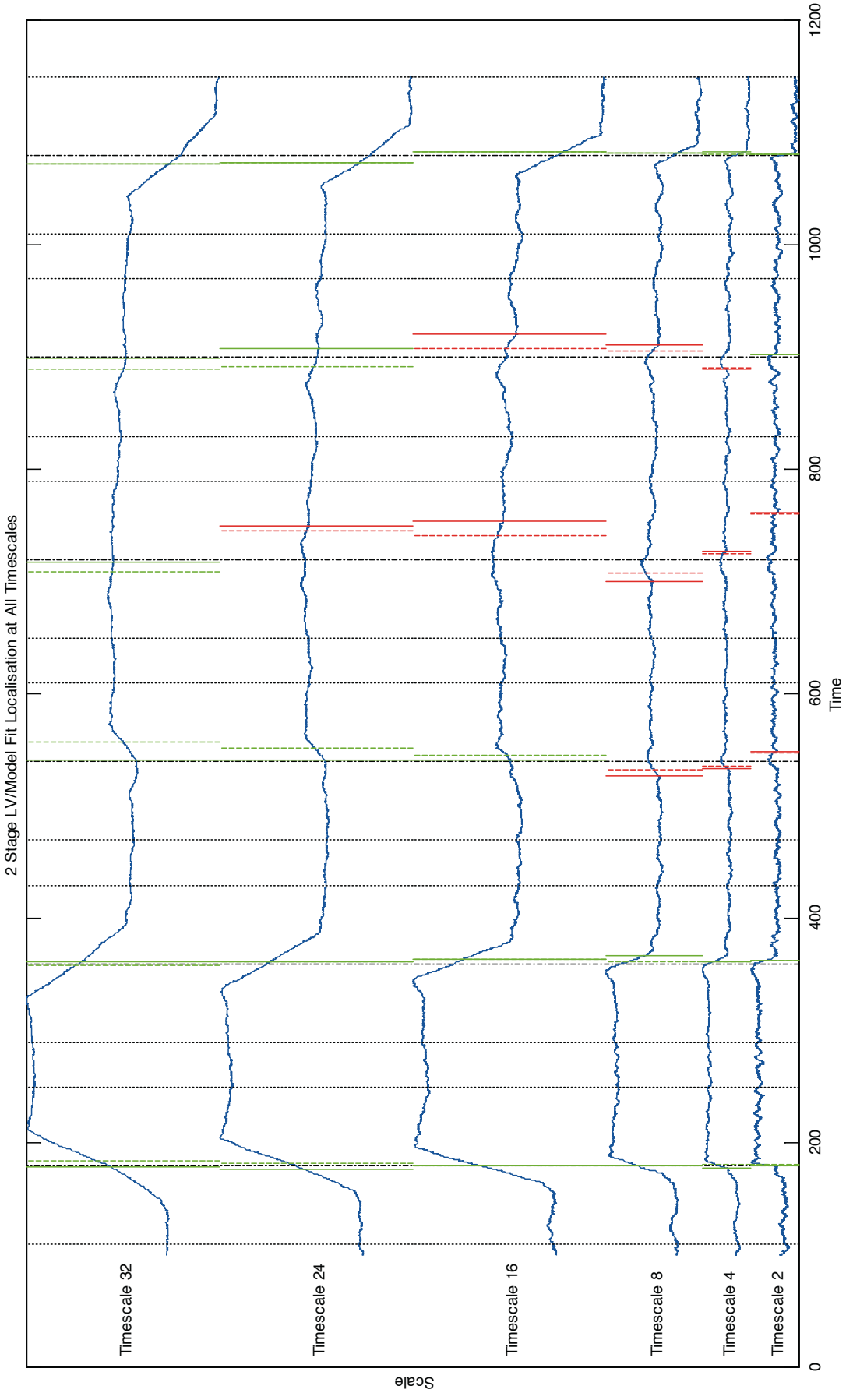

Figure 83 Summary of 2-stage localisation for Rar1. Blue lines: likelihood plots. Black lines: intended twitch times (chain dashed) and the extents the data presented to the localisation algorithm (dashed). Green lines: successful localisations, broken line for first stage estimate, solid line for second stage estimate. Red lines: failed localisations (not possibly related to actual twitch), broken line for first stage estimate, solid line for second stage estimate.

Scale	Twitch			Stage 1	Stage 2	Err.	Imprv.	Result
2	4.5mm	axial, slow	@180s	181.24	179.87	-0.13	Yes	PASS
	9mm	axial, slow	@360s	361.49	361.49	1.49	No	PASS
	4.5mm	axial, fast	@540s	541.6	540.17	0.17	Yes	PASS
	9mm	trans, slow	@720s	715.38	713.95	-6.05	N/A	FAIL
	1.5mm	axial, slow	@900s	894.49	892.74	-7.26	N/A	FAIL
	1.5mm	axial, fast	@1080s	1143.43	1144.99	64.99	N/A	FAIL
	4.5mm	trans, fast	@1260s	1252.65	1250.99	-9.01	N/A	FAIL
	4.5mm	trans, fast	@1440s	1442.14	1443.72	3.72	N/A	FAIL
	fore-leg pull		@1620s	1687.34	1685.72	65.72	N/A	FAIL
4	4.5mm	axial, slow	@180s	179.8	177.8	-2.2	No	PASS
	9mm	axial, slow	@360s	359.8	362.5	2.5	No	PASS
	4.5mm	axial, fast	@540s	542.18	540.12	0.12	Yes	PASS
	9mm	trans, slow	@720s	714.94	712.24	-7.76	N/A	FAIL
	1.5mm	axial, slow	@900s	896.76	895.33	-4.67	No	PASS
	1.5mm	axial, fast	@1080s	1137.43	1135.28	55.28	N/A	FAIL
	4.5mm	trans, fast	@1260s	1263.02	1265.29	5.29	N/A	FAIL
	4.5mm	trans, fast	@1440s	1434.02	1432.94	-7.06	N/A	FAIL
	fore-leg pull		@1620s	1557.46	1559.32	-60.68	N/A	FAIL
8	4.5mm	axial, slow	@180s	183.4	181.02	1.02	Yes	PASS
	9mm	axial, slow	@360s	358.65	360.88	0.88	Yes	PASS
	4.5mm	axial, fast	@540s	544.26	541.11	1.11	Yes	PASS
	9mm	trans, slow	@720s	779.13	786.37	66.37	N/A	FAIL
	1.5mm	axial, slow	@900s	879.64	886.24	-13.76	N/A	FAIL
	1.5mm	axial, fast	@1080	1123.28	1129.08	49.08	N/A	FAIL
	4.5mm	trans, fast	@1260	1248.7	1243.07	-16.93	N/A	FAIL
	4.5mm	trans, fast	@1440	1454.19	1461.53	21.53	N/A	FAIL
	fore-leg pull		@1620	1587.16	1592.17	-27.83	N/A	FAIL

Table 7 Results for the 2-stage localisation for Rat2, for timescales 2, 4, and 8 seconds. Columns as for Rat1 tables above.

Scale	Twitch			Stage 1	Stage 2	Err.	Imprv.	Result
2	4.5mm	axial, slow	@180s	190.81	183.85	3.85	Yes	PASS
	9mm	axial, slow	@360s	355.82	361.79	1.79	Yes	PASS
	4.5mm	axial, fast	@540s	546.74	540.67	0.67	Yes	PASS
	9mm	trans, slow	@720s	709.04	719.79	-0.21	Yes	PASS
	1.5mm	axial, slow	@900s	908.52	900.63	0.63	Yes	PASS
	1.5mm	axial, fast	@1080s	1073.75	1073.75	-6.25	No	PASS
	4.5mm	trans, fast	@1260s	1234.91	1230.77	-29.23	N/A	FAIL
	4.5mm	trans, fast	@1440s	1429.7	1441.66	1.66	Yes	PASS
	fore-leg pull		@1620s	1631.2	1631.31	11.31	No	PASS
4	4.5mm	axial, slow	@180s	195.08	182.15	2.15	Yes	PASS
	9mm	axial, slow	@360s	351.06	361.36	1.36	Yes	PASS
	4.5mm	axial, fast	@540s	554.45	540.72	0.72	Yes	PASS
	9mm	trans, slow	@720s	702.84	720.14	0.14	Yes	PASS
	1.5mm	axial, slow	@900s	906.75	891.55	-8.45	No	PASS
	1.5mm	axial, fast	@1080s	1071.61	1071.61	-8.39	No	PASS
	4.5mm	trans, fast	@1260s	1302.07	1321.96	61.96	N/A	FAIL
	4.5mm	trans, fast	@1440s	1423.03	1440.38	0.38	Yes	PASS
	fore-leg pull		@1620s	1621.41	1621.41	1.41	No	PASS
8	4.5mm	axial, slow	@180s	201.13	181.77	1.77	Yes	PASS
	9mm	axial, slow	@360s	345.27	360.21	0.21	Yes	PASS
	4.5mm	axial, fast	@540s	559.63	541.34	1.34	Yes	PASS
	9mm	trans, slow	@720s	697.4	720.21	0.21	Yes	PASS
	1.5mm	axial, slow	@900s	907.99	883.28	-16.72	No	PASS
	1.5mm	axial, fast	@1080	1058.08	1058.08	-21.92	No	PASS
	4.5mm	trans, fast	@1260	1279.38	1279.38	19.38	No	PASS
	4.5mm	trans, fast	@1440	1416.76	1440.06	0.06	Yes	PASS
	fore-leg pull		@1620	1603.94	1630.87	10.87	Yes	PASS

Table 8 Results for the 2-stage localisation for Rat2, for timescales 16, 24, and 32 seconds. Columns as for Rat1 tables above.

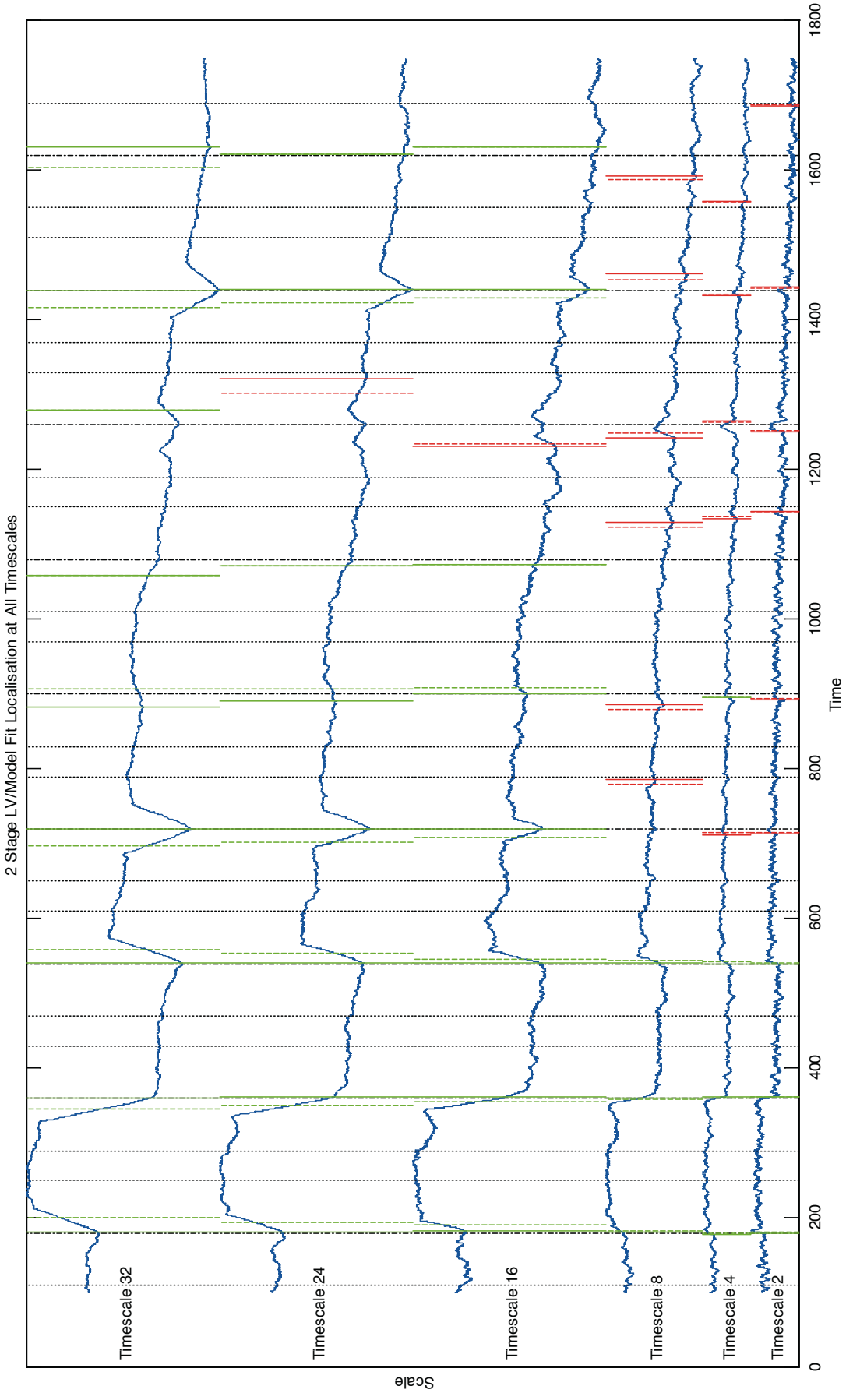

Figure 84 Summary of 2-stage localisation for Rat2. Blue lines: likelihood plots. Black lines: intended twitch times (chain dashed) and the extents the data presented to the localisation algorithm (dashed). Green lines: successful localisations, broken line for first stage estimate, solid line for second stage estimate. Red lines: failed localisations (not possibly related to actual twitch), broken line for first stage estimate, solid line for second stage estimate.

the longer time-scales do produce more reliable localisations, as would be expected, although there are exceptions to this: 900s in both Rat1 and Rat2.

2.6 Multiscale Localisation

The purpose of this section is two fold: to demonstrate the multiscale localisation by applying it to both the Rat1 and Rat2 datasets and to compare the two approaches to thinning the local maxima proposed in §3.3 of the previous chapter. The method was developed using the Rat1 data, so applying it to the Rat2 data is a test free from training bias.

Figure 85 and Figure 86 show the thinning strategies by identifying local maxima of the local variance at each timescale for the Rat1 data. Note that the 'collision thinning' is applied during the identification of the maxima, selecting the greatest one of any set of local maxima within one and a half timescales of each other. The 'morphological thinning' is applied by modifying the local variance before detecting local maxima, the result of this modification is visible on the plots in Figure 86. Figure 87 and Figure 88 demonstrate the maxima tracking using both thinning approaches for Rat1. Figure 89 and Figure 90 do likewise for Rat2.

The morphological approach is very effective – one false positive, one omitted twitch. The 'collision thinning' approach seems less effective, it reduces the number of repeated maxima, but would still leave one maximum for most of the clusters of maxima. That this approach is less effective is not too surprising, after all it only considers the proximity of the maxima to each other, not their significance. The morphological approach flattens small oscillations in local variance, instead of just requiring them to be distant from each other.

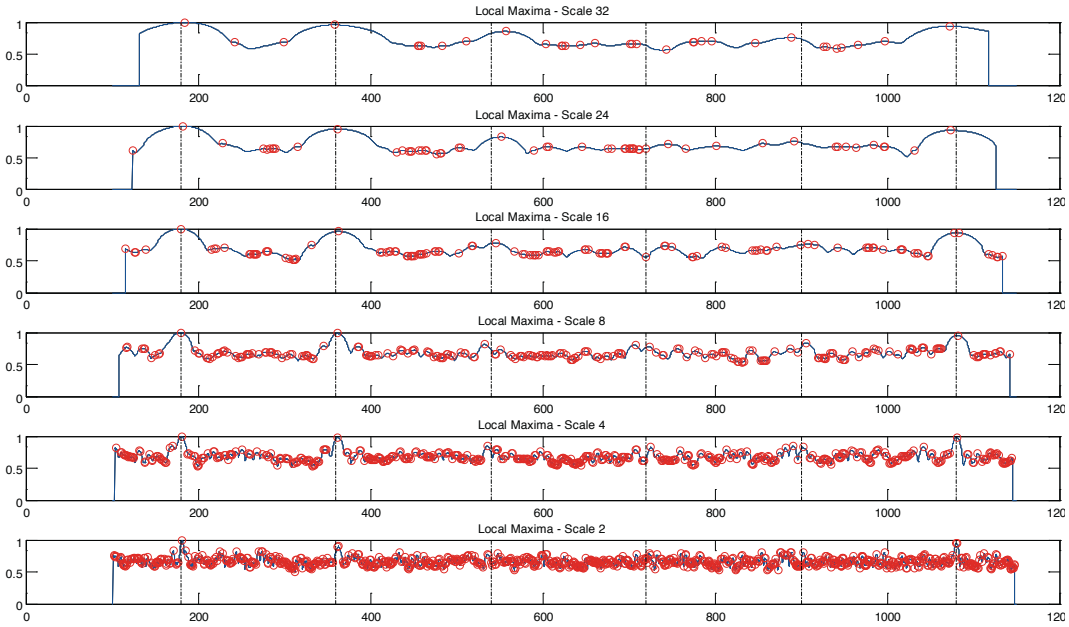

Figure 85 Local maxima (red circles) of the local variance of the likelihood plot at each timescale for Rat1. Here the maxima are 'collision thinned': no local maxia can be labelled withing one and a half timescales of each other. This is applied to all timescales.

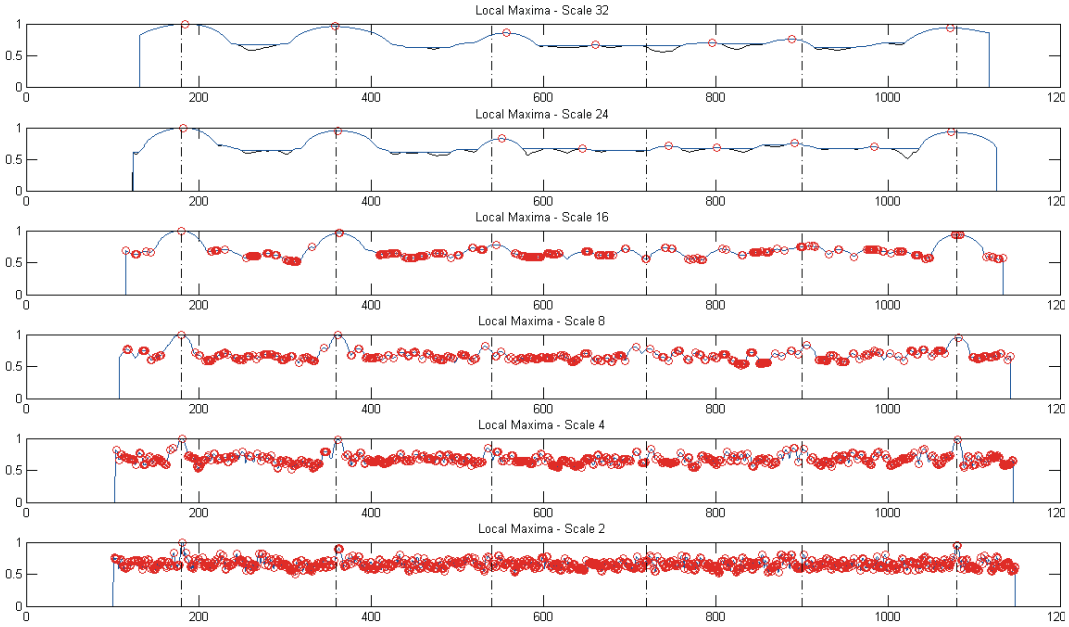

Figure 86 Local maxima (red circles) of the local variance of the likelihood plot at each timescale for Rat1. Here the maxima are 'morphologically thinned': the local variance is 'closed' with a shape element of a flat rod of length equal to one and a half timescales. This is applied to the two coarsest timescales – the blue line is after thinning, the black one before.

Note that both approaches limit the potential to recover twitches close to one another, especially in conjunction with the funnelling discussed above. Ideally the maxima tracking should allow a coarse scale maxima formed by the interaction of two motions to bifurcate to recover twitches close to each other.

As would be expected the performance is better for Rat1 (which was used to guide the choice of parameters for the thinning). In Rat2 the thinning at the 24 seconds timescale seems to have removed 3 twitches that should not have been removed. However, the method is still promising and there is plenty of scope to optimise it, for instance changing the shape element to incorporate more knowledge about the expected shape of the local variance near twitch times. (Also, note that the last motion is too close to the end of the data to be detected.)

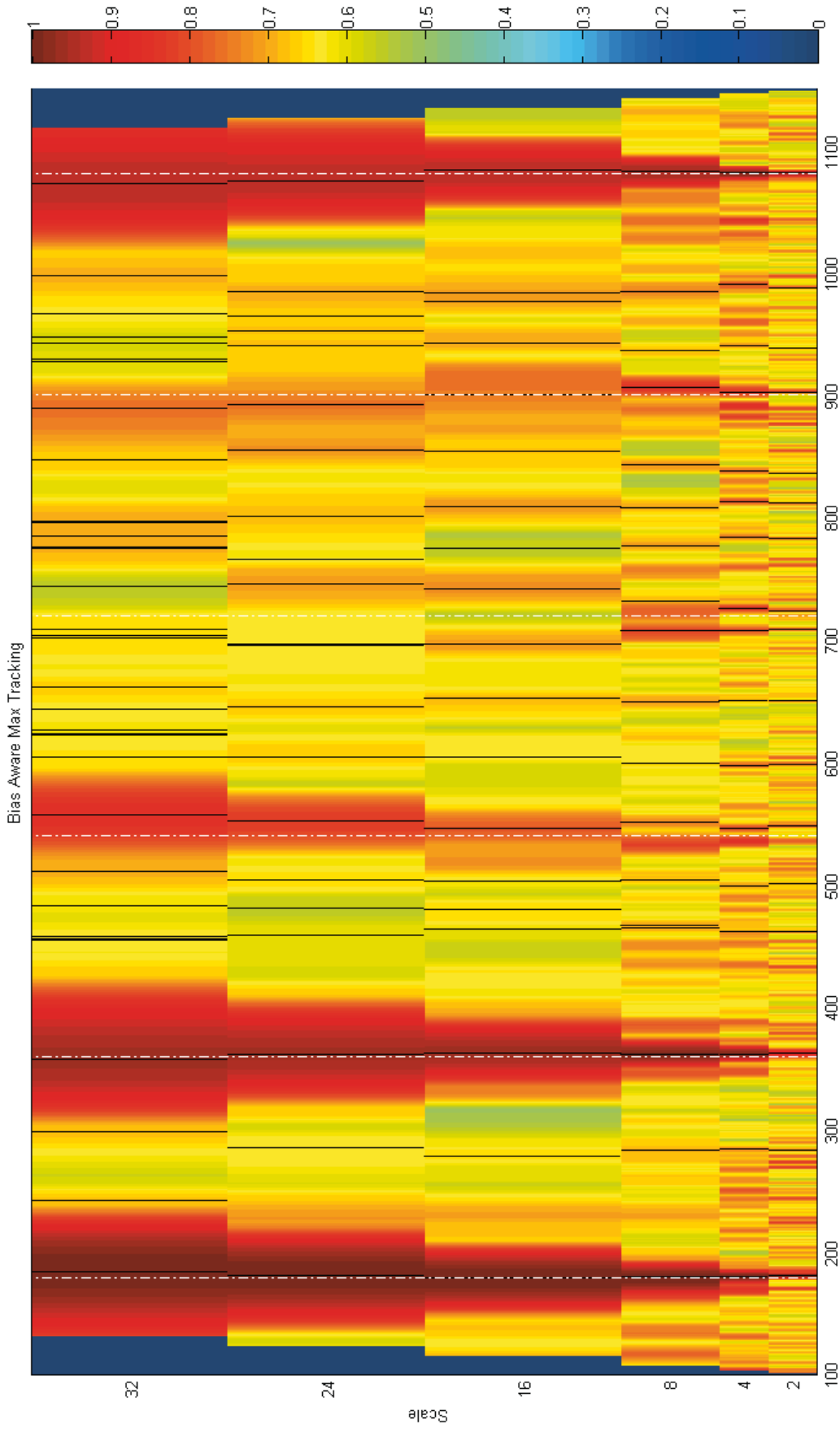

Figure 87 Local maxima tracking for Rat1. Here the maxima are 'collision thinned'. Tracking the local-maxima through scale space. The identified maxima are shown with the black lines. Intended twitch times are shown with the white chain-dashed lines.

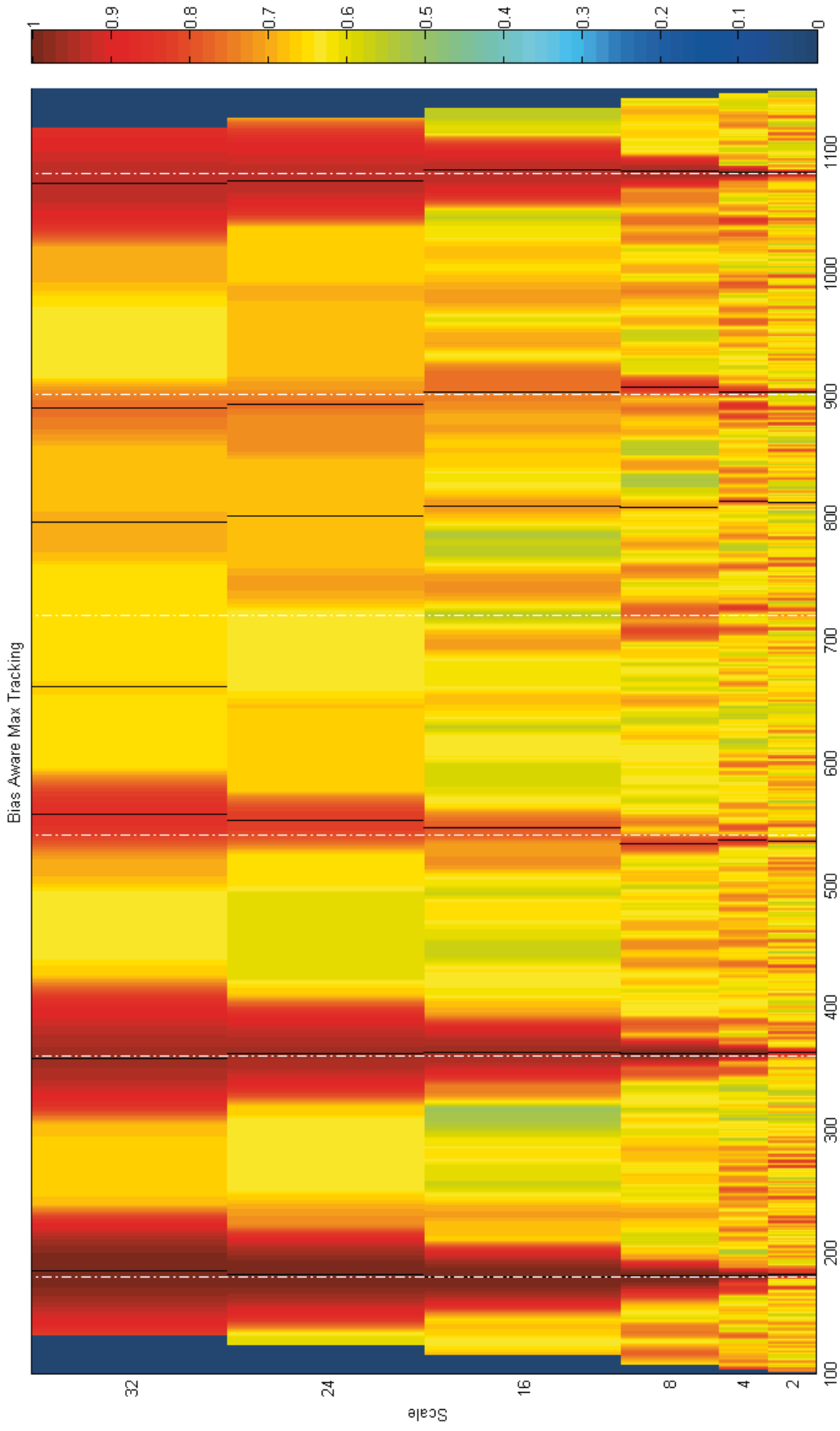

Figure 88 Local maxima tracking for Rat1. Here the maxima are 'morphologically thinned'. Tracking the local-maxima through scale space. The identified maxima are shown with the black lines. Intended twitch times are shown with the white chain-dashed lines.

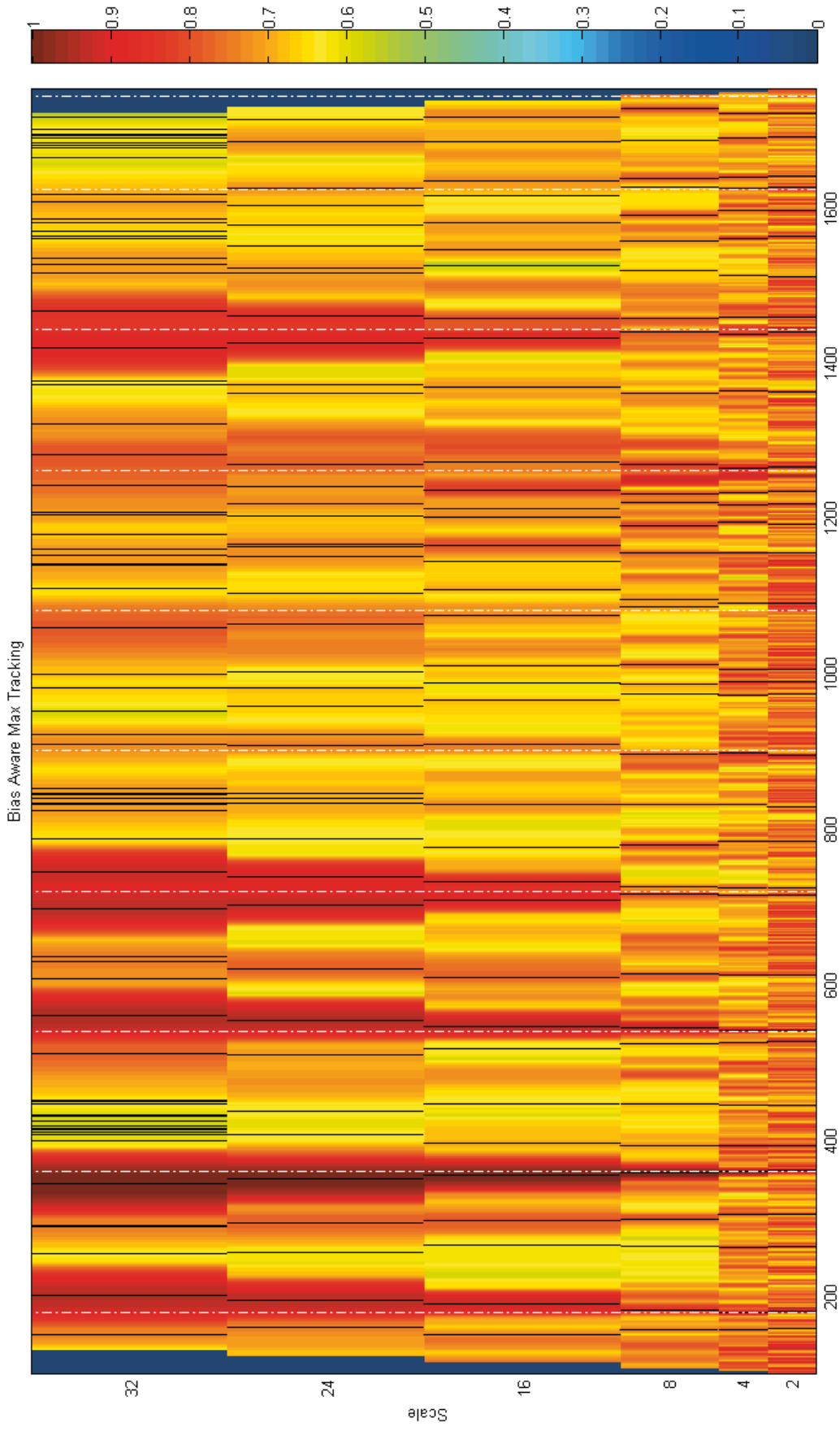

Figure 89 Local maxima tracking for Rat2. Here the maxima are 'morphologically thinned'. Tracking the local-maxima through scale space. The identified maxima are shown with the black lines. Intended twitch times are shown with the white chain-dashed lines.

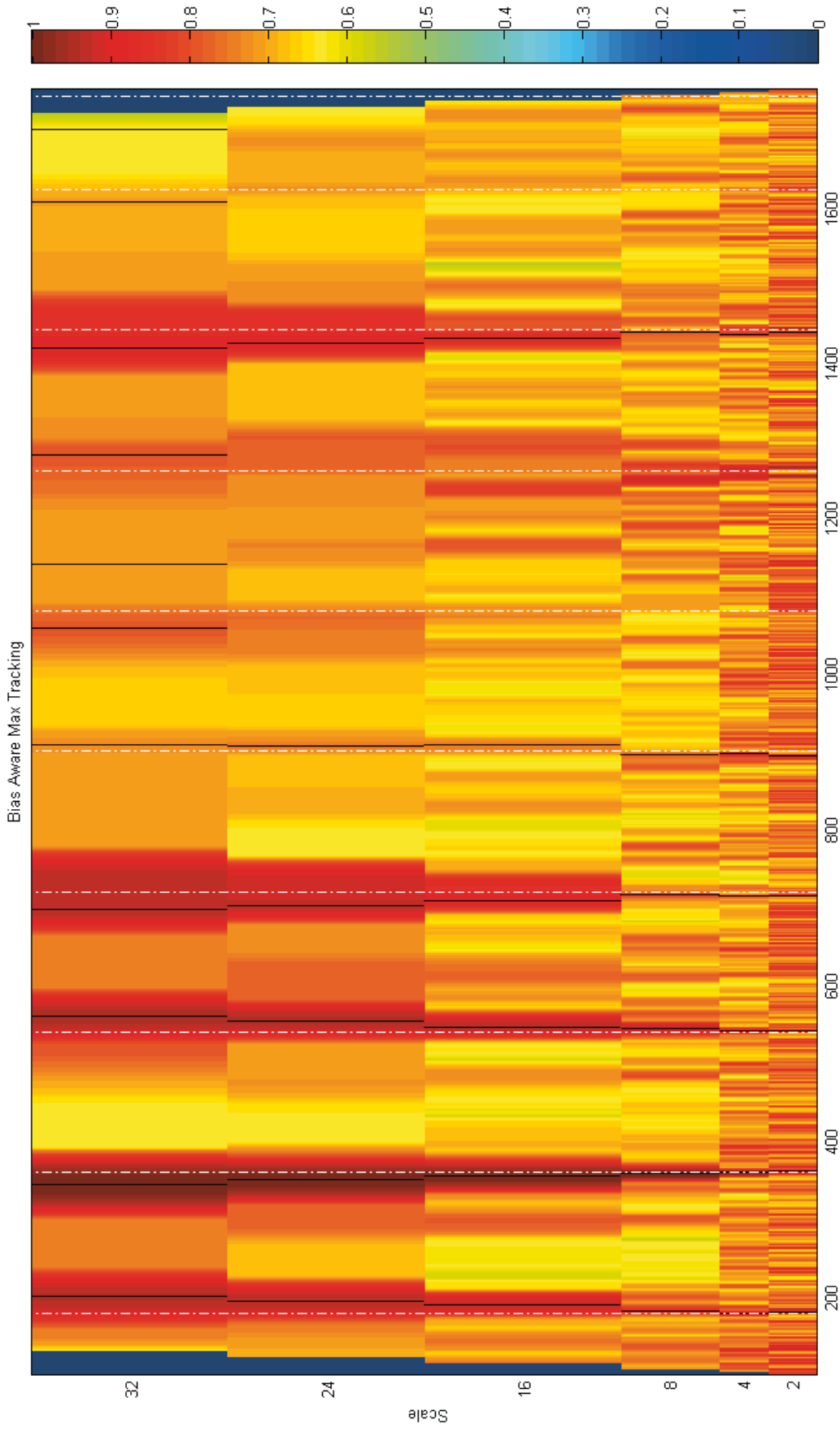

Figure 90 Local maxima tracking for Rat2. Here the maxima are 'morphologically thinned'. Tracking the local-maxima through scale space. The identified maxima are shown with the black lines. Intended twitch times are shown with the white chain-dashed lines.

3. Conclusions

The experiments presented in this chapter have started to explore the behaviour of the twitch detection method. In general that has been as expected: larger motions produce more significant manifestations on the likelihood plot. Although not predicted in advance, that axial motions also produce more prominent manifestations than trans-axial ones is sensible. Fast and slow motions did not produce an identifiable difference, although this might be due to a lack of difference between 'fast' and 'slow' relative to the timescale of the twitch detector.

The unexpected phenomenon in this chapter was the drift in the likelihood plot for Rat2. This seems to be a result of imaging a live subject, and two explanations have been suggested and tested, which strong suggests that one (ongoing bulk tracer migration) is a factor.

Automated twitch localisation has been demonstrated successful, both the 2-stage local variance method which requires the likelihood plot to be divided into section where only one twitch has occurred and a multiscale approach able to handle multiple twitches in the data that has proven itself very promising.

One experiment was planned but proved impractical in the time available: that was to look at the impact of the breathing and heart beat in the Rat2 study on the likelihood plot. The intention was to take the heart rate and breathing rater recorded by the gating system on the scanner and to look for corresponding frequencies in the no-twtich sections of the likelihood plot. Unfortunately, given the need to implement a means of extracting the gating signals from the scanner data this experiment not possible during this work.

Chapter VII: Future Work

What has been presented in the previous chapters is the development of the twitch detection method to date and a snapshot of its present form. As a novel technique as more work is done with it the range of ways it could be used and potential improvements expands seemingly exponentially. It would not have been practical to explore every option along the way, not before the method was shown to work and was better understood. The significance of the count rate on the likelihood plot, and hence that decay correction would always been needed, for instance, was not appreciated until the method was first tested on real data.

1. Continued Development

It is possible at this juncture to consider which unexplored options seem most promising and so should be given priority for further work. These are modifications to the current formulation and therefore not radically new projects. Further experimental work has been highlighted in Chapter VI, so will not be repeated here.

1.1 Twitch Localisation

More than just promising, the localisation of twitch features on the likelihood plot is a pressing area. As mentioned earlier, the current approach was put together to facilitate experiments, and was not the result of extensive research in signal processing and feature detection.

These are both very well developed fields and so the task of finding alternative ways to locate the twitch feature is not difficult, rather the difficulty comes from the range of possible options to explore. *There is a discussion of some of the options looked into to date in Appendix B.*

Secondly the interaction between the twitch detection and the twitch localisation methods should be better understood. This is arguably an inherent part of designing a method for twitch localisation. An analytic investigation of the way that changes in the sinogram affect the formation of the likelihood plot could be undertaken. For instance, an expression relating the motion of a point source to the deviations seen on the likelihood plot would be very interesting.

1.2 Multiple Timescales

Although it is harder to detect twitches on the likelihood plot for small timescales, the drawback with long timescales is that each twitch's manifestation on the likelihood plot is much longer and therefore more likely to overlap and interact with those of other twitches. Initial investigation of multiple timescale localisation seems promising.

Further investigation of multiscale methods could take two approaches either the twitch localisation method to simultaneously consider multiple scales or a supplementary step in which the twitch location estimates from each individual scale are fused together. The former seems more appealing, although the latter is closer to what has been presented.

1.3 Sinogram Resolution

The list mode data stores coincidences events as a pair of crystal addresses, and these are converted into sinogram coordinates during the twitch detection process. Multiple crystal pairs will contribute to the same sinogram bin. For the purposes of developing the twitch detection method, this conversion was based on the default used in the scanner's software, however it need not be: any grouping of crystal pairs should produce bin counts that satisfy the Poisson distribution required by the method.

Therefore, it is possible to alter the resolution of the sinogram and use coarser or finer bin sizes for the analysis. Coarser bin sizes would capture less spatial detail and consequently be less sensitive to small motions (or even large motions, if the only changes in tracer distribution that result from them are from small scale variations of tracer concentration, the 'texture' of a large region for instance). However the larger bins would be expected to collect more counts. This has two effects: firstly there will be fewer bins with no counts – therefore there will be fewer cases when no delta can be defined (see §2.3.2.1 of Chapter IV); secondly, as the Poisson parameter grows larger the Poisson

distribution could be better approximated by a Gaussian. The latter is of little computational benefit (as the deltas can be precomputed) but might be of some analytic use.

It would even be possible to histogram the crystal pairs in a manner that does not correspond to projection. For instance, a 1D histogram based on the distance between the two crystals. There is a wide range of options here, and, which although unnecessary at first glance, may prove useful in the context of 'regional twitch detection', proposed later in this chapter (§2.4).

1.4 Time of Flight

The time of flight information captured by modern scanners could be incorporated into the twitch detection method, with the similar effects expected as for finer resolution sinograms. The lines of response for each sinogram bin could be further divided into segments, with which segment the event originated in estimated from the time of flight information. This, in effect, adds another dimension to the sinogram.

There is a difference however: as discussed in §2.5 of Chapter I, the localisation provided by time of flight is coarse and not exact, whereas the allocation of crystal pairs to bins is exact. When time of flight information is incorporated into reconstruction it is used to centre a 'kernel' (often a Gaussian) modelling a probability density function of where the annihilation actually occurred. Something similar might have to be used in deciding which 'sub-bin' along the line of response to allocate the count to. That, however, would mean that the twitch detector would become stochastic – presently for a given list mode file the same timescale will always produce the same likelihood plot.

1.5 Different Window Arrangements

One of the first options that was chosen not to be explored was how best to arrange the windows: although obvious, taking two windows of equal length, one before each time and the other after each time, is not the only way. Nor is it necessarily the best option.

In order to estimate the log joint likelihood what is needed is an estimate of the rate parameter for the Poisson distribution. As used in this thesis, that parameter is the expected number of counts in each window width. It can easily be scaled to account for different window widths – indeed the no twitch hypothesis states that the parameter should not change so the scaling with window width is just linear (providing that the effect of decay correction is removed from the window width before the scaling is considered).

As a demonstration of the possible utility of this consider having a 'reference' window being the entire period since the last twitch was identified, and a fixed width 'test window' forward from the current time. By scaling the counts observed in the 'reference' window the expect number of counts for each sinogram bin in the 'test' window could be predicted under the no-motion hypothesis and from there the likelihood of observing that test window could be computed. The potential advantage of this is that more data is used to estimate the Poisson parameters and they will be more stable because there will not be as much variation of their estimates as the twitch detection moves through the scan.

It is even conceivable that this approach could be extended into an iterative formulation, where the objective is to find the optimal sections of the scan. The task would be to divide the scan up into sections such that any pair of windows from within a section has a higher likelihood than a pair of windows where one of them is in a different section.

Less radical than this would be to compare a short 'test' window against the parameters estimated for the whole scan. This would be especially useful to identify sections to discard, for example if twitches found in clinical practice tend to conclude with the patient retuning to their original position this would trim out the contaminated data.

1.6 Odds Ratio

The suggested iterative window placement of the previous section would be very satisfying because the twitch times it suggests would mean that any two windows (of some fixed length) are sufficiently (by some definition) more similar if they come from the same section of the scan than from a different one. Its conclusions would be binary and optimal, given its parameters.

Something similar could be pursued on the likelihood plot without the need for a radical reformulation into an iterative process. At present the likelihood plot shows the likelihood of observing both windows assuming a common set of Poisson parameters for the sinogram bins, where those parameters are chosen to maximise the likelihood. Consider the reverse: assume that there was a twitch at the time so that each window should have its own set of parameters, in which case the best choice of parameter for any bin on either sinogram would be the observed number of counts. A new log joint likelihood could be computed that describes the chance of observing the data assuming there was a twitch.

The ratio of the likelihoods under the twitch and no-twitch hypotheses would be a more binary approach to estimating where there are twitches: either one hypothesis makes the observed data more likely or the other does. Twitch localisation would be much simpler.

It was briefly tried during the early development of the twitch detector, but felt to be quite ill conditioned and difficult to predict its results. The suspicion at the time was that

of the two hypothesis used to pick the parameters the twitch hypothesis is predisposed to being able to pick better suited parameters as they have to fit a shorter section of data. In other words there is an over-fitting bias that favours the twitch hypothesis over the no twitch hypothesis. Altering the window widths might help ameliorate this, as would estimating each window's parameters from a double window width of data on each side of the time (to reduce the over-fitting bias). Alternatively, Bayes rule could be used to introduce a prior that corrects for the over fitting bias. The strength of this 'correction' would be an a priori choice, however, which the development of the existing method sought to avoid.

1.7 Online Implementation

The final development suggested here is that the implementation of the twitch detector, perhaps reformulated to use the odds ratio, be implemented in such a way as to be able to read the list mode as it is outputted from the scanner. This would enable online use of the twitch detector (albeit with a short delay for to allow the 'after' window time), indeed this was the initial motivation for the event-wise implementation described in Chapter IV – all that remains is the 'plumbing' from the relevant port on a workstation to the software (not that that is a trivial task). Online twitch localisation would be more of a challenge, and that is a factor to bear in mind when considering options for twitch localisation.

Having an online twitch detector would enable new use cases to be explored, beyond dividing the scan into sections for registration. The simplest would be to acquire data for as long as needed until a stable section is encountered that lasts long enough to form and image.

1.8 Parallelisation

The performance of the twitch detection is such that there has been little need to explore parallel computation. However, there are two interesting aspects to this, the first practical point on how the process can be parellelised and the second is that it enables some of the alternative uses discussed in the next section to be implemented easily.

Because the calculation of each delta requires an up to date value of the bin count in both the before and after windows the event stream must be analysed in the correct sequence – except between each time marker. There is no assumed order between coincidences that occur between the same pair of timing packets, so it would be possible to paralise these computations. However, the sinogram memory must be shared and so enforcing any ordering of how events are to be processed (e.g. the rules of tiebreaker when different sorts of delta are pending) has to consider the implication that other delta calculations may be accessing that memory.

The good news is that every sinogram bin is independent of each other. Current before and after bin counts, and transition & exit queues can be maintained separately for each bin (or subset of bins). Furthermore, the likelihood plot for each bin can be computed separately and then, to compute the overall ***log*** joint likelihood the binwise liklihood plots can be summed together at each point in time. It would be impractical from a memory perspective to do this for every bin, but the key thing is that there is little penalty to combine the likelihood plots from subsections of the sinogram. (There is also a cost in dispatching the events to different streams for processing, but much of this exists anyway as which bin the event falls into needs to be computed anyway.) This is the second aspect, how this enables other things that will be mentioned again over the coming pages.

2. New Projects

The following suggestions outline extensions that require more substantial work and, in some cases, leaving the concept of twitch detection behind and instead finding new uses for the likelihood plot.

2.1 Data Framing

In many cases during dynamic studies the scan is divided into frames and this is done a priori. The normal practice is to use short frames near the start of the scan, when rapid changes in the tracer kinetics are expected, and longer frames later on, because short frames have lower signal to noise. The problem with this approach is that it is arbitrary.

Given that the likelihood plot directly measures the self consistency of a section of scan data it could be used as a proxy for the amount of variation in the scan at any chosen time in the scan. Therefore low values on the likelihood plot would suggest rapid changes which would compel the use of short frames, whereas comparatively high values of likelihood would permit longer frames to be used.

By considering the variation of likelihood plot over the scan duration the optimal set of frames to balance the need to capture fast kinetics without sacrificing too much signal to noise ratio should be computed, specifically for any given scan.

2.2 Data Gating

A periodic motion should, for a suitable timescale, manifest as a periodic variation on the likelihood plot. This is not unrelated to the work in [9] where the variation of count rate as result of the heart moving through the uneven sensitivity of the scanner across its field of view is used to gate breathing.

Finer timescales would be expected to resolve this better, but at the cost of more noise that might mask the effect. Although this idea was part of the motivation of testing the twitch detector with live animals further work needs to be done to make a meaningful assessment of its potential.

The advantage of this approach follows from the inconvenience of setting up the gating equipment and the reliability of gating – both more of an issue for breathing that for the cardiac cycle.

First, however, how to optimise the clarity of the signal should be investigated. Some of the extensions suggested earlier, like asymmetric windows, might help control noise while still using a fine enough test window to produce the effect on the likelihood plot. The second area requiring attention is the validity of the idea itself. Note that the effect on the likelihood plot is based on how consistent the data in the two windows is, so there is some concern that it may not be the motion that is driving the effect but rather some other process at best only partially correlated with the motion.

2.3 Data Optimal Gating

Optimal gating is the trade name for a PET procedure in which the gating signal for breathing is used to switch on and off the recording of PET data, so that only one gate's data is acquired. Because the CT scan for attenuation correction is acquired very quickly it will only match one of the gates, so there will be attenuation and scatter correction artefacts for other gates. Data from gates that do not match the attenuation correction CT are ignored. The word 'optimal' refers to the way that a balance is struck between having a longer gate (and hence more data per breath) and more motion effects. Using the likelihood plot concept it would be possible to refine this procedure such that the consistency of the data is used directly, rather than assuming that the amount of motion effects are related to the variation of the respiratory gating signal.

If the CT for attenuation correction is acquired with the patient holding their breath at end expiration (for example) after moving to the PET acquisition the patient could be asked to repeat the breath hold, thereby acquiring a 'reference' window of data against which a derivative of twitch detector method could compare windows from rest of the scan. Only if the likelihood is high enough (i.e. the patient is close to full inspiration, and so in the same configuration as for the density map) will any given window be recorded. Again acquisition would continue until enough 'good' data is acquired. The advantages and concerns about validity discussed with in §2.2 are equally true here.

2.4 Regional TD

In the case of data gating, performance would obviously be improved if the likelihood plot was formed from counts exclusively originating around the relevant part of the body – the mediasteinum for the heart or the diaphragm for breathing. Likewise, if the intention is to image part of the subject then motions the do not affect that region might be less important.

The problem is that it is not possible to exclusively select regions in the image domain on the sinogram – as discussed in §2.7 of Chapter I, each point in the subject contributes to many points in the sinogram domain, and each point in the sinogram will be influenced by many points in the subject. It is possible to say that activity in a region might contribute to some sinogram bins and not to others, and indeed to estimate the relative impact a region has on two sinogram bins or that two regions have on a specific bin. That does not, however, mean that the fraction of counts contributed by a region to a bin can be estimated: that depends on the activity levels for all points in the image that can affect that bin. The process of working out the relative effect is simple: the region, described by a mask so that all points outside the region are zero, is projected into the sinogram domain. (The mask need not be binary, the region can 'partially include' some parts of the subject.)

Therefore, for some region of the subject, the sinogram can be partitioned into bins that are not related to the region and those that are. Some bins are 'partially' included, even with binary masks.

This suggests the strategy of forming several likelihood plots, each using a different subset of sinogram bins. The simple example would be based on the portion of bins according to whether or not they are influence by a specified region – producing two likelihood plots 'not-out' and 'out' (...of the region's influence). A motion confined to the region would have and effect exclusively on the 'not-out' likelihood plot. Any motion exclusively outside the region would affect both, but the 'out' plot more. Lastly a global motion, equally affecting both the region and the surroundings, would have an effect on both plots, more equally than the outside-region motion. (The effect would be equal or perhaps better equivalent, but not identical because the effect of a motion is too dependant on the subject that is moving).

Note from the discussion of parallelisation that the finest set of subsets could be computed simultaneously and then various combinations added together to form the likelihood plots needed for multiple regional twitch detection.

The trick to localising the twitch (spatially) is then similar to the task of localising the twitch (in time): the likelihood plots must be inspected to find out where and when there was a motion. This extends the twitch localisation task from a signal processing plus feature detection to a multivariable (or multi-channel) signal processing plus feature detection and identification task. Incidentally, an area of literature felt to be worth investigating is the analysis of EEG signals. The multiple channels of data recorded exhibit similar interrelationships as brain activity will affect many or all the electrodes. (Although, in the case of EEG signals there is some delay between when each channel will detect the effect, so the task here may prove more challenging.)

2.5 Multi-scale Twitch Detection

In [64] the relationship between wavelet transforms of the sinogram domain data and those in the image domain data is discussed, with the result that the decomposition can be performed before reconstruction and then multiple images reconstructed corresponding to different scales. If a similar logic were to be applied here it would be possible to filter the singoram domain data so that the twitch detector could be tuned to detect motions whose effects apply at different scales in the image.

Moreover, with reference to the ease of computing multiple likelihood plots corresponding to different subsets of the sinogram in parallel, the multi-scale likelihood plots could be computed simultaneously. It is also likely that the filtering could be delayed, applied as a weighting of the likelihood sub-plots when they are added together.

Conclusions

To conclude this thesis an integrated demonstration of all the work is presented, taking a section of preclinical data with a twitch, finding the twitch, performing a registration using the two sections of data identified. Then the image is reconstructed, as if the motion had not occurred, thereby demonstrating the complete process of twitch correction.

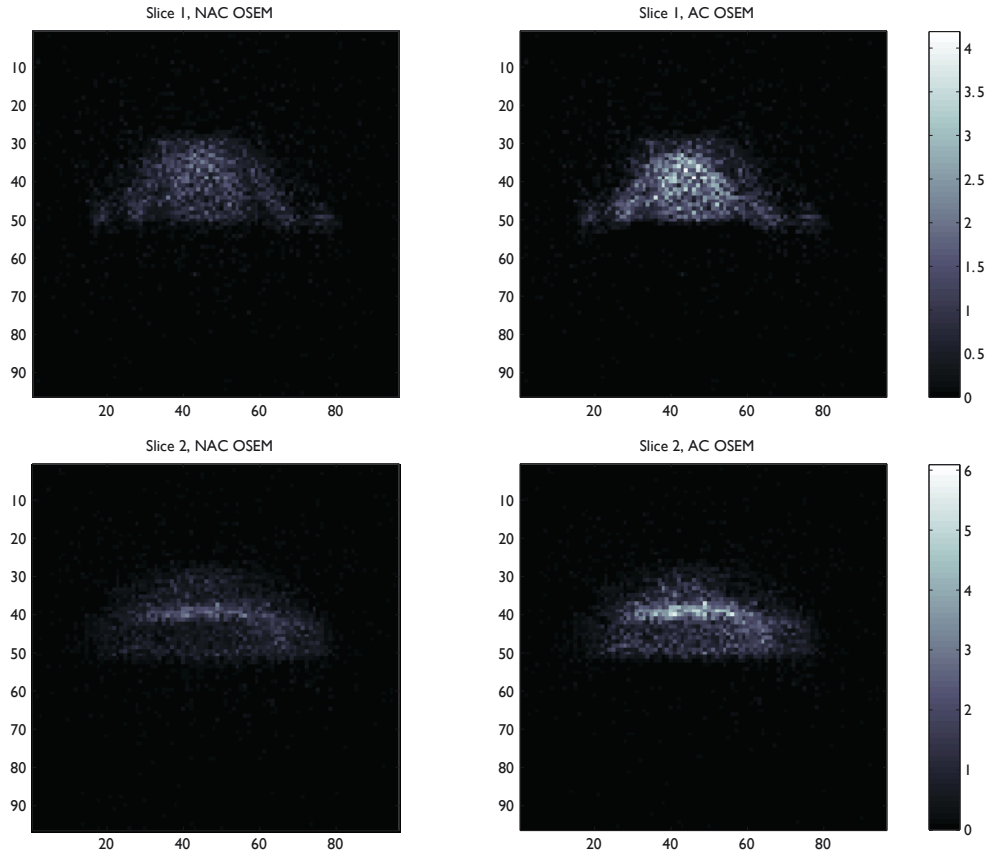

Figure 91 The reconstruction of the data without any motion correction. Two cross sections are shown (the rows), without any attenuation corrections (left) and with the approximate attenuation correction described in the text (right).

1. Raw Data

A section of data from the Rat 2 study (see page 172) was selected that included a twitch (specifically a slow 9mm translation across the field of view). If this data is reconstructed directly (using the OSEM algorithm of [23]) the result is as in Figure 91. Two representative cross-sections through the animal are used.

Unfortunately it was not feasible to extract the CT data from the scan. This means that the effect of attenuation correction and the artefacts of motion that result from that cannot be properly studied. However, to try and show an indication of the effect a crude approximation to attenuation correction is used, much as in [13]. From the first

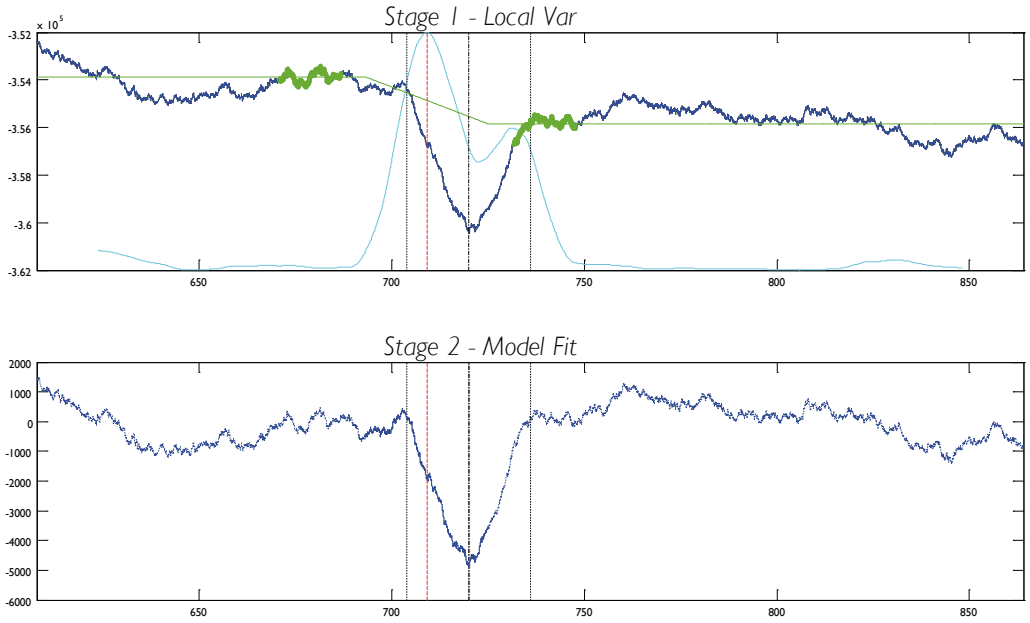

Figure 92 The likelihood plot that results from twitch detection, undergoing the two stage twitch localisation method. For details see page 156. Here each stage works as planned, producing an estimate of 720.18 seconds. (Intended time of movement: 720 seconds.)

position the outline of the rat is identified and a binary mask is then projected back to the sinogram. This gives an indication of the distance through the animal that each line of response travels, and thus some idea of the amount of attenuation that would be expected. The sinogram bin counts can then be up-weighted to compensate for the attenuation. Images using this trick are also shown in Figure 91 – note that because the sinogram up-weighting to account for attenuation deliberately only corresponds to one position it reduces some of the appearance that two positions have been mixed together.

2. Twitch Detection

Before any motion can be estimated or corrected the time of the motion must be found. In this case, it could be easy, after all the motions were deliberately introduced. In practice, with the motions being unexpected, it is also desirable to be able to detect them without having to watch the entire scan or use additional equipment. So, in fact, being able to deduce the time of motion retrospectively with nothing more than the list mode data is an ideal test.

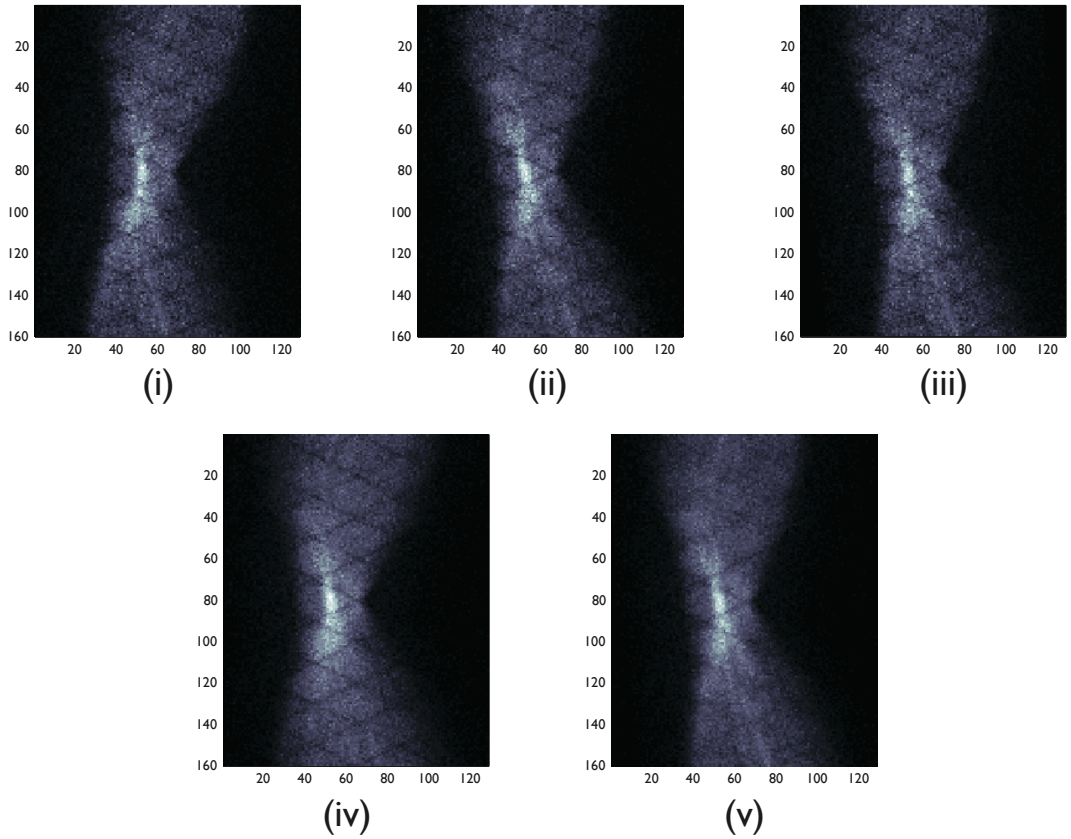

Figure 93 The sinograms before and after registration for one of the cross sections. In (i) and (ii) the sinograms from the two sections of data identified by the registration are shown, template and reference respectively. The registered template sinogram is shown in (iii). The combined sinogram, without registration is shown in (iv) – i.e. (i) + (ii); whereas (v) shows the combined sinogram after the registration is performed – i.e. (ii) + (iii).

This is done by computing the likelihood plot with a timescale of 16 seconds, as shown in Figure 92. The 2-stage localisation method outlined in Chapter V was used to automatically choose a time for the twitch, selecting 720.18 seconds (the likelihood plot is generated at the same resolution as the list mode data – 200 μs – and then resampled to $^1/_{100}$ second for the localisation code). The localisation for this case is shown in Figure 92. The actual time of the twitch was approximately 720 seconds – only 'approximately' because the movement was done by hand, and there may be some offset between when the motion was done and the time recorded in the list mode. Indeed the estimated time of the motion is certainly within the margin of error in recording when the motion was performed – felt to be up to a second.

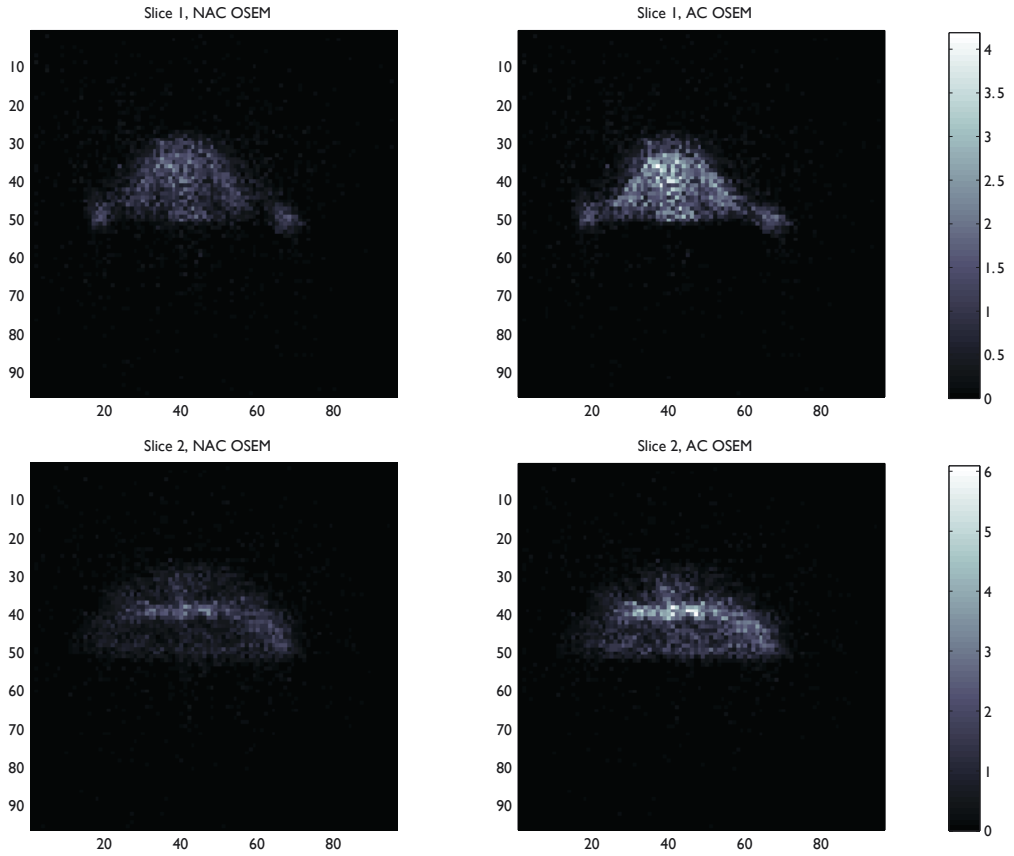

Figure 94 The reconstruction of the data with motion correction. Two cross sections are shown (the rows), without any attenuation corrections (left) and with the approximate attenuation correction described in the text (right).

3. Registration

The twitch detection used all the in plane counts, but for the registration the two cross sections selected above are used again. The rotation is estimated to be 0.45 and 0.562 degrees respectively for the two slices – there should be none – and the translation 10.137mm and 9.475 mm for the two slices, compared to the jig motion of 9 mm. Both registration steps used correlation as a metric. The data from the second position is transformed using these estimates so that it matches that from the first and then is combined to form a single sinogram as shown for one slice in Figure 93.

Note that the registration of the sinogram reduces the pattern of the block gaps because when one sinogram is transformed to remove motion the pattern is also transformed, so

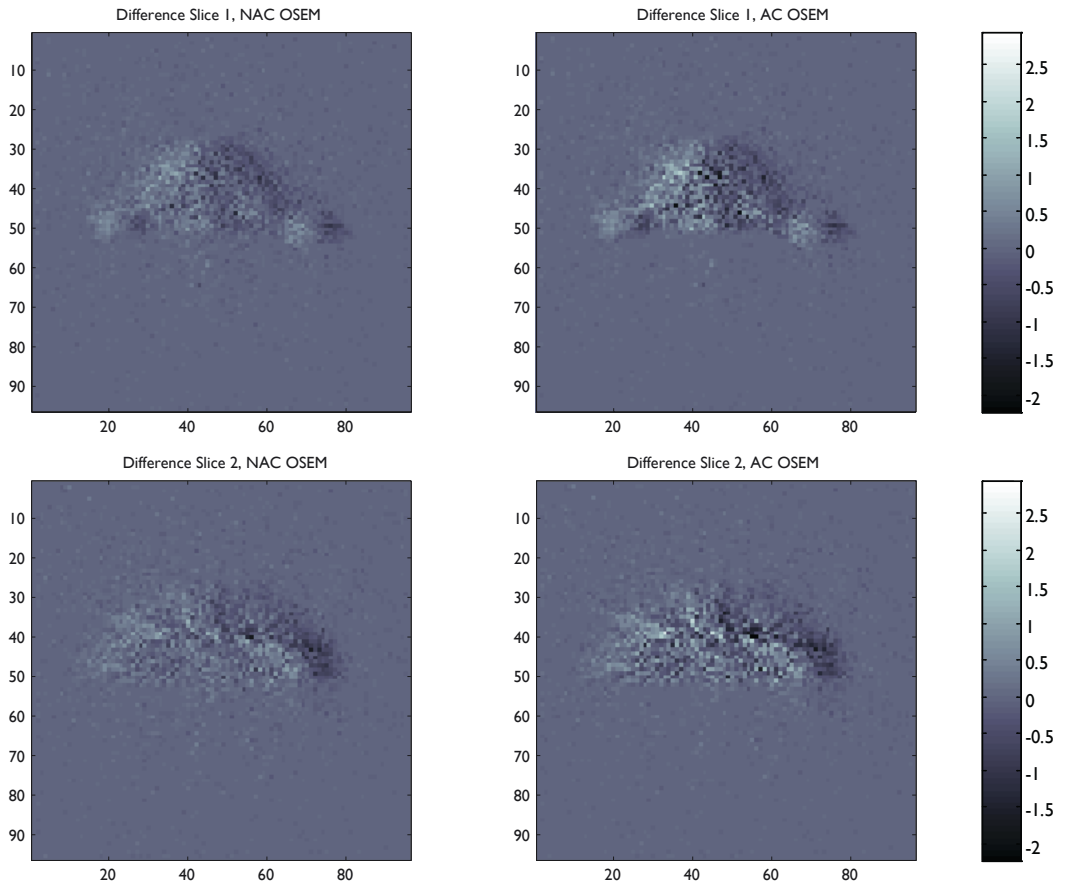

Figure 95 The differences between the reconstructions with and without reconstruction; i.e. Figure 91 - Figure 94.

no longer lines up with that in the reference sinogoram. In practice the motion occurs 'behind' this pattern and thus the registration should as well. This would be a concern using more advanced reconstruction systems that will take into account the block gap pattern of the specific scanner.

4. Reconstruction

Figure 94 shows the result of reconstruction of these corrected sinograms, with and without the approximate attenuation correction. (To aid comparison the colour scale and position on the page have been set to match those of Figure 91.) The differences between the images with the motion estimation & correction and the original ones without are shown in Figure 95.

There is a visible increase in the details that can be seen within the rat thanks to the motion correction, and the artefacts from the approximate attenuation correction are reduced. In short, the benefits of removing a twitch are clear, and had this been a dynamic study without the correction the scan might well have been useless, although demonstrating this is left for future experimental work.

That succinctly illustrates the motivation for dealing with twitches, but the key point is that it was possible to split the single list mode file up into two sections corresponding to distinct poses. That was only practical by operating in the sinogram domain, where the statistical properties of the data a well defined and so a meaningful measure of self consistency could be computed.

> *In conclusion, using the method developed over the past 4 years and presented in this thesis it is now possible to address the problems caused by abrupt motions in real scans, and to do so with no impact on the scanning process itself... something not possible 232 pages ago.*

Appendix A: Programming

This appendix describes the way that some of the methods discussed in the preceding chapters were programmed. Although the programming work done for this thesis cannot be considered to be authoritative, two projects are described to illustrate how they were done, and because the contrast between them proved an instructive experience regarding programming. There were many other pieces of software written along the way; it would not be interesting to discuss all of them.

1. Development Platform Choices

C++ was chosen to implement the algorithm for use with real data because the number of packets involved in a typical PET scan compels the use of a high-performance language with scope for optimization. This ruled out the use of Matlab, which was often used initially for prototyping methods, e.g. the twitch detection with explicit histogramming of simulated datasets. Matlab also has seriously limitation in the way that it handles memory and variable naming.

C++ offers extensive library support, although in practice only a handful of features from the Boost C++ Libraries were used. The language also offers object orientated programming, which was appealing for the compartmentalisation and encapsulation it offers and the subsequent ease of continued development. Admittedly, a large part of the motivation for choosing C++ was that it is a safe choice (support is easily available and the choice would not be controversial) and the project provided the opportunity to learn what is, arguably, the reference against which all other languages are currently compared.

C++ is not, admittedly, the most elegant language. For example, the way templates are handled leads to slow compilation, and approaches to dynamic arrays are basic and require the use of pointers. C# [52], Objective-C [2], Java [32], D [18] etc. are all examples of more recently developed languages, that to some extent aim to serve, more easily, similar use cases as C++ (although, with the exceptions of Objective-C and D, without the same degree of C-style, low level, capability). At the same time there have been a rise in the use of domain-specific languages, for instance PHP [68], Go [33], etc., which are specifically designed to be easy to use (compared to the 'mainstream' languages) in their chosen scenarios.

C++'s support for concurrency (increasingly important nowadays with the rise of multiple core CPUs) is very poor indeed. However, because of the necessarily serial

nature of LMF processing - due to the need for synchronised, in order, delta calculations (as they depend on the state of the sinogram bin counts) - concurrency is not a natural fit for this method anyway, nor is it felt to be needed for adequate performance.

Nothing should be read into the choice of Microsoft Windows XP and Visual Studio C++ Express as a development platform other than the fact that the computers available at the time were running this platform (and the cost of Visual Studio C++ Express: free). In fact, as will be mentioned later, the memory limitations inherent in a 32-bit operating system (and in particular MS Windows, which imposes a further restriction on the address space by reserving the upper 2GB, leaving only 2GB for applications) proved potentially awkward at times. *Note that today the 64-bit versions of Windows 7 are now considered to be the mainstream version.*

2. Twitch Detection Software

The following sections describe the major components within the software system developed to execute the twitch detection algorithm in order to enable experimentation. The rationale, interaction, and limitations of the system as currently implemented will be mentioned. The intention in writing the following is not to propose an optimal approach to the design of the software, rather to record how things have been done and the lessons learnt.

2.1 Classes

2.1.1 LMF Class

The data from the scanner are a binary file with the list mode data and a header file containing the scanning parameters. These need to be wrapped by a class to represent them within the software system, providing access to data stored within. Therefore the LMF class has to read the relevant parameters from the header and provide a means of accessing sections of data from within the scan. This means loading specific sections into memory so that they can be processed – given the size of the raw list mode data (potentially many gigabytes) and the memory limitations of computer systems, in particular 32-bit operating systems, a whole scan cannot be loaded at once – also, for many experiments, there is no need to load an entire scan.

The list mode file contains all the time packets in line with all the other data. Therefore the file must be indexed such so that sections can be accessed by their time, without having to search through the entire file from the beginning looking for the relevant time marker packet. On the first loading of a scan the LMF class generates an index file that maps times to positions in an abridged list mode file (aLMF), generated during the same procedure. The aLMF file contains just the time marks and prompt coincidences

required for the protection algorithm, discarding the myriad of other packets recorded in the full list mode data.

In use, first the index file is searched for the target time. As it just contains time information, it is possible to make a reasonable first guess as to where to start searching in the index file using the target time as a proportion of the scan's overall duration. Having found the relevant time, the abridged list mode data file can be opened and the read pointer move to put the position recorded in the index file. The prompt data can then be loaded into memory. The data is loaded into a special loaded data object, which will be explained next.

2.1.2 Loaded Data Storage Class

During debugging of the software there were some concerns about memory fragmentation because of the large data sizes and low memory availability (with 2 GB per process 'low' is used only in relative terms!). Therefore, rather than hold loaded data as a long contiguous array in memory the loaded data class was designed to store the information in many, smaller, sections – essentially paging the data into memory. The class then provided methods so that the loaded data could be accessed as though it were a single contiguous array. This retains the elegance of being able to access prompts one after another using simple indexing while simultaneously allowing the data to be stored wherever there is space within memory at runtime. *Incidentally, the memory fragmentation turned out not to be the cause of the problems being encountered, however the prospective solution implemented remains a useful idea.*

If one were to be clever, one could design the abridged list mode data and online loaded data storage to match, which would simplify the process of loading data by eliminating the steps that are required presently to convert between the abridged list mode data and the memory structures used online. This would require designing the loaded data object

such that it could be serialise onto disk in a manner that would not depend on which particular chunk of data it stores (as the loaded data object could consist of any section of the scan data, and one load could overlap with the previous loading there can be no specific start or end to load data sections).

2.1.3 Projector Class

The ill named projector class provides a means of storing the geometry and parameters needed to convert the crystal and addresses to sinogram coordinates, and provides the methods to do so. This class is thought to be an awkward corner of implementation: given the chance to do it again, it would probably be better to store the geometry as a struct (within the other classes that require it) then provide the methods to convert the crystal addresses to sinogram co-ordinates either within the client classes or in a specific class to represent the prompt coincidences.

The geometry and parameters that govern how the conversion is done need to be shared between those parts of software system – for instance they would be specified at the beginning of a twitch detection run, and then need to be handed off to the two histogram instances used to store the sinograms during and algorithm.

The concept of using a projector class came about as a means of encapsulating this information, and also the functions needed to do the conversion, in a single package. The desire was to minimise the amount of information needing to be passed as function arguments again and again. Further there was a desire to keep the individual prompts as lightweight as possible, for the sake of performance.

In hindsight there was probably too much caution about this. It would certainly have been possible to have a current prompt instance which has its crystal addresses updated (and only its crystal addresses updated) to change from representing one event to the

next. The geometry could have been represented by a struct and passed by constant reference to functions requiring it. This would have been more elegant as the constant reference attributes of the argument would mean that the geometry struct would actually be shared by all the client instances and owned by the overall algorithm, ensuring that the geometry can not, for instance, be different between the two windows sinograms. Instead, at present, each client class owns a separate projector instance and there is no strong guarantee that the projectors are identical.

2.1.4 Histogram Class

Instances of the histogram class are used to represent sinograms within the software. It owns an instance of a projector and a 3-D storage array (discussed next) and provides the methods needed for creating histograms from sections of data, accessing individual bin counts, and modifying individual bin counts. It also allows the sinogram, or sections thereof, to be exported to disk as a comma separated value file, as a convent means of importing the data into other applications – e.g. Matlab for visualisation.

2.1.5 3D Storage Array Class

This class wraps a standard template library vector providing translation from 3-D indexing (using i, j, and k coordinates). It provides a much easier and more reliable means of storing three-dimensional data than a pure C++ multidimensional array (i.e. an array of arrays of arrays). The additional layer of abstraction provides an opportunity to debug array bound violation errors.

Indeed, it was one such error that motivated the creation of this class. The bug was particularly awkward because it was only detected at the time that the object using the 3-D array was destructed. A faulty piece of pointer arithmetic for one of the lower dimensions resulted in a pointer erroneously pointing to another section of the 3-D

array. As a result when that section was modified there were no immediate ill effects. (The calculation of which column to place a count in resulted in a value that was significantly greater than the number of columns and therefore push the point where you passed the memory for that slice of the sinogram, but not so far as to fall into the memory space of any other variables. This meant that there was no immediate problem, only a heap violation error at the time of destruction.)

Also, this abstraction allows the underlying storage to be changed in future. For example at present a standard template library vector is used as it provides a second layer of bounds checking, over and above the explicit checking of the three-dimensional bounds. However it would be more efficient to remove this extra check now the pointer arithmetic is known to be accurate; it is easy to swap the STL vector for a pure C++ array without any change to code outside this class. Indeed, the checks for the 3-D array bounds could also be disabled in production code – the D language provides specific constructs for doing this sort of thing.

2.1.6 tdRun Class

In what is probably the most overzealous use of object-oriented programming in this work this class serves to represent the overall twitch detector. It is created and setup of all the relevant parameters – timescale of the twitch detection, geometry for the histogramming, etc. It then creates the various instances of the classes described above is needed to run the algorithm. It also provides the methods needed to compute the deltas (including a means to precomputed the values of the deltas such that the online processing can simply look up the relevant values), the window widths accounting for decay correction.

This class also provides the method that actually runs that algorithm, stores the results (a vector containing the likelihood of values for each point in time, and the vector to the values of each point in time, and provides a method that export these vectors

to comma separated value files on disk. Therefore, the routine that would run twitch detector would first create a twitch detector class and then call the method to run that the algorithm between two points in time.

2.2 Build Output

All the classes are stored in a DLL which exports a set of C routines for doing things: for example, histogramming section of data and exporting the resulting sinogram to disk, or running a twitch detection over a section of scan and exporting the likelihood plot.

The rationale behind this design was the C functions exported by the DLL could be called from within Matlab by the 'mex' functionality provided by the Matlab environment. In practice, it proved simpler to import the results into Matlab through .csv files rather than calling these routines directly. As a result these routines exported by the DLL are now driven by small C++ executables which serve to feed parameters into the DLL exported routines, which in turn set up the relevant class instances and drive the process.

2.3 Software Design Regrets

Classes should have been more freely used to represent lower level constructs, in particular prompts. The prompt class could have provided a method to calculate the sinogram coordinates from the crystal addresses using the geometry given to it as a structure through its arguments. Concerns about the impact on performance of using classes at lower levels were overwrought and it would have made for much more elegant software.

As already mentioned, the projector class is bulky and incongruous. The motivation to package the geometry and routines that use the geometry in a single entity could have been better served.

Some things, for example information retrieved from the scans header file or the geometry needed to convert crystal addresses are natural examples of global data. The software might have been more elegant had these been implemented as a handful of singleton classes (i.e. classes where the code enforces the existence of only one instance). Singletons provide pseudo-global storage and would, therefore, have fitted these things better than the classes used – at present it is only careful use that makes for good use.

The interface to the loaded LMF data was written before the queues of tokens used to store events waiting transition or exit the windows, and therefore its interface does not match that of the double ended queues provided by the standard template library. The code would have been easier to follow if the loaded data class provided methods such as `pop()` and `back()`.

The object-oriented programming paradigm was often only used for encapsulation – that is to say it was used to package up data and methods together and to cut down on the number of arguments that had to be passed between functions. As a result the current code largely represents a procedural approach using the object orientated constructs to package the code up and make it tidier. It is now, with a better understanding of object-oriented programming, felt that a more elegant approach might be possible – for example, singletons and a greater role for abstraction. (The latter did prove useful on the handful of occasions mentioned, but as with the previous point about the interface to the loaded LMF data, there were missed opportunities to benefit from it.)

The processing of the twitch detection algorithm is quite quick, and much of the time required to perform experiments is taken up by the process of setting up or loaded data constructs – this seems wasteful, and is a result of the emphasis given to fast online performance at the expense of the overall software architecture (e.g. the lack of a class for prompts).

3. LMF Splicing Software

This software was written to allow two sections of list mode data to be extracted from a source file and spliced together to form a single new list mode file. The objective is to create an idealised, instant, twitch using real data by butting two sections of data from different poses up against one another so that there is a change of position but no data from the time during the motion. The reconstruction software is not able to do this itself and so new list mode data must be created.

3.1 Counters

The processes is complicated by the fact that there are many other pieces of data interleaved into the list mode data. Some of these are counters, recording the number of events observed in different categories (such as by a specific channel of the DAQ system). It is not possible to correctly adjust all of these counters (for example, it is not possible to know which events are handled by which IO channel) and so the result is imperfect.

The counters can be made roughly correct by incrementing those inserted into the output by the same amount as those in the input. This requires keeping track of the value of each sort of counter as the packets are encountered in the input file. When a counter of some sort is encountered its value is compared with the last value for that sort and the change can then be used to generate a new value for the counter packet being inserted into the output. The values (for both input and output) are then sorted for the next time a counter of this type is to processed.

There will be a wrinkle at the join, because it is impossible to work out by how much to increment each counter given that some of the source data is being edited out.

3.2 Contexts

The need to keep track of the values of the different counters in both the input and output gives rise to the concept of a 'context': when a packet is fetched from the input file it comes from the input context, to be put into the output file the packet must be moved from the input context to the output context. The concept of the context is captured in a struct, which provides a group of storage for the current values of all the counters and other things like total number of packets. In keeping with a common convention in C++ a struct is differentiated from a class in that methods are not used with structs (even though they can be, and the only difference between class and struct enforced by the language is the default level of sharing for the data members: public for structs, private for classes – although this is easily overridden).

3.3 Packets

The packets have a natural hierarchy. At the very top level all packets have the same data storage – six bytes. They also have certain tasks that are common, for instance moving a packet out of and into a context, but the exact steps required to do this depend on what type (and subtype in some cases) of packet it is.

Therefore the top level packet class specifies the data storage and a set of ***abstract methods***: member functions without implementations that serve as a placeholder for functionality that will be implemented by other classes derived from this class. A class with abstract methods is called an ***abstract class*** and cannot be instantiated itself. Its purpose is to capture a common ***interface*** to a family ***concrete classes*** (which have implementations for all methods) that are derived from it.

Individual packet types have their own classes that inherit from the basic packet type. Some types themselves have subtypes that may have further subtypes, notably the

counters. There are many types of counters and some of these are further subdivided by, for instance, which DAQ channel they refer to. The top level abstract class helps masks this complexity: in use what is needed is the ability to create a packet object from the bytes in a file and move it between contexts so that it can then be written out to the new file. All of those tasks are captured in the abstract packet class – it is an interface between the logic of 'moving between contexts' and the implementation of how any given packet needs to be modified and the context updated. The implementation is in the classes of the packet subtypes and is separate from the definition of the interface – this is the very nature of ***abstraction***. The result is that the implementation can be developed for each packet type separately and changed at will without disrupting the rest of the code (provided that it still conforms to the interface).

3.3.1 Moving Context

The implementation of the methods for moving between contexts takes as its arguments references to the context object. Declaring the arguments as references means that inside the methods the handle (name) of the context objects is an alias for the original one. In addition to saving the computational time and memory access needed to copy the context into a new variable to be used within the method (as is the case with ordinary function arguments) this means that changes made to the contexts will affect the original context, not just a local copy. Without this any changes made to the context (like updating it with the most recent count value) would be lost when the method ends and the local copy was destroyed.

The implementation of each subtype's methods for moving context also call any other implementations in their hierarchy. This means that common tasks for a family of subtypes (or rather sub-subtypes at least) can be captured at the relevant point in the tree of the hierarchy of packet types. This saves duplicating the common steps and makes the code easier to maintain: if a change must be made to those steps it need only

be made once and it is impossible to forget to make the change to some subtypes – the one change affects all the children in the subtype hierarchy.

3.3.2 Advantages of Separate Packet Types

Inspecting the packet type with raw packets in the list mode file involves examining the bits within the packet, and because of the multiple levels of hierarchy there will be several different sets of bits that may need to be queried. This is how the code was written in the twitch detection software: packets were always kept as 48 bit long 'bitsets', and a library of functions were available to inspect them; for example there was a function that returned true if the packet had the relevant bits set to be a prompt coincidence, as opposed to a delayed coincidence. Having this library of functions available made the main code easier to write and read than doing the inspection directly every time.

The disadvantage to that approach is there is nothing to stop the programmer from using a non-sensical inspection function, continuing the example: the function to determine if a coincidence packet is for a prompt or delayed coincidence could be supplied with a timing packet, and the result of that would be nonsensical.

The function could also check that the packet is a coincidence packet, in which case a return value of 'false' could be interpreted as 'not a prompt coincidence, but anything else'. (Alternatively an ***exception*** could be thrown, but exceptions – a means of handling unexpected conditions by passing a message, and program flow, back up through a stack of function calls – are often more hassle than they are worth.) As usual, there is a desire to be able to make any change (here to the way a coincidence packet is identified) at a single point in the code. This means there would have to be a function to check if a packet is a coincidence, which could be called preemptively by a conscientious programmer as well as being called within the function for identifying prompts.

Having unique types for the different nodes in the packet hierarchy is a far more elegant and safer approach. Only a coincidence packet, or one of prompt coincidence or delayed coincidence (its children in the hierarchy, which would inherit its member functions) would have a method to inspect the packet to see if it were prompt. This is an example of ***encapsulation***, a principle of object orientate programming: data and the methods needed to operated on the data are grouped together. It is impossible for the programmer to ask a non-coincidence packet if it is prompt or not, and the complier will flag this specific problem when the program is built (as opposed to the programmer having to realise the origin of whatever runtime bugs there are). This is an inherent advantage of a ***statically typed*** language like C++, in which the type of every variable is known at compile time.

3.3.3 Polymorphism

This extended hierarchy has a drawback: although in use there is no need to worry about the specific type of a packet (at least not in this application) this is important when working out which class in the hierarchy to instantiate.

If there were to be a need to inspect the type of a packet during use the easy way to do so is by having a 'type' method in the abstract base class that all concrete subclasses must implement, returning their type. This can be used by the program logic to determine what sort of packet it is dealing with and branch program flow accordingly.

When first instantiating a packet the constructor method of the relevant concrete class must be called. This will, unavoidably, require direct inspection of the packet's bits. However, it is desirable to split the inspection up through the type hierarchy, e.g. so that the way of deciding identifying each subtypes of coincidence packet is made by code in that class. That would be, after all, the obvious place to look if a change were made to

the format that changed how the difference between the two was stored in a packet, and in keeping with the concept of encapsulation.

Also, recall that C++ is a strongly typed language, and so when creating a packet instance or acting on it its type must be known. This would create an undesirable branching in the program flow and duplication of very similar code if the process of reading in a packet, moving contexts, and writing it out to the new list mode file had to be written for each subclass in the packet hierarchy.

The solution to this is ***polymorphism***, which in object oriented programming allows an instance to behave differently depending on how it is treated. Here polymorphism avoids the need to treat every sub-type separately. Because all the subtypes inherent from the abstract packet class they can claim to be a 'packet' and be treated as such. So one program flow can read in 'packets', move 'packets' between contexts, and write out 'packets' – recall that the abstract class declares the methods for this, and in doing so requires the concrete classes to provide their implementation, guaranteeing that there will be a way to this whatever flavour of 'packet' is being dealt with at the time.

Should the need arise to branch the program flow based on packet type – e.g. to do twitch detection, which would need the prompt coincidences to have their crystal coordinates queried – then it is possible to do so, by explicitly speaking to the 'packet' as a 'coincidence packet' and thus gaining the ability to use the methods for querying the coordinates. This is the essence of object polymorphism: how the instance is treated determines how it behaves.

3.4 Factory Design Pattern

There is a small trick to making this work, and that is a design pattern known as a ***factory method***: a static method that returns a pointer to an instance of a class. 'Static'

(here... it is a highly reused term in many languages) it means that it is available before a class is instantiated, 'pointer' means that what is returned is an address which allows the instance to be found, rather than all of the data contained within the instance. The reason these aspects are important is that it allows the factory method to worry about how to create the instance, and this includes deciding what subclass to instantiate. By returning the address the code using the factory need not be concerned about the difference in the data at that address. If the actual data were returned it would lead to the problems discussed above in reference to polymorphism because, being a statically typed language, the complier has to know what type of data is being returned (but an address is just an address).

So in this scenario the factory method for packets can inspect the packet and call the appropriate constructor to instantiate the sub-class. The constructor takes care of allocating the space in memory and setting the relevant values. The factory then returns a pointer to the generic 'packet', so the program code can treat all packets the same (unless it specifically needs to treat them differently). In some cases the packet factory method may have to call another factory method: for instance as there are two sorts of coincidence the packet factory must call a coincidence factory which would chose between calling the prompt coincidence constructor or the delayed coincidence constructor – either way it returns a pointer to a 'coincidence', which the packet factory then passes back up the chain to the main program as a pointer to a 'packet' (which of course a pointer to a 'coincidence' is also).

The factory design pattern is, if complicated to explain, very elegant to use and makes it possible to benefit from polymorphism and inheritance: the differences between packets are hidden by the factory, but available later if and when needed. Commonality can be captured in the program at the top of the class hierarchy, without limiting the ability to use specialisations lower down on demand.

Design patterns are a very interesting aspect of programming. To practically use a language as complex as C++ certain conventions must be adopted to make it easy to understand code: as mentioned structs and classes are very similar in C++ but there is a common convention not to include methods in structs, so that when a struct is used it may be assumed to be a 'passive' entity, serving only to group bits of data together. Design patterns are, like conventions, something used by the programmer and not part of the language itself. They are not just recipes for doing common tasks, they capture the common aspects of the solution to problems encountered in a family of tasks. For an introduction to design patterns and description of important examples the reader is refereed to [30].

Design patterns are to software design what algorithms are to solving problems. Here, for instance, the factory pattern has enabled the effective use of polymorphism and inheritance, and thus allowed the software design to capture the relationships between packet types, what is common between some or all of them, and still allow their individual details to be captured.

3.5 IO Classes

In addition to the packet hierarchy and the context classes two classes are used to wrap the input and output files, providing an interface to the input and output file streams of the C++ standard library that understands the concept of packets. This means that the main program flow can operate purely at a packet level – e.g. asking the input file object for the packet, rather than the next six bytes.

The wrapper classes has also been designed to read batches of packets from disk to memory. Past experience has found that choosing the correct batch size can boost performance by allowing the pending packets to be cached in fast memory and amortising the delay in the trip out to disk (especially for mechanical hard disks with the seek

time needed to put the read heads in the correct place) over all the packets in the batch. This is done for both input and output.

3.6 Program Flow

The program starts by setting up the input and output files by instantiating the wrapper classes. The two contexts are also instantiated and several other variables used to keep track of program state – in particular whether packets from the input should be being copied or not.

The program then enters a loop, reading in a packet at a time and updating the input context. When the input context's time reaches the start time for the first section to be copied subsequent packets are moved to the output context after they have updated the input context. They are the written out to disk through the output file interface object. When the input time reaches the end of the first section marked for transcription output is disabled.

The second section is processed likewise. When that is complete, or if the input file has run out of packets, the loop terminates. At this point the output interface object if flushed to ensure any buffered packets are written, the files are closed and the program terminates.

3.6.1 Build Output

In contrast to the twitch detection program here no separate library (DLL) was used – all the functionality was directly built into the executable. This is because it is easy to build a library out of the reusable code (like the hierarchy of packet classes) at a later date if it is needed, and doing so now would only complicate the compilation and linking

steps needed every time the program is built. The slight irony is that with this second project there are aspects much more suitable to reuse in future projects.

3.7 Performance Comparison

As mentioned in §2.3 one of the regrets about the twitch detection project was a failure to use the object orientated paradigm for anything more than providing glorified namesakes. The reason for doing so was, in part, a lack of understanding of how better to use it (for example the factory design pattern). However in addition there was a concern about the overhead of object instantiation that lead to packets being kept as small byte arrays.

The new code is noticeably slower, and by more than can be explained by the need to write to disk as well as read. Profiling the code reveals that much of the time is spent inside the factory method, selecting which constructor to use and running it. More time, in fact, than is spent moving between contexts... the program is spending more time making the program work than it is doing anything useful. There is scope to improve the performance of the packet constructors by changing how they handle the packet data, but even so it does illustrate that there was merit to the concerns about performance in the twitch detection project.

However, of the two approaches the more recent one, properly optimised, is probably preferable. It could be capable of sufficient performance to be used in real time for online twitch detection (the twitch detection code is several times faster than real time) and the proper hierarchy of packet types is much easier to develop programs with, both because of clearer design and less runtime debugging.

Appendix B: Explorations on Twitch Localisation

As highlighted in Chapter V the method of identifying the manifestation of a twitch on the likelihood plot used was only one sufficient to enable experimental work. This appendix outlines some of the other approaches that were tried along the way.

1. Linear Observer

The first exploration into processing the likelihood plot was to convolve it with a kernel describing the 'V' shape. This is an example of a linear observer, in which the inner product between the data and a feature template is taken to produce a decision variable, which is compared against some threshold. Here, convolution is used because the inner product must be taken with the feature template centred on every time in the likelihood plot.

This produces a new plot of the decision variable's value for a feature centred on each point in the likelihood plot. This plot will be high when the likelihood plot resembles the 'V' and low otherwise. The meaning of 'high' and 'low' is relative, of course. Also some threshold is needed to identify 'significantly' high regions, and the twitch time then fixed by taking the maximum within each region. This is the first problem with this approach – the thresholding is a parameter that requires tuning. There are various approaches to this, some with strong theoretical foundations, others more heuristic.

The second problem is more difficult: how to capture the variability of the feature template? An adaptive kernel was tried briefly, in which the slope and 'V' were fitted to the data at each point in the convolution process. This was not without promise as an approach, although it does bias the process towards detecting features, by making the template the best fit to the data available.

This approach was abandoned as other options were explored, largely as a result of its simplicity and difficulty in capturing the range of different sorts of variation in the feature.

§

2. Phase Space Observer

Considering the 'V' shape component of the feature led to the realisation that the shape was much better known than the scale. This prompted considering the problem in phase space (i.e. considering only the phase of the Fourier spectrum of the data). For any given block of the likelihood plot the inner product between the phases of its Fourier spectrum and those corresponding to the 'V' shape give the decision variable.

In addition to treating this as a linear observer in another space this can be considered in terms of Phase Congruency [44], in which it is observed that many prototype 'features' in signals (ramps, impulses, 'V's etc.) correspond to a systematic pattern of phases in frequency space.

The problem with this is that in the absence of a twitch the noise in the likelihood plot can still produce a significant response from the observer. This is, ironically, a common problem with phase based approaches: because the magnitude is ignored, small 'V's due to noise cannot be differentiated from big ones corresponding to features.

One approach to redress this limitation is to allow some use of the magnitude information to suppress 'minor' frequency components. Also, some frequency components may be more significant than others; the relative probability of observing any given phase for a component as a result of noise or a feature will vary. There are various different formulations in the observer framework that incorporate this sort of statistical information into the calculation of the decision variable.

This approach was also abandoned – again, the difficulty in capturing the variation between the slope and 'V' was the main factor.

3. Empirical Mode Decomposition

The recurrent problem about the variation between a slope and a 'V', but the clear knowledge about the duration of the feature motivate a brief exploration of Empirical Mode Decomposition (EMD) [38]. This is a data driven approach to signal analysis, where a signal is split into several constituent oscillatory components and a residual (or trend). The method imposes very few restrictions – there may be any number of components and, in contrast to typical decompositions like the Fourier transform, the components are not predefined. (Another advantage of the method is that it is non-linear, and thus better suited to analysing many real world phenomena – although that is not relevant here.)

The reason that EMD is of interest here is that over the timescale of a block – twice the the twitch detector timescale – the variation due to a slope between two baseline levels cannot appear to be oscillatory: it will be recovered as the residual after the EMD algorithm terminates. A pure 'V' might appear oscillatory at that scale (high at the start, low in the centre, high at the end would appear to be one period of an ongoing sawtooth wave) but would be recovered in the final component of the decomposition. Noise would be recovered in the earlier components as any periodicity would not be expected to have a relationship to the timescale of the twitch detector.

So by reconstituting the signal using only the residual and a few components each block of the likelihood plot could be de-noised in a manner that uses the known scale of the features sought to condition the de-noising to preserve the target features.

Note, however, that this is a de-noising strategy, it does not itself infer anything about the presence of a feature. There is also a fundamental limitation that it operates on a 'block level' and the blocks overlap – therefore if the objective is to de-noise the whole likelihood plot there is a need to address how to combine the blocks together.

It is roughly at this point that the 2-stage Local Variance / Model Based Refinement method described in Chapter V was developed.

4. Weak String

The 2-stage Local Variance / Model Based Refinement works well when there is a single twitch, but extending it to deal with multiple twitches requires an additional stage to identify sections of high variance. The obvious way of doing that – regions above some threshold – and the local variance itself are quite heuristic.

For balance some methods were investigated which would be inherently able to handle multiple twitches and in which the parameters of the twitch localisation would be chosen less heuristically, for example by configuring a cost function whose terms could be related to expectations about the twitches like their number. The first of these explored was the Weak String and Weak Rod model of [7].

The Weak String model is the easiest to understand. Its objective is to recover a piece-wise constant approximation to some data. The points in the new approximation are chosen by an optimization. The cost function is as if each point in the approximation is connected to a point in the data by a spring, so that the approximation will try to follow the data. However, there are also terms that are as if the points in the approximation are connected to each other in a chain by springs – these springs are stretched by the change in value from one point to the next and so will tend to smooth the approximation. The trick is that these smoothing springs are allowed to break, for a fixed penalty the approximation is allowed to have a discontinuity at any point.

The Weak Rod model is similar, but aims to recover a piece-wise linear approximation: the inter-point penalty is not based on gradient but rather curvature (change in

gradient) and instead of breaking the approximation is allowed to 'kink', introducing an discontinuity in the gradient.

The trick in the method is the construction of the cost function such that there is no need for a separate stage in which the discontinuities at placed. This would significantly complicate matters, as the optimal values of the approximation points would be depended on where the discontinuities were, and vice versa. Instead there is only one optimization which directly finds the new values. The locations of the discontinuities can then be recovered by inspecting the approximation (in short, if the cost of being continuous between two points exceeds the penalty of being discontinuous then there is a discontinuity there).

The Weak String and Weak Rod methods are very quick, and controlled by designing the cost function – setting terms for how smooth the approximation should be, how closely it should follow the data, and the cost of introducing various discontinuities. For twitch localisation the weak rod model is more appealing: it could recover the flat, no twitch regions and the two sloping sections of the likelihood plot surrounding the twitch time. The twitch time could be found by looking for constellations of discontinuities (knowing that the timescale of the twitch detector approximately controls their spacing).

4.1 Limitations

A caveat is that the Weak String and Weak Rod methods are about recovering smooth data, allowing for discontinuities; they are not about locating them. Locating the discontinuities is a separate step, analysing the approximation. The methods are therefore inherently unsuited to feature localisation.

Until the constellations of discontinuities are sought no use is made of the prior knowledge about how twitches manifest on the likelihood plot. This is an inherent limitation of the approach as the models are purely local: the cost functions only consider each point's relationship to its immediate neighbours and corresponding data point. Indeed, this is why the optimization must iterate thorough several times. Introducing long range interactions would be needed to incorporate the prior knowledge of the timescale. Further the placement of the discontinuities would not be independent – the cost of doing so would have to be lower if others are nearby. It would also be desirable to enforce the approximation to be constant unless there is an adjacent sloping region, but again, this is not possible in the framework without fundamental modification.

5. Dijkstra Based Approaches

The elegance of the Weak String and Weak Rod methods is that they fold the discrete optimization problem of placing the discontinuities into the continuous optimization process needed to set the values of the approximation. Here, what is actually wanted is a discrete optimization problem: labelling sections as continuous or discontinuous. For an introduction to discrete optimization, see [22].

The task at hand suggests looking at graph labelling methods. A graph is used to capture the structure of a problem which is then solved by one of the available methods. Given a the graph structure determined by the approach taken to solve the type of problem the specific problem is captured by assigning weights to the links between nodes (or 'edges'). The process by which the edge weights are calculated is very flexible and incorporate exceptions etc. not easily captured mathematically, so the resulting cost function can be more bespoke than those generally encountered in continuous optimization methods. However the weights are assigned, once they have been the solver is able to take the graph and produce the optimal result.

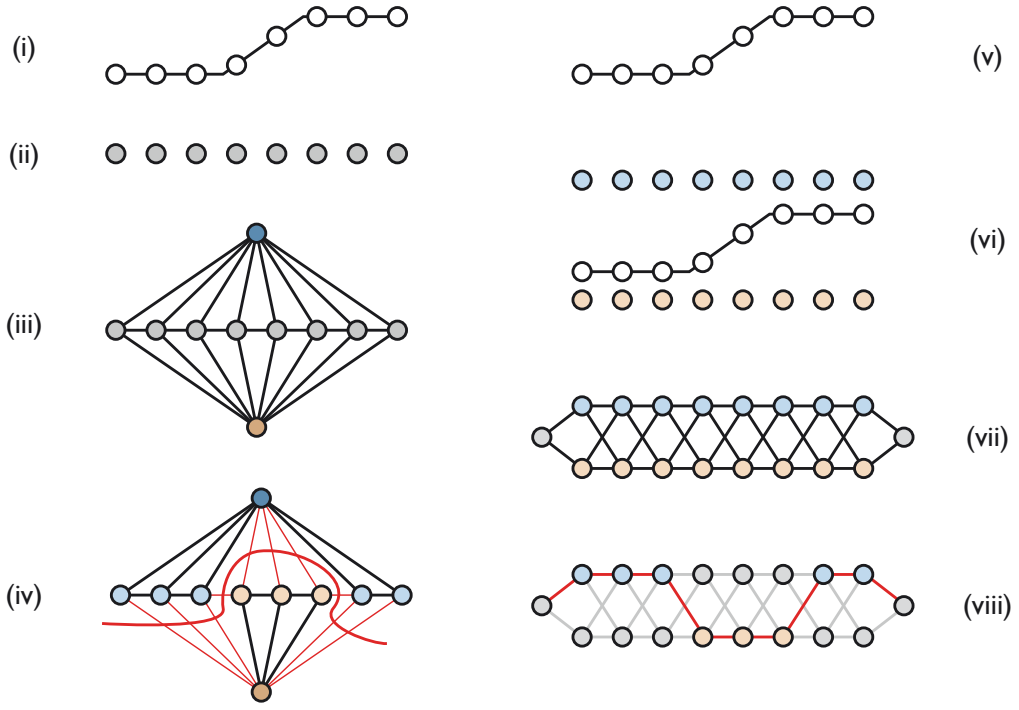

Figure 96 Graphs for discontinuity labelling using graph cuts (left) and Dijkstra's algorithm (right). From the top, for graph cuts: (i) the likelihood plot is split into sections (potentially overlapping). (ii) Nodes for the graph relate to the segments of the likelihood plot. (iii) Nodes are connected to two terminal nodes, one for each label. (iv) The nodes are labelled buy cutting links. And for Dijkstra: (v) the likelihood plot is split into sections (potentially overlapping). (vi) Two nodes are created for each segment. (vii) The nodes are connected together, and to two terminal nodes. (viii) The algorithm find the shortest path, each section is lables by which of its nodes is used in the path.

One approach is that of Graph Cuts. In this formulation the points are labelled by partitioning the graph into two halves, with the cost of doing so being the weights of the links severed by the partition. Although the structure of the graph is obvious – each node representing one or more points on the likelihood plot – assigning costs to the edges is less so: consider the combination of links cut in the example of graph cuts shown in Figure 96 and the extra cuts needed to include another node, adjacent or otherwise. The mix of edges cut by each does not lend itself to an easy allocation of costs.

An alternative approach is to have two nodes on the graph for each point (or group of points) on the likelihood plot. One node represents labelling that point as discontinuous, the other labelling it continuous, as also shown in Figure 96. The cost of doing so is used to set the weight of the edges entering the node. The edges can have further costs added to them as well: for instance a penalty for switching label. Then the shortest

path (taking the edge weights to be analogous to distances) through the graph is sought by the solver. For each point (or group of points) on the likelihood plot the label can be found by seeing which of the nodes corresponding to that point the shortest path goes through. The shortest path through this sort of graph can easily be found using the Dijkstra algorithm [19].

5.1 Grouping Points

In order to capture the information about the duration of a twitch's effect on the likelihood plot each node of the graph refers to a section or 'neighbourhood' of the likelihood plot of that duration. This is necessary anyway as a single point cannot be discontinuous on its own. However, as there is no way to arrange the division into neighbourhoods such that the start and end of the manifestations of a twitches will correspond to the boundary between neighbourhoods.

This compels either using neighbourhoods smaller than the duration of a twitch's effect and looking for a cluster of discontinuous neighbourhoods or overlapping the neighbourhoods, which will also likely produce a cluster of discontinuous neighbourhoods. Both these approaches also allow for the duration of a disruption to be extended a little, for instance to account for a non-instant motion.

The overlapping neighbourhoods approach was tried, using the variance within each neighbourhood as the basis for the cost of labelling it. Penalty terms were added to the cost of graph edges transitioning from continuous to discontinuous labelling, to capture the assumption that twitches were rare. Further cost terms were added to penalise extending discontinuous regions by declaring adjacent neighbourhoods to be discontinuous as well, but this term was less than the cost of transitioning from the continuous label as it should be preferable to extend a discontinuous region that creating a separate one. The results were sensitive to the tuning of the cost terms, and tended to produce

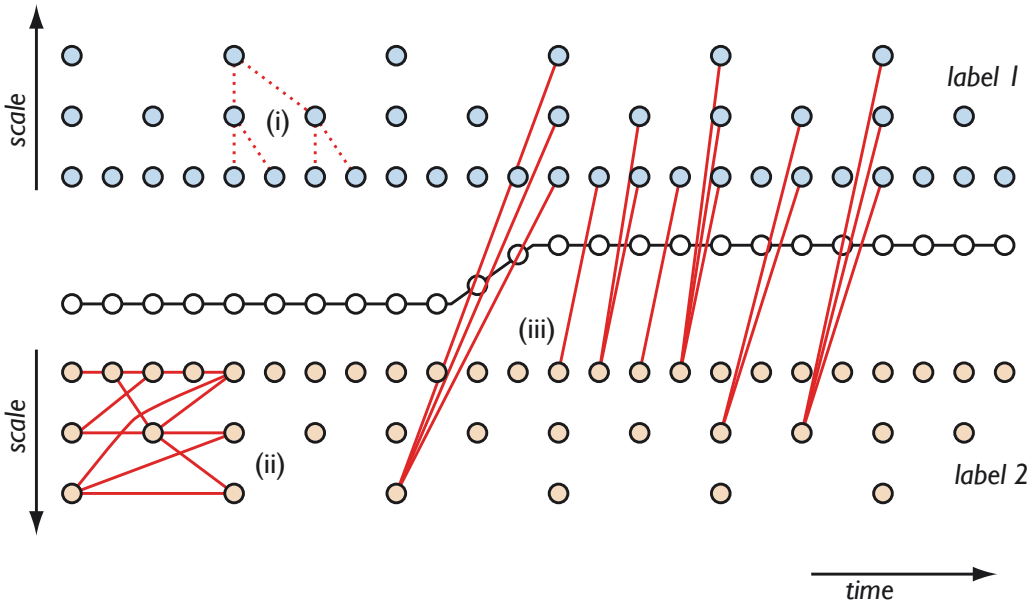

Figure 97 The multiple scale graph for the multi-scale Dijkstra method. Not all of the graph is shown for, clarity. The hierarchy of nodes is shown (i), one instance of the pattern of the connections between nodes of the same label (ii), and examples of the type of connections that allow the label to change (iii). In short: node cannot connect to their children or parents, only siblings or neighbours, and the same applies when crossing label. Labeling a node implies that all its children share that label, so this prevents redundant or contradictory labeling. As in the original Dijkstra case there are two sides to the graph, one for each label, and nodes represent sections of the likelihood plot, possibly overlapping.

overly large regions labelled as discontinuous, probably as a result of the overlapping neighbourhoods.

5.2 Multi-scale graph

However, the continuous regions between twitches would be expect to be much larger – in fact this is another characteristic of twitches that has, up to here, not been used in trying to localises twitches on the likelihood plot. This suggested modifying the Dijkstra based approach to allow larger neighbourhoods to be considered.

The resulting graph structure is shown in Figure 97. In addition to having two sides, one set of nodes for each label, there is a hierarchy of nodes corresponding to scale. The nodes are connected such that if a path through the graph enters a node it cannot proceeded to another node who's neighbourhood would overlap – so neither a child nor

parent in the hierarchy (or those in the corresponding hierarchy on the other side of the graph.

This means that the continuous regions can consider large sections in one go. Costs can be set up to reward this behaviour, and penalise labelling as discontinuous regions of the wrong size. Note that there is no longer any need for overlapping neighbourhoods. The optimization algorithm will select the optimal scale at which to consider any given section of the likelihood plot at the same time as choosing the labels.

Costs can be allocated based on the data in a neighbourhood, the scale or label of the neighbourhood and for transitions between them. Although the meaning of each is very initiative the number of different contributions to cost makes the process of choosing all the terms quite involved.

Initial results were quite encouraging. Compared to the single scale approach it seems better at limiting discontinuous labels to the sections of the likelihood plot where there is a disruption form a twitch, as would be expected from being better able to consider a long section of continuos data as a long section.

However these approaches would take further development to be able to perform as well as the 2-stage Local Variance / Model Based Refinement approach, which has the benefit of producing biases that are much easier to understand and correct than an optimization based approach would. Perhaps these approaches could eventually serve as a replacement for the first stage, the local variance, for identifying the number of twitches and their approximate locations.

Bibliography

[1] R. Allemand, C. Gresset & J. Vacher. **Potential Advantages of a Cesium Fluoride Scintillator for a Time-of-Flight Positron Camera.** *Journal of Nuclear Medicine, vol. 21, no. 2,* pages 153–155, 1980.

[2] Apple Computer **Web Page: The Objective-C Programming Language** http://developer.apple.com/library/mac/#documentation/Cocoa/Conceptual/ObjectiveC/Introduction/introObjectiveC.html#//apple_ref/doc/uid/TP30001163 *Last checked 14 July 2011*

[3] J. L. Barron, D. J. Fleet & S. S. Beauchemin. **Performance of Optical Flow Techniques.** *International Journal of Computer Vision, vol. 12, no. 1,* pages 43–77, 1994.

[4] C. P. Behrenbruch, K. Marias, P. A. Armitage, M. Yam, N. Moore, R. E. English, J. Clarke & M. Brady. **Fusion of Contrast-enhanced Breast MR and Mammographic Imaging Data.** *Medical Image Analysis, vol. 7, no. 3,* pages 311 – 340, 2003.

[5] T. Beyer, D. W. Townsend, T. Brun, P. E. Kinahan, M. Charron, R. Roddy, J. Jerin, J. Young, L. Byars & R. Nutt. **A Combined PET/CT Scanner for Clinical Oncology.** *Journal of Nuclear Medicine, vol. 41, no. 8,* pages 1369–1379, 2000.

[6] B. Bendriem & D. W. Townsend, editors. **The Theory and Practice of 3D PET**, *Volume 32 of Developments in Nuclear Medicine.* Kluwer Academic Publishers, 1998.

[7] A. Blake & A. Zisserman. **Visual Reconstruction.** MIT Press, 1987.

[8] R. G. Boyes, D. Rueckert, P. Aljabar, J. Whitwell, J. M. Schott, D. L.G. Hill & N. C. Fox. **Cerebral Atrophy Measurements using Jacobian Integration: Comparison with the Boundary Shift Integral.** *NeuroImage, vol. 32, no. 1,* pages 159 – 169, 2006.

[9] F. Buther, M. Dawood, L. Stegger, F. Wubbeling, Michael Schafers, O. Schober & K. P. Schafers. **List Mode–Driven Cardiac and Respiratory Gating in PET.** *Journal of Nuclear Medicine, vol. 50, no. 5,* pages 674–681, May 2009.

[10] R. E. Carson & K. Lange. **A Statistical Model for Positron Emission Tomography: Comment.** *Journal of the American Statistical Association, vol. 80, no. 389,* pages 20–22, March 1985.

[11] CERMEP. **Web Page: Database of simulated [18F]FDG, [18F]Dopa and [11C]Raclopride volumes,** http://sorteo.cermep.fr/database.html 2011. *Last checked 13 July 2011*

[12] CERMEP. **Web Page: PET-SORTEO.** http://sorteo.cermep.fr/ *Last checked 11 July 2011*

[13] L. T. Chang. **A Method for Attenuation Correction in Radionuclide Computed Tomography.** *IEEE Transactions on Nuclear Science, vol. 25, no. 1,* pages 638 –643, February 1978.

[14] J. Cizek, K. Herholz, S. Vollmar, R. Schrader, J. Klein & W-D. Heiss. **Fast and Robust Registration of PET and MR Images of Human Brain.** *NeuroImage, vol. 22, no. 1,* pages 434–442, May 2004.

[15] J. Dauguet, A.-S. Herard, J. Declerck & T. Delzescaux. **Locally constrained cubic B-spline deformations to control volume variations.** *In ISBI '09. IEEE International Symposium on Biomedical Imaging: From Nano to Macro,* pages 983 –986, July 2009.

[16] M. Dawood, F. Buther, Xiaoyi Jiang & K.P. Schafers. **Respiratory Motion Correction in 3-D PET Data With Advanced Optical Flow Algorithms.** *IEEE Transactions on Medical Imaging, vol. 27, no. 8,* pages 1164 –1175, Aug. 2008.

[17] A. P. Dempster, N. M. Laird & D. B. Rubin. **Maximum Likelihood from Incomplete Data via the EM Algorithm.** *Journal of the Royal Statistical Society B, vol. 39,* pages 1–38, 1977.

[18] Digital Mars **Webpage: The D Programming Language Specification** http://www.digitalmars.com/d/2.0/spec.html *Last checked 14 July 2011*

[19] E. W. Dijkstra. **A Note on Two Problems in Connexion with Graphs.** *Numerische Mathematik, vol. 1,* pages 269–271, 1959.

[20] Y. E. Erdi, S. A. Nehmeh, T. Pan, A. Pevsner, K. E. Rosenzweig, G. Mageras, E. D. Yorke, H. Schoder, W. Hsiao, O. D. Squire, P. Vernon, J. B. Ashman, H. Mostafavi, S. M. Larson & J. L. Humm. **The CT Motion Quantitation of Lung Lesions and Its Impact on PET-Measured SUVs.** *J Nucl Med, vol. 45, no. 8,* pages 1287–1292, 2004.

[21] S. Espana, J. L. Herraiz, E. Vicente, J. J. Vaquero, M. Desco & J. M. Udias. **PeneloPET, a Monte Carlo PET Simulation Tool Based on PENELOPE: Features and Validation.** *Physics in Medicine and Biology, vol. 54, no. 6,* page 1723, 2009.

[22] P.F. Felzenszwalb & R. Zabih. **Dynamic Programming and Graph Algorithms in Computer Vision.** *IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 33, no. 4,* pages 721 –740, April 2011.

[23] J. Fessler et al. **Webpage: Image Reconstruction Toolbox** http://www.eecs.umich.edu/~fessler/code/ *Last checked 10 July 2010*

[24] E. E. Fitchard, J. S. Aldridge, K. Ruchala, G. Fang, J. Balog, D. W. Pearson, G. H. Olivera, E. A. Schloesser, D. Wenman, P. J. Reckwerdt & T. R. Mackie. **Registration Using Tomographic Projection Files.** *Physics in Medicine and Biology, vol. 44, no. 2,* pages 495–507, 1999.

[25] E. E. Fitchard, J. S. Aldridge, P. J. Reckwerdt & T. R. Mackie. **Registration of Synthetic Tomographic Projection Data Sets Using Cross-correlation.** *Physics in Medicine and Biology, vol. 43, no. 6,* pages 1645–1657, 1998

[26] E. E. Fitchard, J. S. Aldridge, P. J. Reckwerdt, G. H. Olivera, T. R. Mackie & A. Iosevic. **Six Parameter Patient Registration Directly From Projection Data.** *Nuclear Instruments and Methods in Physics Research, vol. 421, no. 1-2,* pages 342–351, January 1999.

[27] R. Fulton, I. Nickel, L. Tellmann, S. Meikle, U. Pietrzyk & H. Herzog. **Event-by-Event Motion Compensation in 3D PET.** *In IEEE Nuclear Science Symposium Conference Record, volume 5,* pages 3286–3289 Vol.5, 2003.

[28] R.R. Fulton, L. Tellmann, U. Pietrzyk & H. Herzog. **Compensation for Head Movement in 3D PET.** *In IEEE Nuclear Science Symposium Conference Record, volume 4,* pages 2013 –2017, 2001.

[29] R. Fulton. **Correction for Patient Head Movement in Emission Tomography.** *PhD thesis, University of Technology, Sydney,* 2000.

[30] E. Gamma, R. Helm, R. Johnson & J. Vlissides. **Design Patterns: Elements of Reusable Object-Oriented Software.** Addison Wesley, 1994.S. German & D.E. McClure. **Bayesian Image Analysis: An Application to Single Photon Emission Tomography.** *In Proceedings of the American Statistical Association, 1985,* pages 12–18, 1985

[31] R Graham, D Knuth, O Patashnik **'Concrete Mathematics'** Addison Wesley, 2nd printing, December 1988

[32] J. Gosling, B. Joy, G. Steele & G. Bracha. **The Java Language Specification.** Addison Wesley, 3rd edition, 2005.

[33] Google **Webpage: The Go Programming Language Specification** http://golang.org/doc/go_spec.html *Last checked 14 July 2011*

[34] K. L. Gould, T. Pan, C. Loghin, N. P. Johnson, A. Guha & S, Sdringola. **Frequent Diagnostic Errors in Cardiac PET/CT Due to Misregistration of CT Attenuation and Emission PET Images: A Definitive Analysis of Causes, Consequences, and Corrections.** *J Nucl Med, vol. 48, no. 7,* pages 1112–1121, 2007.

[35] J.S. Goddard, S.S. Gleason, M.J. Paulus, S. Majewski, V. Popov, M. Smith, A. Weisenberger, B. Welch & R. Wojcik. **Real-time Landmark-based Unrestrained Animal Tracking System for Motion-corrected PET/SPECT Imaging.** *In IEEE Nuclear Science Symposium Conference Record, 2002, volume 3,* pages 1534–1537 vol.3, 2002.

[36] R.N. Gunn, S.R. Gunn & V.J. Cunningham. **Positron Emission Tomography Compartmental Models.** *Journal of Cerebral Blood Flow and Metabolism, vol. 21, no. 6,* pages 635–652, 2001.

[37] H.M. Hudson & R.S. Larkin. **Accelerated Image Reconstruction Using Ordered Subsets of Projection Data**. *IEEE Transactions on Medical Imaging, vol. 13, no. 4,* pages 601 –609, December 1994.

[38] N. E. Huang, Z. Shen, S. R. Long, M. C. Wu, H. H. Shih, Q. Zheng, N-C. Yen, C. C. Tung & H. H. Liu. **The Empirical Mode Decomposition and the Hilbert Spectrum for Nonlinear and Non-stationary Time Series Analysis.** *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences, vol. 454, no. 1971,* pages 903–995, 1998.

[39] S. Jan, G. Santin, D. Strul, S. Staelens, K. Assie, D. Autret, S. Avner, R. Barbier, M. Bardies, P. M. Bloomfield, D. Brasse, V. Breton, P. Bruyndonckx, I. Buvat, A. F. Chatziioannou, Y. Choi, Y. H. Chung, C. Comtat, D. Donnarieix, L. Ferrer, S. J. Glick, C. J. Groiselle, D. Guez, P-F. Honore, S. Kerhoas-Cavata, A. S. Kirov, V. Kohli, M. Koole, M. Krieguer, D. J. van der Laan, F. Lamare, G. Largeron, C. Lartizien, D. Lazaro, M. C. Maas, L. Maigne, F. Mayet, F. Melot, C. Merheb, E. Pennacchio, J. Perez, U. Pietrzyk, F. R. Rannou, M. Rey, D. R. Schaart, C. R. Schmidtlein, L. Simon, T. Y. Song, J-M. Vieira, D. Visvikis, R. Van de Walle, E. Wieers & C. Morel. **GATE: a Simulation Toolkit for PET and SPECT.** *Physics in Medicine and Biology, vol. 49, no. 19,* page 4543, 2004.

[40] N. Joshi. **Non-Parametric Probability Density Function Estimation For Medical Images.** *PhD thesis, University of Oxford,* 2007.

[41] D. J. Kadrmas, M. E. Casey, M. Conti, B. W. Jakoby, C. Lois & D. W. Townsend. **Impact of Time-of-Flight on PET Tumor Detection.** *Journal of Nuclear Medicine, vol. 50, no. 8,* pages 1315–1323, August 2009.

[42] J. S. Karp, S. Surti, M. E. Daube-Witherspoon & G. Muehllehner. **Benefit of Time-of-Flight in PET: Experimental and Clinical Results.** *J Nucl Med, vol. 49, no. 3,* pages 462–470, 2008.

[43] P. E. Kinahan, D. W. Townsend, T. Beyer & D. Sashin. **Attenuation Correction for a Combined 3D PET/CT Scanner.** *Medical Physics, vol. 25, no. 10,* pages 2046–2053, 1998.

[44] P. Kovesi. **Phase Congruency: A Low-Level Image Invariant.** *Psychological Research, vol. 64,* pages 136–148, 2000.

[45] E. Liapi, J. H. Geschwind, M. Vali, A. A. Khwaja, V. Prieto-Ventura, M. Buijs, J. A. Vossen, S. Ganapathy & R. L. Wahl. **Assessment of Tumoricidal Efficacy and Response to Treatment with 18F-FDG PET/CT After Intraarterial Infusion with the Antiglycolytic Agent 3-Bromopyruvate in the VX2 Model of Liver Tumor.** *Journal of Nuclear Medicine, vol. 52, no. 2,* pages 225–230, 2011.

[46] Weiguo Lu & Thomas R Mackie. **Tomographic motion detection and correction directly in sinogram space.** *Physics in Medicine and Biology, vol. 47, no. 8,* page 1267, 2002.

[47] F. Maes, A. Collignon, D. Vandermeulen, G. Marchal & P. Suetens. **Multimodality Image Registration by Maximization of Mutual Mnformation.** *IEEE Transactions on Medical Imaging, vol. 16, no. 2,* pages 187–198, 1997.

[48] J. B. A. Maintz & M. A. Viergever. **A Survey of Medical Image Registration.** *Medical Image Analysis, vol. 2, no. 1,* pages 1 – 36, 1998.

[49] A. Matsumura, S. Mizokawa, M. Tanaka, Y. Wada, S. Nozaki, F. Nakamura, S. Shiomi, H. Ochi & Y. Watanabe. **Assessment of microPET Performance in Analyzing the Rat Brain Under Different Types of Anesthesia: Comparison Between Quantitative Data Obtained with microPET and Ex-Vivo Autoradiography.** *NeuroImage, vol. 20, no. 4,* pages 2040 – 2050, 2003.

[50] A. McLennan, A. Reilhac & M. Brady. **SORTEO: Monte Carlo-based Simulator with List- Mode Capabilities.** *In Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBC, 2009,* pages 3751 –3754, Sept. 2009.

[51] A. J. Montgomery, K. Thielemans, M. A. Mehta, F. Turkheimer, S. Mustafovic & P. M. Grasby. **Correction of Head Movement on PET Studies: Comparison of Methods.** *J Nucl Med, vol. 47, no. 12,* pages 1936–1944, 2006.

[52] Microsoft **C# Language Specification** http://download.microsoft.com/download/3/8/8/388e7205-bc10-4226-b2a8-75351c669b09/CSharp%20Language%20Specification.doc *Last checked 14 July 2011*

[53] W.W. Moses. **Time of Flight in PET Revisited.** *IEEE Transactions on Nuclear Science, vol. 50, no. 5,* pages 1325–1330, 2003.

[54] Y. Nakamoto, B. B. Chin, C. Cohade, M. Osman, M. Tatsumi & R. L. Wahl. **PET/CT: Artifacts Caused by Bowel Motion.** *Nuclear Medicine Communications, vol. 25, no. 3,* 2004.

[55] S. A. Nehmeh, Y. E. Erdi, C. C. Ling, K. E. Rosenzweig, O. D. Squire, L. E. Braban, E. Ford, K. Sidhu, G. S. Mageras, S. M. Larson & J. L. Humm. **Effect of Respiratory Gating on Reducing Lung Motion Artifacts in PET Imaging of Lung Cancer.** *Medical Physics, vol. 29, no. 3,* pages 366–371, 2002.

[56] S. A. Nehmeh, Y. E. Erdi, K. E. Rosenzweig, H. Schoder, S. M. Larson, O. D. Squire & J. L. Humm. **Reduction of Respiratory Motion Artifacts in PET Imaging of Lung Cancer by Respiratory Correlated Dynamic PET: Methodology and Comparison with Respiratory Gated PET.** *Journal of Nuclear Medicine, vol. 44, no. 10,* pages 1644–1648, 2003.

[57] P. H. G. Nordberg, J. Declerck & J. M. Brady. **Pre-reconstruction Rigid Body Registration for Positron Emission Tomography.** *In Proceedings of the 2010 Medical Image Understanding and Analysis Conference.* University of Warwick, July 2010.

[58] P. H. G. Nordberg, J. Declerck & J. M. Brady. **Pre-reconstruction Rigid Body Registration for Positron Emission Tomography: An Initial Validation Against Ground Truth.** *In Proceedings of the 32nd Annual International Conference of the IEEE Engineering in Medicine and Biology Society,* September 2010.

[59] K. Oda, H. Toyama, K. Uemura, Y. Ikoma, Y. Kimura & M. Senda. **Effect of reconstruction algorithm on parametric images in PET dynamic study: comparison between FBP and OS-EM.** *In IEEE Nuclear Science Symposium, 1999. Conference Record.* pages 1542 –1545. 1999.

[60] M. M. Osman, C. Cohade, Y. Nakamoto & R. L. Wahl. **Respiratory Motion Artifacts on PET Emission Images Obtained using CT Attenuation Correction on PET-CT.** European Journal of Nuclear Medicine and Molecular Imaging, vol. 30, pages 603–606, 2003. 10.1007/s00259-002-1024-x.

[61] V.Y. Panin, F. Kehren, C. Michel & M. Casey. **Fully 3-D PET Reconstruction with System Matrix Derived from Point Source Measurements.** *IEEE Transactions on Medical Imaging, vol. 25, no. 7,* pages 907–921, 2006.

[62] V.Y. Panin, F. Kehren, H. Rothfuss, D. Hu, C. Michel & M.E. Casey. **PET Reconstruction with System Matrix Derived from Point Source Measurements.** *IEEE Transactions on Nuclear Science, vol. 53, no. 1,* pages 152–159, 2006.

[63] L. Parra & H.H. Barrett. **List-Mode Likelihood: EM Algorithm and Image Quality Estimation Demonstrated on 2-D PET.** *IEEE Transactions on Medical Imaging, vol. 17, no. 2,* pages 228 –235, April 1998.

[64] F. Peyrin, M. Zaim & R. Goutte. **Multiscale Reconstruction of Tomographic Images.** *In Proceedings of the IEEE-SP International Symposium Time- Frequency and Time-Scale Analysis,* pages 219–222, 1992.

[65] M. E. Phelps. **PET: Molecular Imaging and its Biological Applications.** Springer-Verlag, 1 edition, 2004.

[66] M. E. Phelps, E. J. Hoffman, N. A. Mullani & M. M. Ter-Pogossian. **Application of Annihilation Coincidence Detection to Transaxial Reconstruction Tomography.** *Journal of Nuclear Medicine, vol. 16, no. 3,* pages 210–224, 1975.

[67] M. E. Phelps, E. J. Hoffman, S. Huang & D. E. Kuhl. **ECAT: A New Computerized Tomographic Imaging System for Positron-Emitting Radiopharmaceuticals.** *Journal of Nuclear Medicine, vol. 19, no. 6,* pages 635–647, 1978.

[68] The PHP Group **Webpage: PHP Language Reference** http://php.net/manual/en/langref.php *Last checked 14 July 2011*

[69] Y. Picard & C.J. Thompson. **Motion Correction of PET Images Using Multiple Acquisition Frames.** *IEEE Transactions on Medical Imaging, vol. 16, no. 2,* pages 137–144, 1997.

[70] U. Pietrzyk, K. Herholz, A. Schuster, H-M. v. Stockhausen, H. Lucht & W-D. Heiss. **Clinical Applications of Registration and Fusion of Multimodality Brain Images from PET, SPECT, CT, and MRI.** *European Journal of Radiology, vol. 21, no. 3,* pages 174 – 182, 1996.

[71] J. Qi & R. H. Huesman. **List Mode Reconstruction for PET with Motion Compensation: A Simulation Study.** *In IEEE International Symposium on Biomedical Imaging, 2002. Proceedings.* pages 413–416, 2002.

[72] A. Reilhac, G. Batan, C. Michel, C. Grova, J. Tohka, D.L. Collins, N. Costes & A.C. Evans. **PET-SORTEO: Validation and Development of Database of Simulated PET Volumes.** *IEEE Transactions on Nuclear Science, vol. 52, no. 5,* pages 1321 – 1328, Oct. 2005.

[73] A. Reilhac, C. Lartizien, N. Costes, S. Sans, C. Comtat, R.N. Gunn & A.C. Evans. **PET- SORTEO: a Monte Carlo-based Simulator with High Count Rate Capabilities.** *IEEE Transactions on Nuclear Science, vol. 51, no. 1,* pages 46 – 52, Feb. 2004.

[74] D. Schottlander, A. Louis, J. Declerck & M. Brady. **Preliminary Evaluation of a Kinetic Parameter Estimator with Application to Direct Parametric Reconstruction.** *In 3rd IEEE International Symposium on Biomedical Imaging: Nano to Macro, 2006.* pages 932 –935, April 2006.

[75] Y. Shao, S. R Cherry, K. Farahani, K. Meadors, S. Siegel, R. W Silverman & P. K Marsden. **Simultaneous PET and MR Imaging.** *Physics in Medicine and Biology, vol. 42, no. 10,* page 1965, 1997.

[76] L. A. Shepp & Y. Vardi. **Maximum Likelihood Reconstruction for Emission Tomography.** *IEEE Transactions on Medical Imaging, vol. 1, no. 2, pages 113 –122,* October. 1982.

[77] L. A. Shepp & B. F. Logan. **The Fourier Reconstruction of a Head Section.** *IEEE Transactions on Nuclear Science, vol. 21,* pages 21–43, 1974.

[78] N. Smith, I. Meir, G. Hale, R. Howe, L. Johnson, P. Edwards, D. Hawkes, M. Bidmead & D. Landau. **Real-time 3D Surface Imaging for Patient Positioning in Radiotherapy.** *International Journal of Radiation Oncology Biology Physics, vol. 57, no. 2, Supplement 1,* pages S187 – S187, 2003.

[79] S. M. Smith, N. De Stefano, M. Jenkinson & P. M. Matthews. **Normalized Accurate Measurement of Longitudinal Brain Change.** *Journal of Computer Assisted Tomography., vol. 25, no. 3,* pages 466–475, 2001.

[80] D. L. Snyder. **Parameter Estimation for Dynamic Studies in Emission-Tomography Systems Having List-Mode Data.** *IEEE Transactions on Nuclear Science, vol. 31, no. 2,* pages 925 –931, April 1984.

[81] G. Szekely, H. Hahn, V. Potesil, T. Kadir, G. Platsch & M. Brady. **Personalization of Pictorial Structures for Anatomical Landmark Localization,** *Information Processing in Medical Imaging, Lecture Notes in Computer Science, volume 6801,* pages 333–345. Springer Berlin / Heidelberg, 2011.

[82] L. Tellmann, R.R. Fulton, K. Bente, I. Stangier, O. Winz, U. Just, H. Herzog & U.K. Pietrzyk. **Motion Correction of Head Movements in PET: Realisation for Routine Usage.** *In IEEE Nuclear Science Symposium Conference Record, volume 5,* pages 3105–3107, 2003.

[83] M. M. Ter-Pogossian, M. E. Phelps, E. J. Hoffman & Nizar A. Mullani. **A Positron-Emission Transaxial Tomograph for Nuclear Imaging (PETT).** *Radiology, vol. 114, no. 1,* pages 89–98, 1975.

[84] K. Thielemans, S. Mustafovic & L. Schnorr. **Image Reconstruction of Motion Corrected Sinograms.** *In IEEE Nuclear Science Symposium Conference Record, volume 4,* pages 2401–2406, 2003.

[85] H. Toyama, M. Ichise, J-S. Liow, D. C. Vines, N. M. Seneca, K. J. Modell, J. Seidel, M. V. Green & R. B. Innis. **Evaluation of AnestheSia Effects on [18F]FDG Uptake in Mouse Brain and Heart Using Small Animal PET.** *Nuclear Medicine and Biology, vol. 31, no. 2,* pages 251 – 256, 2004.

[86] University of Washington, Division of Nuclear Medicine. **Web Page: SimSET,** http://depts.washington.edu/simset/html/simset_main.html *Last checked 11 July 2011*

[87] University of Washington, Division of Nuclear Medicine. **Web Page: Introduction to PET Physics.** http://depts.washington.edu/nucmed/IRL/pet_intro/toc.html *Last checked 23 June 2011*

[88] *P. Vaska, C.L. Woody, D.J. Schlyer, S. Shokouhi, S.P. Stoll, J.-F. Pratte, P. O'Connor, S.S. Junnarkar, S. Rescia, B. Yu, M. Purschke, A. Kandasamy, A. Villanueva, A. Kriplani, V. Radeka, N. Volkow, R. Lecomte & R. Fontaine.* **RatCAP: Miniaturized Head-mounted PET for Conscious Rodent Brain Imaging.** *IEEE Transactions on Nuclear Science, vol. 51, no. 5,* pages 2718 – 2722, Oct. 2004.

[89] J. S. Warm, W. N. Dember & P. A. Hancock. **Automation and Human Performance: Theory and Applications; Chapter 9 – Vigilance and Workload in Automated Systems.** Mahwah, NJ, Lawrence Erlbaum Associates, 1996.

[90] J. S. Warm, R. Parasuraman & G. Matthews. **Vigilance Requires Hard Mental Work and Is Stressful.** *Human Factors: The Journal of the Human Factors and Ergonomics Society, vol. 50, no. 3*, pages 433–441, 2008.

[91] C.C. Watson, D. Newport, M.E. Casey, R.A. deKemp, R.S. Beanlands & M. Schmand. **Evaluation of Simulation-based Scatter Correction for 3-D PET Cardiac Cmaging.** *IEEE Transactions on Nuclear Science, vol. 44, no. 1,* pages 90–97, 1997.

[92] C.C. Watson. **New, Faster, Image-based Scatter Correction for 3D PET.** Nuclear Science, IEEE Transactions on, vol. 47, no. 4, pages 1587–1594, 2000.

[93] B. Zitova & J. Flusser. **Image Registration Methods: a Survey.** *Image and Vision Computing, vol. 21,* pages 977–1000, 2003.

This concludes the thesis.

Typeset in Adobe Garamond Pro, Trajen Pro, and Gill Sans.
Peter Nordberg, Oxford, July 2011 (revised June 2012)