UCSF
UC San Francisco Previously Published Works

Title

A 3D Reconstruction of RLM Data Using MAP with an Optimized Regularization Parameter for a Total Variation Prior

Permalink

https://escholarship.org/uc/item/7jj5k39d

Authors

Gullberg, Grant
Lyu, Qihui
Seo, Youngho
et al.

Publication Date

2026-11-05

Copyright Information

This work is made available under the terms of a Creative Commons Attribution License, available at https://creativecommons.org/licenses/by/4.0/

eScholarship.org Powered by the California Digital Library
University of California

A 3D Reconstruction of RLM Data Using MAP with an Optimized Regularization Parameter for a Total Variation Prior

Grant T. Gullberg,[1,2] Qihui Lyu,[3] Youngho Seo,[2]
Yongjin Sung[3]

[1]Lawrence Berkeley National Laboratory,
Molecular Biophysics and Integrated Bioimaging Division,
Berkeley, CA, USA

[2]UCSF Physics Research Laboratory,
Department of Radiology and Biomedical Imaging,
University of California San Francisco, CA, USA

[3]Department of Radiation Oncology,
University of California, San Francisco, CA, USA

[4]Department of Biomedical and Mechanical Engineering,
University of Wisconsin-Milwaukee,
Milwaukee, WI, USA

Technical Report LBNL-2001778
Lawrence Berkeley National Laboratory
November 5, 2026

Abstract

A new microscopic nuclear imaging instrument using radioluminescence microscopy (RLM) has been developed to image living cells at high resolution with radiotracers tagged with positron emitters. These are used in biological samples to mimic clinical applications to evaluate their effectiveness to target receptors, antigens, and proteins that are overexpressed during growth of cancerous cells. The methodology has demonstrated impressive results for planar applications using optical reconstruction of β-ionization tracks (ORBIT); however, 3D imaging presents a challenging problem of reconstructing the 3D distribution of the source requiring backtracking of the ionization track of the positron in the scintillator in order to determine its position and 3D orientation at the detector surface of the instrument. Four microlens arrays are used to record projections, limiting the angle of projection measurements to 360° sampling in the azimuthal angle ϕ and 40° sampling in the polar angle θ. The angle information acts as a collimator, but the limited angle acquisition presents a challenging reconstruction problem which increases in difficulty for sources further from the detector. We consider reconstruction as a hierarchical MAP estimation problem for Poisson emission data. For fixed regularization parameter β, the image is obtained by minimizing the sum of the negative Poisson log-likelihood and a total variation penalty, whereas to estimate β jointly with the image, we introduce a hyperprior function of β, yielding a joint objective function in terms of the reconstruction variables and the regulation parameter β. This leads naturally to an alternating optimization scheme in which x is updated by MAP-TV reconstruction and β is updated from the current image total variation. Our focus is on developing methodology that provides a balance between data fidelity and a prior model. Good results are demonstrated using Geant4 simulations of point sources as far as 98 μ from the face of the detector. The novelty is the development of a microscopy application to image individual cells three-dimensionally with radiotracers intended for human imaging and therapeutic applications.

Key words: limited angle reconstruction, Radioluminescence Microscopy, MAP-TV, total variation

Preface

This LBNL report is an extension of an abstract accepted as a poster for the IEEE NSS MIC Meeting, Nov. 7-14, 2026 in Granada, Spain. This report gives results related to data from simulations of a new microscopic nuclear imaging instrument being developed at the University of Wisconsin-Milwaukee by Professor Yongjin Sung. The instrument is designed to perform radioluminescence microscopy (RLM) of living cells to obtain high-resolution three-dimensional images of the distribution of radiotracers tagged with positron emitters. These radiotracers are used in biological samples to evaluate their effectiveness to target receptors, antigens, and proteins that are overexpressed during growth of cancerous cells. This report focuses on the development of MAP-TV algorithms with joint estimation of a reconstructed image from Poisson data and a total variation prior with a regularization parameter β. A hyperprior function of β yields a joint objective function in terms of the reconstruction variables and the regulation parameter β. The algorithms are implemented using Mathematica, yielding a joint objective function in terms of the reconstruction variables and the regulation parameter β. A key result presented in this report demonstrates that minimizing the generalized Kullback-Leibler (KL) divergence is mathematically equivalent to finding the maximum likelihood estimate (MLE) under Poisson noise.

1. Introduction

The goal of this project is to develop a new microscopic nuclear imaging instrument using radioluminescence microscopy (RLM) [1-3] that provides the distributions of radionuclides in single cells of living specimens using the same clinical radiotracers used in clinical imaging and therapy. To provide subcellular resolution (< 1 μm) for confluent living cells 3D backtracking is used to trace the scintillation tracks in a detector backward to the positron emitting radionuclide by recording their 3D orientations at the detector surface. To address the challenge of the tracks decaying within tens of microseconds, snapshot projection optical tomography (SPOT) [4] is used to capture the projection images corresponding to various viewing angles in a single snapshot, without parallex error by using a telecentric arrangement of the objective lens and a micro-lens array (MLA). Previously, SPOT imaging techniques have been demonstrated with fluorescent [4] and absorbing [5] specimens, whereas here we will use SPOT to instantaneously capture the 3D orientations of scintillation tracks for the first time.

In the following we develop a maximum a posteriori (MAP) with total variation (TV) reconstruction algorithm to process RLM data. The data processing for 3D RLM starts with calculating the 3D orientations of scintillation tracks from four projection images of the MLA shown in Fig. 1. Backtracking the scintillation tracks provides the measurement of the position and the angle the positron enters the scintillation detector. Each event is accumulated into a sinogram which forms the projections of the limited angle reconstruction. The challenge is to develop appropriate priors that suppress noise and limited angle artifacts whereas at the same time provide excellent resolution. The final reconstruction is obtained by an iterative method of image values and an optimized regularization parameter. In the following we develop the theory for our MAP-TV algorithm and evaluate our method in reconstructing 24931 events simulated using Geant4.

In the following we present the theory involved in the MAP-TV formulation and the development of algorithms using hierarchical and discrepancy principles for reconstructing tomographic data. Results are then presented for the reconstruction of simulated data. Then we discuss aspects of our limited angle 3D reconstruction of RLM data and the future potential of developing duality algorithms for the reconstruction of this type of limited angle reconstruction problem.

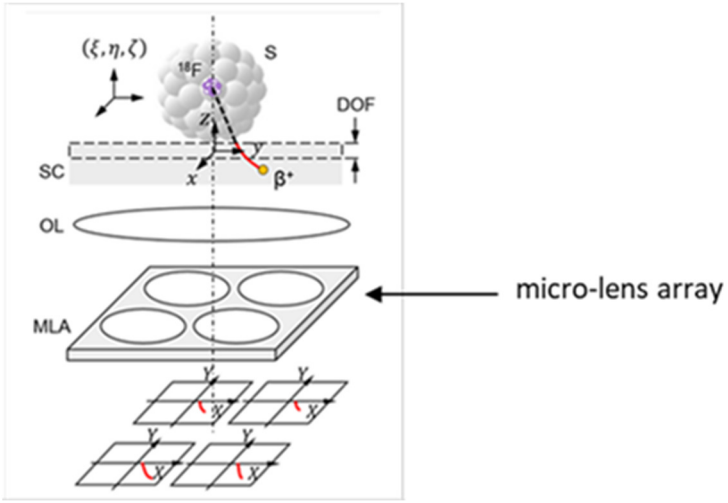


Figure 1. Simulation geometry. S: Sample; SC: scintillation crystal; OL: objective lens; MLA: micro-lens array; DOF: depth of field.

2. A MAP reconstruction algorithm for joint estimation of image and regularization parameter with a total variation prior

In the following we consider various approaches to the joint estimation of the image and regularization parameter β with a total variation prior $TV(x)$. Finding the optimal values for regularization parameters can involve either exhaustive experimentation or using the so-called L-curve method [12]. The L-curve method basically uses scatter plots of the final values of the objective function against the values of the regularization function. The plot is expected to form an L-shaped curve, and the optimal value of the regularization parameter is chosen by selecting the one that corresponds to the one at the corners [13]. The method requires several reconstructions and, it may not always form a neat corner, which complicates the process of selecting the optimum value [14]. Previously [11] we presented a dynamic regularization (DR) algorithm like a primal dual algorithm [7], where the targeted optimized variable and the regularization parameter β are both updated within each iteration. The parameter β is selected in a way to preserve a balance between the fidelity term and the prior. This underlying mathematical approach was originally developed by Ito et al. [17]. Several other methods to preserve a balance between the fidelity term and the prior have been proposed in literature including a discrepancy principle [15][16] and a balancing principle [17][20]. We will discuss in more detail the application of the discrepancy principle which is the application of choice in our implementation of the MAP-TV algorithm.

2.1 MAP–TV objective function

The MAP estimate with total variation prior is:

$$\hat{x} = \arg \underbrace{max}_{x \geq 0} \left[\underbrace{\log p\big((y|x)\big)}_{Poisson\ data\ term} - \beta \underbrace{TV(x)}_{total\ variation} \right] ,$$

where $\hat{x}$ is the optimum for the reconstructed image value for the matrix x, y is the projection matrix, $\log p(y|x)$ is the likelihood function – the probability of the data y given the image x. The log likelihood for a Poisson distribution is

$$\log p(y|x) = \sum_i y_i log[(Fx)_i] - (Fx)_i \ ,$$

summed over all measured projection values y_i. Fx are modeled projections with system matrix F, and the 3D Total Variation prior [10] is

$$\mathrm{TV(x)} = \sum_j \sqrt{\left(\nabla_x x_j\right)^2 + \left(\nabla_y x_j\right)^2 + \left(\nabla_z x_j\right)^2} \ .$$

2.2 Gradient of the MAP Objective Function

To determine the condition for an optimum solution we obtain the gradient of the MAP objective function:

$$\nabla\Phi(x) = \underbrace{F^T\left(1 - \frac{y}{Fx}\right)}_{MLEM\ term} + \beta\, \nabla TV(x) \ ,$$

where

$$\Phi(x) = -\log\left(p(y|x) + \beta\, TV(x)\right) \ ,$$

$$\nabla_x \mathrm{TV}(x) = -\nabla \cdot \left(\frac{\nabla x}{|\nabla x|}\right) \ ,$$

and *MLEM* term is the maximum likelihood expectation maximization term.

2.3 A split update for a standard MAP-TV Iteration

For the iteration, one could consider a **split update**:

(a) EM step (likelihood):

$$x^{\left(k+\frac{1}{2}\right)} = x^{(k)} \cdot \frac{F^T\left(\frac{y}{Fx^{(k)}}\right)}{F^T\mathbf{1}} \ ,$$

and

(b) TV gradient descent step:

$$x^{(k+1)} = x^{\left(k+\frac{1}{2}\right)} - \tau\, \beta\, \nabla TV(x) \ ,$$

where τ is a fixed step size and β is the regularization strength.

2.4 A maximum likelihood expectation maximization - total variation (MLEM-TV) update [10]

The MAP–TV reconstruction is trying to minimize the following objective function

$$\Phi(x) = \sum_i \underbrace{[(Fx)_i - y_i \log(Fx)_i]}_{Poisson\ data\ term} + \beta_{TV} \underbrace{TV(x)}_{total\ variation} \ ,$$

with Poisson negative log-likelihood

$$L(x) = \sum_i \left[(Fx)_i - y_i log\left((Fx)_i + \epsilon_{proj}\right)\right] \ ,$$

and isotropic TV penalty

$$TV(x) = \sum_j \left\|(\nabla x)_j\right\|_2 \ .$$

The MLEM-TV update for the reconstruction parameters are

$$x^{k+1} = x^k \frac{F^T\left(\frac{y}{Fx^k}\right)}{F^T 1 + \beta \nabla \cdot \left(\frac{\nabla x^k}{\left\|\|\nabla x^k\|\right\|}\right)} \ .$$

Note: β is not updated.

2.5 Simultaneous Optimization of x and β

Instead of fixing β in our algorithm, we treat it as a variable and obtain the following joint *optimization*:

$$\underbrace{min}_{x,\beta}\ \Phi(x,\beta)\ \ ,$$

where

$$\Phi(x,\beta) = -\log p(y|x) + \beta\, TV(x) + \Psi(\beta)\ \ ,$$

and $\Psi(\beta)$ is a prior or constraint on β. Stationary conditions are

(a) with respect to x:

$$\nabla_x \Phi(x,\beta) = F^T\left(1 - \frac{y}{Fx}\right) + \beta \nabla TV(x) = 0\ \ .$$

(b) with respect to β:

$$\nabla_\beta \Phi(x,\beta) = TV(x) + \Psi'(\beta) = 0\ \ .$$

Various choices can be made for the prior $\Psi'(\beta)$. For a Bayesian update a common rule

$$\beta^{k+1} = \frac{N}{TV(x^k)}\ \ ,$$

or more generally:

$$\beta^{k+1} \propto \frac{1}{TV(x^k)}\ \ .$$

The joint estimation involves

$$\underbrace{min}_{x}\underbrace{min}_{\beta}\Phi(x,\beta) = [-log\, p\ (y|x) + \beta\, TV(x)]\ \ ,$$

$$\nabla_x \Phi(x,\beta) = F^T\left(1 - \frac{y}{Fx}\right) + \beta\, \nabla \mathrm{TV}(x)\ \ ,$$

$$\nabla_x \mathrm{TV}(x,\beta) = -\nabla \cdot \left(\frac{\nabla x}{|\ \nabla x\ |}\right)\ \ ,$$

$$\nabla_\beta TV(x,\beta) = \nabla TV(x)\ \ .$$

2.6 Practical Joint Estimation Scheme

Iteration k:

1. EM update:

$$\boldsymbol{x^{\left(k+\frac{1}{2}\right)} = x^{(k)} \cdot \frac{F^T\left(\frac{y}{Fx^{(k)}}\right)}{F^T 1}}\ \ ,$$

2. TV update:

$$x^{(k+1)} = x^{\left(k+\frac{1}{2}\right)} - \tau\,\beta^{(k)}\,\nabla\,\mathrm{TV}(x) \quad ,$$

3. β update:

$$\beta^{(k+1)} = \frac{c}{TV(x^{(k+1)}) + \epsilon} \quad .$$

The adaptive β starts large provide smoothness then decreases as structure emerges. For **β large** → strong smoothing → suppress noise, lose resolution; whereas **β small** → sharper image → more noise

Recommended strategy : $c \approx 0.1\ to\ 1 \times TV(x_{MLEM})\ and\ \epsilon \approx 10^{-6}(stability)$, where c is a user-defined scaling constant that controls the *overall strength of regularization* and ϵ is a very small number to prevent division by zero in the algorithm update.

2.7 Poisson Deviance

Before looking at other possibilities for optimizing the joint objective function for x and β, let's first consider the Poisson distribution. The probability of getting k events in an interval with a mean λ is calculated using the following Poisson probability distribution:

$$P(X = k) = \frac{\lambda^k e^{-\lambda}}{k!}$$

In our agorithm we use the Posson deviance to update the solution for the regulation parameter β. Comment from Chat GPT: Strictly speaking, I **do not know of a paper that uses exactly the adaptive update**

$$\beta^{k+1} = \beta^k \left(\frac{D_{target}}{D(x^k)}\right)^{\gamma}$$

for TV-regularized SPECT/PET reconstruction. In our algorithm we use

$$D\left(x^{k+1}\right) = 2\sum_i \left[y_i log\left[\frac{y_i}{(Fx^{k+1})_i + \epsilon_{proj}}\right] - y_i + \left(Fx^{k+1}\right)_i\right]$$

where the deviance of y given the modeled projections Fx is

$$D(y|Fx) = 2\sum_i \left(y_i log\left[\frac{y_i}{(Fx)_i}\right] - y_i + (Fx)_i\right) \quad .$$

2.8 EM+TVGrad3D+Adaptive Beta Update

The current EM+TVGrad3D+Adaptive Beta implementation involves the following algorithm

For iteration *k*:

Step 1 – EM Data Update:

$$x_{EM}^{k} = \prod\nolimits_{x\geq 0} x^{k} \cdot \frac{F^{T}\left(\frac{y}{(Fx^{k}+\epsilon_{proj})}\right)}{F^{T}1+\epsilon_{proj}} \quad .$$

Step 2 – Explicit TV gradient step:

$$g_{TV}^{k} = \nabla TV_{\epsilon}(x_{EM}^{k}) = -\nabla \cdot \left(\frac{\nabla x}{\sqrt{|\nabla x|^{2}+\epsilon_{TV}^{2}}}\right) \quad ,$$

$$x^{k+1} = \prod\nolimits_{x\geq 0}\left[x_{EM}^{k} - \frac{\tau_{TV}\beta^{k}}{max|g_{TV}|+10^{-12}}\, g_{TV}^{k}\right] \quad .$$

Step 3 – Following damped update uses an adaptive beta update based on Poisson deviance:

$$\beta^{k+1} = 0.8\beta^{k} + 0.2\ clip\left[\beta^{k}\left(\frac{D_{target}}{dev^{k+1}}\right)^{\gamma^{\beta}}, \beta_{min}, \beta_{max}\right] \quad .$$

The update is a **heuristic controller** motivated by discrepancy-principal ideas [21-24].

2.9 Adaptive Beta Update

The adaptive β update is supported in literature by the **discrepancy principles** [21, 22] and **adaptive regularization parameter selection**. Bertero et al. is probably the **most important reference** for our work [21]. Their main idea is: **1.** measure the Poisson data misfit, **2.** compare it with an expected target, and **3.** choose the regularization parameter so the reconstruction satisfies the desired discrepancy. The adaptive β is essentially an iterative realization of this philosophy. Bevilacquar et al. [22] is particularly relevant because our reconstruction is based on Poisson statistics. It discusses: **1.** KL/Poisson likelihood, **2.** TV regularization, **3.** automatic parameter selection, and **4.** discrepancy principles. Sixou et al. [23] is interesting because it explicitly studies adaptive regularization parameter estimation for Poisson inverse problems. Although it uses bilevel optimization rather than our present update, it supports the idea that the regularization parameter should be adapted automatically. Calatroni et al. [24] develops adaptive TV parameter selection and provides theoretical motivation for varying the TV weight rather than keeping it fixed. The TV regularization parameter was updated adaptively using a discrepancy-principle-inspired feedback rule. The update increases β when the Poisson deviance exceeds the desired target and decreases β when the data fidelity is overfit. This adaptive strategy is motivated by discrepancy-principle methods for Poisson inverse problems but is implemented here as a damped multiplicative controller tailored to our proposed LM-MAP-EM-TV algorithm. Looking at our reconstruction work, there is probably a stronger theoretical justification for a new update than the current update. Instead of

$$\beta^{k+1} = \beta^{k}\left(\frac{D_{target}}{D(x^{k})}\right)^{\gamma} \quad ,$$

one should consider

$$\beta^{k+1} = \prod_{[\beta_{min},\beta_{max}]} \left[\beta^k + \eta \left(\frac{D_k - D_{target}}{D_{target}} \right) \right] ,$$

where Π denotes projection onto the interval $[\beta_{min}, \beta_{max}]$. This **1.** behaves like a **projected gradient ascent/descent controller** on the hyperparameter; **2.** is easier to analyze for stability; **3.** is closer in spirit to bilevel optimization and primal-dual control methods; and **4**. avoids the tendency of the multiplicative rule to drive β rapidly toward βmin or βmax. This additive projected update would make a stronger methodological contribution while remaining consistent with the discrepancy-principle literature. In mathematics and computational science, Morozov's discrepancy principle [25] is an a posteriori parameter choice rule used for regularization. When solving ill-posed inverse problems (like image deblurring or signal processing), this principle stops the regularization process. It ensures that you do not over-smooth the data or fit the underlying noise, providing a stable, reliable numerical solution.

2.10 Other possible priors

The total variation prior is one of possible priors. Other priors are: (a) No prior on beta; TV(x)=0; trivial (over-smoothed solution) (b) Quadratic prior -- $\Psi(\beta) = \frac{\alpha}{2}\,\beta^2$ then: $\beta = -\frac{TV(x)}{\alpha}$. This is not ideal physically, but analytically simple; (c) Bayesian / evidence-based update (more useful). A common practical rule: $\beta^{(k+1)} = \frac{N}{TV(x^{(k)})}$ or more generally: $\beta^{(k+1)} \propto \frac{1}{TV(x^{(k)})}$; (d) Discrepancy principle (very important for practice). Choose β such that: $\sum_i \frac{(y_i - (Fx)_i)^2}{(Fx)_i} \approx N_{data}$. This balances data fidelity and smoothness. A practical joint iteration scheme is what is presented in our algorithm.

2.11 The Discrepancy Principle for Poisson Data

The **Poisson discrepancy** measures data fidelity or deviation between an observed dataset y and a fitted theoretical model or reconstruction under Poisson noise. It is mathematically defined as the **generalized Kullback-Leibler (KL) divergence** [26, 27, 28, 29]. In practical applications like inverse problems (e.g., image denoising) or regression, this discrepancy is often paired with a regularization parameter according to the **discrepancy principle**. The goal is to find a model where the discrepancy value is approximately equal to the expected noise level of the data [26, 27, 28, 29, 30, 31].

The Poisson KL divergence is a measure of how well the reconstructed image predicts the measured data [26]:

$$D_{KL}\,(y \| \mu) = \sum_{i=1}^{N} \left(y_i log \frac{y_i}{\mu_i} - (y_i - \mu_i) \right)$$

where y_i is the actual observed count and $\mu_i = (Fx)_i$ is the predicted mean or counts. Minimizing the likelihood is equivalent to minimizing the Poisson KL divergence. The Poisson deviance in our algorithm is simply $D_P = 2D_{KL}$.

The Kullback–Leibler (KL) divergence (also called relative entropy and I-divergence) [26], is a type of statistical distance, which is a measure of how much an approximating probability distribution *Q* is different from a true probability distribution P [27,28]. Mathematically, it is defined as

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x)\, \log \frac{P(x)}{Q(x)}.$$

The citations [26,27,28, 29] and [26,27,28, 29, 30, 31] point to classic foundational papers that established the generalized Kullback-Leibler (KL) divergence and Morozov's discrepancy principle for Poisson-distributed noise in inverse problems [26,27,28, 29, 30]. This literature include foundational works involving the definition of generalized KL divergence under Poisson noise [26,27,28, 29]. These citations point to the original formulation of the Kullback-Leibler divergence, its expansion to unnormalized measures (generalized KL), and its first applications to Poisson maximum-likelihood problems like the Richardson-Lucy [30] or EM algorithms. Kullback and Leibler original paper [26] defined the standard Kullback-Leibler divergence. Csiszár formally defined the "I-divergence" (generalized KL divergence) for unnormalized measures, which allows the total intensity of the reconstruction to vary from the observed data [27]. Shepp and Vardi [28] proved that maximizing the Poisson log-likelihood is mathematically equivalent to minimizing the generalized KL divergence between the data and the forward model. The Richardson-Lucy algorithm is the classic iterative method used to minimize this exact Poisson discrepancy function in optical imaging and astronomy [29].

Morozov [30] invented the **discrepancy principle**, which is a stopping criteria/regularization parameter choice. Morozov's discrepancy principle is not a cost function. It is a rule for choosing the regularization parameter. Instead of minimizing $L(x) + \beta\, TV(x)$, Morozov: asked how well should the reconstruction fit the data? Only to the level expected from measurement noise. Instead of driving the likelihood as low as possible, Morozov choose β such that $D(x) \approx D_{target}$, where $D(x)$ is discrepancy (often the Poisson deviance), D_{target} is the discrepancy expected from the noise.

Morozov [30] original work was built for Gaussian noise (Least Squares), stating that you should stop regularizing when your residual matches the noise variance. Traditional Morozov data-fidelity doesn't work for Poisson noise because the variance depends on the signal intensity itself. These papers explicitly redefined how to calculate the "expected noise level" under a generalized KL divergence framework so researchers know exactly when to stop an image denoising or deblurring algorithm [21, 31, 32]. Csiszár [27] provided the foundational information-theoretic definitions for the discrepancy principle while Bertero et al. [21] and Hohage & Werner [32] are sources for applying the discrepancy principle to Poisson inverse problems.

2.12 Common application of Poisson deviance and maximum a posteriori (MAP)

1. Shared cost functions. Maximum a posteriori (MAP) estimation with a negative log-likelihood derived from a Poisson distribution yields the generalized Kullback–Leibler (KL) divergence as the data-fidelity term, which looks identical to the variational objective function [26,27].

2. Conceptual divergence. Bayesian inference treats the regularization parameter (or hyperparameter) through hierarchical prior distributions or marginalization, whereas the discrepancy principle sets it deterministically by evaluating the size of the residual against the noise level [26,27].

3. Frequentist rule. The principal functions as a posteriori stopping or tuning rule based on the statistical concentration of the Poisson discrepancy, not as a direct consequence of Bayes' theorem [26,27,28].

4. The Poisson Log-Likelihood. Let $y \in \mathbb{R}_{\geq 0}^{N}$ be the observed vector of photon or particle counts. Let Fx be the expected counts, where $x \in \mathbb{R}_{\geq 0}^{M}$ is the unknown true signal and F is a linear forward operator (like a blurring matrix). For each pixel or bin i, the data follows a Poisson distribution:

$$P(y_i|(Fx)_i) = \frac{(Fx)_i^{y_i} e^{-(Fx)_i}}{y_i!} \quad .$$

Assuming independent noise across bins, the total likelihood is the product over all i. Taking the negative natural logarithm gives the negative log-likelihood:

$$-logP(y|Fx) = \sum_{i=1}^{N} [(Fx)_i - y_i log((Fx)_i) + log(y_i!)]$$

5. Dropping constant terms. In optimization and inverse problems, we minimize this function with respect to x. The factorial term $log(y_i!)$ depends only on the observed data y not on the parameter x. For optimization purposes, it acts as a constant and can be ignored. Adding a matching constant term $+y_i log(y_i) - y_i$ (which evaluates to 0 if $y_i = 0$, using the convention $0log0 = 0$) allows us to reorganize the equation without changing the minimizer:

$$D(y|FX) = \sum_{i=1}^{N} \left[y_i log\left(\frac{y_i}{(Fx)_i}\right) + (Fx)_i - y_i \right] \quad .$$

6. The generalized KL divergence. The resulting functional $D(y|FX)$ is exactly the **generalized Kullback-Leibler divergence** (also known as the I-divergence) between the data y and the model estimate Fx [26]. Unlike the standard KL divergence used in probability theory, the generalized version does not require y or Fx to sum to 1 [26]. **Core Takeaways: 1. Equivalence:** Minimizing the generalized KL divergence is mathematically identical to finding the Maximum Likelihood Estimate (MLE) under Poisson noise [26]. 2. **Information Theory Link:** The divergence measures the information loss when the model Fx is used to approximate the true empirical distribution of the data y. One can show this relates to **Csiszár's algebraic framework** for inverse problems and show this function as the **data-fidelity term** in algorithms like Richardson-Lucy deconvolution.

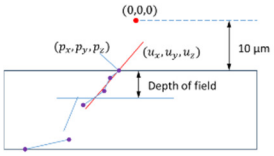
3. Materials and Methods

3.1 Data Input

A Geant4 simulation was performed to provide RLM projection data to evaluate our joint estimation algorithm. Geant4 simulated data provided the 3D orientation on the detector surface (Fig. 2), including the 3D position and the angle of the detected positron. Figure 2 illustrates the results of a Geant4 simulation of positron trails for a F-18 point source located at a distance 10 microns from the detector surface. The results of the Geant4 simulation of 24,931 events were included in an Excel file. Each row represents one scintillation trail, and the columns are values for the coordinates (px, py, px) of the starting point of a scintillation trail, and (ux,uy,uz) are the directions of the point trail at the surface. The task was to reconstruct the 3D distribution of point sources at 2, 59, and 98 µms from the detector.

Genant4 Simulation of ^{18}F Point Source

24,931 Events



px	py	pz	ux	uy	uz
-6.6869	-4.55169	10	-0.49148215	-0.283579273	0.823424613
-7.16131	-4.8801	10	-0.442679566	-0.167026624	0.880986327
-4.33504	8.89787	10	-0.22176683	0.602658076	0.766565533
1.03895	-8.02056	10	0.093476932	-0.519691683	0.849224716
-12.8715	-6.914	10	-0.475733798	-0.376854621	0.794769116
-2.0348	-5.62691	10	-0.247797176	-0.469645882	0.847366098
1.4596	-0.0622373	10	-0.044336088	0.259171266	0.964813229
-2.83432	2.99764	10	0.148953763	0.334714288	0.930472526
12.5718	5.32271	10	0.422621511	0.24598376	0.872286105
6.46098	-3.04908	10	0.157700258	-0.485930422	0.859652403
-13.2006	-183.704	10	-0.272233475	-0.228548627	0.934694849
-121.05	142.106	10	-0.300932326	0.31164161	0.901287547
-198.544	-581.019	10	0.059525976	-0.580661132	0.811966322
-0.525323	4.5085	10	0.012283154	0.429559081	0.902955215
-7.98275	-3.77677	10	-0.554888661	-0.28872934	0.780214037
0.275827	-4.28141	10	0.169855386	-0.394757425	0.90294835
4.88445	0.340886	10	0.507783984	0.179961334	0.842478097
5.55379	4.50009	10	0.494706929	0.376457808	0.783290861
2.11716	-0.81613	10	0.192227747	-0.314837254	0.929476195
8.13453	3.16254	10	0.501189081	0.251783452	0.827897698
-4.34978	0.765566	10	-0.518493545	-0.05879772	0.853057602
98.3586	-193.493	10	-0.571068214	0.237945749	0.785660815
-111.892	130.115	10	-0.40269678	0.277888575	0.872131437
9.46601	-2.06311	10	0.370989185	-0.176320645	0.911744512
-5.18696	-6.17182	10	-0.444477294	-0.203947705	0.872264449
-2.78902	-9.33404	10	-0.006641684	-0.578047327	0.81597621
0.105605	3.45717	10	0.037694929	0.306634311	0.951080697
10.1213	-3.28258	10	0.535596733	-0.108055141	0.837532224
0.628075	7.91237	10	0.039833974	0.636981293	0.769849392
-3.31912	-0.774675	10	-0.213543251	0.095524664	0.972252189
11.4083	8.98358	10	0.517331901	0.138940763	0.844430677

Figure 2. Geant4 simulated positron trails for a ^{18}F point source located 10 µm from the crystal surface. The result of 24,931 events was compiled in an Excel file to be processed. Each row represents one scintillation trail, and the columns (px,py,pz) are the coordinates of the starting point of a scintillation trail (unit: µm), and (ux,uy,uz) is the direction of the positron trail at the surface. The coordinate (0,0,0) is the center of the imaging coordinate system. In our work presented here the origin was located 50 µm from the detector surface.

3.2 3D Coordinate System

The geometry [6] for the simulation is assumed to be a 3D coordinate system shown in Fig. 3 with the events measured by a detector that is translated by the coordinates (px,py,pz). The slopes (ux,uy,uz) of the measured events define the angles θ and ϕ of the assumed projection measurements:

$$sin\phi = u_y \Big/ \sqrt{u_x^2 + u_y^2}\,, \qquad cos\phi = u_x \Big/ \sqrt{u_x^2 + u_y^2}\,,$$

$$cos\theta = \frac{u_z}{\sqrt{u_x^2 + u_y^2 + u_z^2}}\,, \qquad sin\theta = \frac{\sqrt{u_x^2 + u_y^2}}{\sqrt{u_x^2 + u_y^2 + u_z^2}}\,.$$

The vectors for the rotating coordinate system are given by the following equations:

$$\underline{\alpha} = (-sin\phi, cos\phi, 0)\ ,$$
$$\underline{\beta} = (-cos\theta cos\phi, -cos\theta sin\phi, sin\theta)\ ,$$
$$\underline{\theta} = (sin\theta cos\phi, sin\theta sin\phi, cos\theta)\ ,$$
$$\underline{p} = (px, py, pz)\ \ .$$

Note that the vector $\underline{\alpha}$ remains in the xy-plane. Also, the plane of the uv-coordinate system ($u = \underline{x} \cdot \underline{\alpha}$, $v = \underline{y} \cdot \underline{\beta}$) is parallel to the $\theta\beta$- coordinate system—the coordinate system definded by the vectors $\underline{\theta}$ and $\underline{\beta}$.

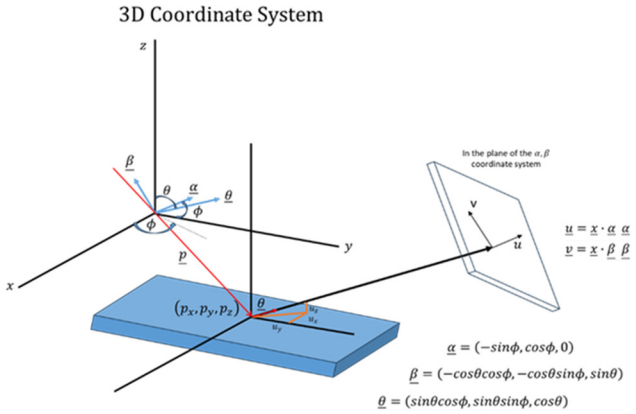


Figure 3. Reconstruction geometry. For the simulated data, the angle θ ranges from 0 to $\pi/4$ and the angle ϕ ranges from 0 to 2π. Note that the 2D projection plane is parallel to the plane of the $\alpha\beta$-coordinate system.

3.3 Sinogram

Each projection event was binned into a sinogram (Fig. 4). For the angle θ, the bin index was determined by

$$I_\theta = Int\left[\frac{9.}{45.}\theta(I)\right] + 1\ \ ,$$

for *I*=1,…,9. For the angle ϕ the bin index is determined by

$$I_\phi = Int\left[\frac{18.}{360.}\phi(I)\right] + 1\ \ ,$$

for *I*=1,…,18. Suppose we let umax=Max[u(i), i=1,…,24931 events] and vmax=Max[v(i), i=1,…,24931 events], where i is the index number of the simulated events. Then, the bin index for u and v is determined by

$$I_u = Int\left[\frac{Nbins}{2\ Maxuv}u(i) + \frac{Nbins}{2} + 1\right]\ \ ,$$
$$I_v = Int\left[\frac{Nbins}{2\ Maxuv}v(i) + \frac{Nbins}{2} + 1\right]\ \ .$$

For the Geant4 simulated events, umax=1715.05 and vmax=1713.57, so Maxuv=max[umax, vmax]+1 = 1716.05. Suppose we bin the projection samples into Nbins=256 with max I_u and max I_v = 129; then, the bin size = 1716.05/129 = 13.3μm; whereas, to have a bin size of 1μm we would need Nbins = 1716. A reasonable option is to truncate the data to a more reasonable size of 256. Using this sinogram, one can apply various analytical and reconstruction algorithms.

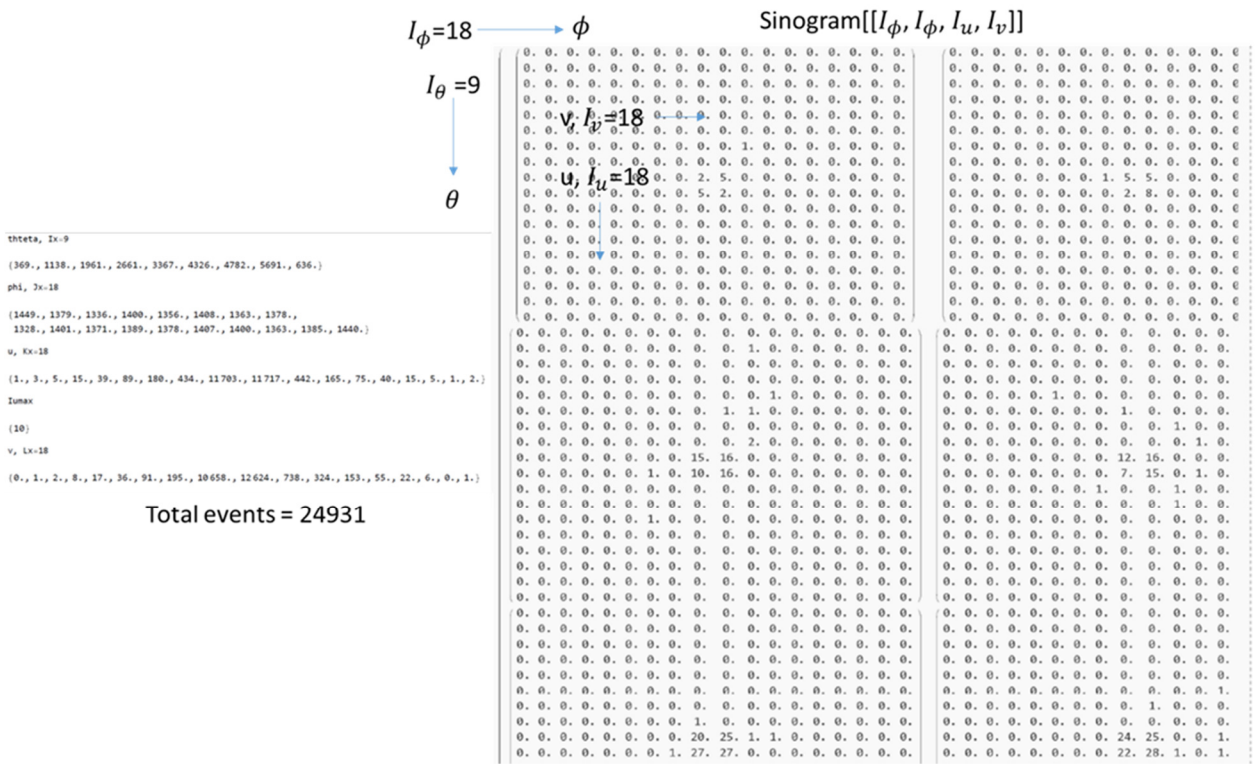


Figure 4. Sinogram: Geant4 simulated positron trails for a ^{18}F point source located 10 μm from the crystal surface. For the angle θ, the bin index was determined by $I_\theta = Int\left[\frac{9.}{45.}\theta(I)\right] + 1$, for I=1,…,9. For the angle ϕ the bin index is determined by $I_\phi = Int\left[\frac{18.}{360.}\phi(I)\right] + 1$, for I=1,…,18. Suppose we let umax=Max[u(i), i=1,…,24931 events] and vmax=Max[v(i), i=1,…,24931 events], where i is the index number of the simulated events. Then, the bin index for u and v is determined by $I_u = Int\left[\frac{Nbins}{2\,Maxuv}u(i) + \frac{Nbins}{2} + 1\right]$, $I_v = Int\left[\frac{Nbins}{2\,Maxuv}v(i) + \frac{Nbins}{2} + 1\right]$.

3.4 Algorithm

In our algorithm we treat β as a variable and jointly estimate x and β by alternating the minimization of: $\mathrm{J(x,\beta)} = \mathrm{L(x)} + \beta\,\mathrm{TV(x)} - \log \mathrm{p(\beta)}$, so $x^{(k+1)} = \arg\min_x J(x, \beta^k)$ and $\beta^{(k+1)} = \arg\min_\beta J(x^{k+1}, \beta)$; where $\mathrm{TV(x)} = \sum_j \sqrt{\left(\nabla_x x_j\right)^2 + \left(\nabla_y x_j\right)^2 + \left(\nabla_z x_j\right)^2}$ is the total variation prior, $\mathrm{L(x)} = \sum_i [(Fx)_i - y_i \log((Fx)_i)]$ is the negative Poisson log-likelihood, $y = Fx$ are projection data, x is the image matrix, and F is the system matrix with line length weighting [3,4]. A deviance-based adaptive beta update is used which aims to keep the Poisson deviance D near a target value. Poisson deviance measures how well predicted counts match measured counts under Poisson noise, or how much worse the model is than a model that fits every data point perfectly. TV is applied after the EM update as an additive correction.

For iteration *k*

1. **Step 1 – x update:**

$$x_{EM}^{k} = \prod\nolimits_{x\geq 0} x^{(k)} \cdot \frac{F^{T}\left(\frac{y}{\left(Fx^{(k)} + \epsilon_{proj}\right)}\right)}{F^{T}\mathbf{1} + \epsilon_{proj}}$$

$$\nabla TV_{\in}\left(x_{EM}^{k}\right) = -\nabla \cdot \left(\frac{\nabla x}{\sqrt{|\nabla x|^{2} + \epsilon_{TV}^{2}}}\right)$$

$$x^{(k+1)} = \prod\nolimits_{x\geq 0}\left[x_{EM}^{(k)} - \frac{\tau_{TV}\,\beta^{(k)}}{max\left|\nabla TV_{\epsilon}\left(x_{EM}^{(k)}\right)\right| + 10^{-12}}\nabla TV_{\epsilon}\left(x_{EM}^{(k)}\right)\right]$$

2. **Step 2 – β update:**

$$\beta^{(k+1)} = 0.8\beta^{(k)} + 0.2\; clip\left[\beta^{k}\left(\frac{D_{target}}{D\left(x^{k+1}\right)}\right)^{\gamma}, \beta_{min}, \beta_{max}\right]$$

$$D\left(x^{k+1}\right) = 2\sum_{i}\left[y_i log\left[\frac{y_i}{\left(Fx^{k+1}\right)_i}\right] - y_i + \left(Fx^{k+1}\right)_i\right]$$

where D is the Poisson deviance and ϵ is a very small number to prevent division by zero in the algorithm update. Notice each iteration invovles first updating x, which is used to udate β.

4. Results

Figure 5 shows the results of reconstructing 22000 Genat4 simulated events using Mathematica (Wolfram Research, Champaign, Illinois); comparing the MAP-TV reconstruction of a 51×51×51array at 2μ, 50μ, and 98μ away from the detector using the Siddon system matrix [9] (also refered to the line-length system matrix in our RECLBL Library [7] and later developed to model attenuation [8]). The sinogram was constructed so that the simulated events were accumulated into a matrix of 501×501 units corresponding to projection coordinates (u, v). The central 51×51 bins of the sinogram were used for the sampled projections. The reconstruction of the point source 2μ from the detector (Fig. 5a) show a shift of approximately 7.5 voxels which gives a resolution of approximately 48/7.5 = 6.4μ per voxel. The reconstruction was obtained with 200 iterations fixing τ at 0.10 while β converged to 1.04, starting initially with a value of 0.10. The reconstruction of the point source 50μ from the detector (Fig. 5b) shows less resolution of the reconstructed point source than the result at 2μ from the detector. The reconstruction was obtained with the same 200 iterations and $\tau = 0.10$; but in this case β converged to a value of 0.51. The reconstruction of the point source 98μ from the detector (Fig. 5c) shows good resoution in the transaxial slice but somewhat more dispersed in the 3D view. The total variation provides less dispersion but even still the result does not completely resolve the point source as can be seen by a maximum grayscale amplitude of **0.49** compared to **4.5** for 50 μm from the detector and **23.5** for 2 μm from the detector. In this case the reconstruction was obtained with the same 200 iterations and $\tau = 0.10$ while β converged to 0.41; in ths case the sinogram was compressed to 201 units in order to obtain a more pointed central peak.

MAP EM TV & β Iteration Steps

```
Starting MAP EM TV iterations...
betaTV = 0.1
tauTV = 0.1
betaMin = 0.0001
betaMax = 10.
gammaBeta = 0.25
adaptiveBetaQ = True
nIter = 200
betaTV initial = 0.1
tauTV = 0.1
nIter = 200
adaptiveBetaQ = True
targetDeviance = 421362
Starting MAP EM TV iterations...
MAP-TV iter 1 | betaTV = 0.105123 | dev = 169241.| min/mean/max = {0., 0.00107912, 0.373427}
MAP-TV iter 50 | betaTV = 0.835502 | dev = 338752.| min/mean/max = {0., 0.00586554, 14.3931}
MAP-TV iter 100 | betaTV = 1.01414 | dev = 406657.| min/mean/max = {0., 0.00708442, 14.9275}
MAP-TV iter 150 | betaTV = 1.03361 | dev = 414012.| min/mean/max = {0., 0.00721573, 15.0725}
MAP-TV iter 200 | betaTV = 1.03557 | dev = 414751.| min/mean/max = {0., 0.00722896, 15.1283}
MAP EM TV reconstruction complete.
```

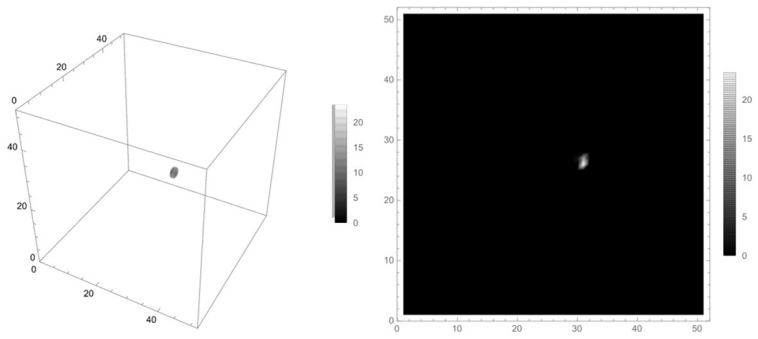
Figure 5. Iteration updates. Iteration updates for Fig. 6a show the algorithm maximizes the objective function as both the data fidelity term given by the deviation and the TV prior strength β decreases. The iteration updates show that the algorithm is maximizing the objective function as both the data fidelity term and the total variation prior strength betaTV (β) increases. The variable β begins with a value of 0.10 and converges to 1.04 after 200 iterations while the deviation begins with a value of 169241 and increases to 414715. The strength of the step size stays fixed at tauTV=0.1. At each iteration we monitored the Poisson deviance between the measured data y_i and the predicted projection $(Fx)_i$, defined by

$$D_{Pois}(y\|Fx) = 2\sum_i \left[(Fx)_i - y_i + y_i log\left(\frac{y_i}{(Fx)_i}\right)\right]$$

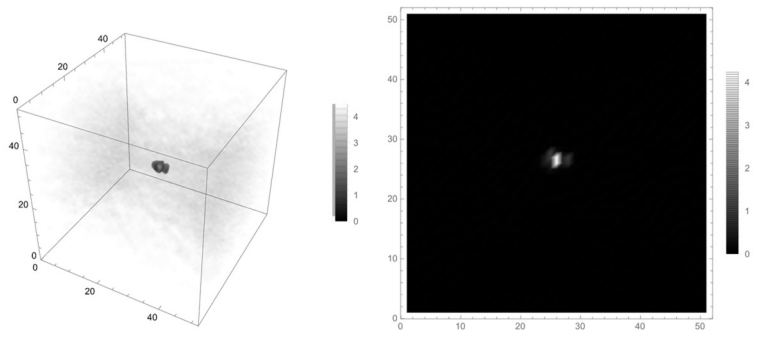
with a small stabilization constant added in practice to avoid division by zero. This quantity measures the agreement of the current reconstruction with the observed Poisson data.

Figure 6 shows the results of reconstructing 22000 Genat4 simulated events using Mathematica (Wolfram Research, Champaign, Illinois); comparing the MAP-TV reconstruction of a 51×51×51array at 2μ, 50μ, and 98μ away from the detector using the Siddon system matrix. The sinogram was constructed so that the simulated events were accumulated into a matrix of 501×501 units corresponding to projection coordinates (u, v). The central 51×51 bins of the sinogram were used for the sampled projections. The reconstruction of the point source 2μ from the detector (Fig. 5a) show a shift of approximately 7.5 voxels which gives a resolution of approximately 48/7.5 = 6.4μ per voxel. The reconstruction was obtained with 200 iterations fixing τ at 0.10 while β converged to 1.04, starting initially with a value of 0.10. The reconstruction of the point source 50μ from the detector (Fig. 5b) shows less resolution of the reconstructed point source than the result at 2μ from the detector. The reconstruction was obtained with the same 200 iterations and τ = 0.10; but in this case β converged to a value of 0.51. The reconstruction of the point source 98μ from the detector (Fig. 5c) shows good resoution in the transaxial slice but somewhat more dispersed in the 3D view.

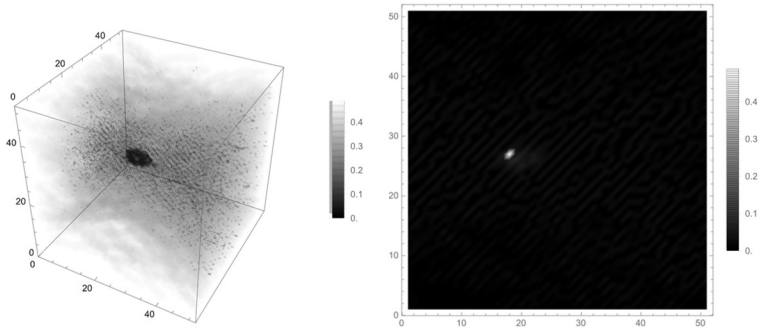
The total variation provides less dispersion but even still the result does not obtain a complete resolve of the point source as can be seen by a maximum grayscale amplitude of **0.49** compared to **4.5** for 50 μm from the detector and **23.5** for 2 μm from the detector. In this case the reconstruction was obtained with the same 200 iterations and τ = 0.10 while β converged to 0.41; but the sinogram was compressed to 201 units in order to obtain a more pointed central peak.



(a)



(b)



(c)

Figure 6. MAP-TV reconstructions of a 51×51×51 array using Siddon system matrix, 1000 iterations, and 201×201-unit sinogram: (a) 2 μ from the detector, (b) 50 μ from the detector, and (c) 98 μ from the detector. A shift in z of ≈ 7.5 voxels give a scale of ≈ 6.4 μ,

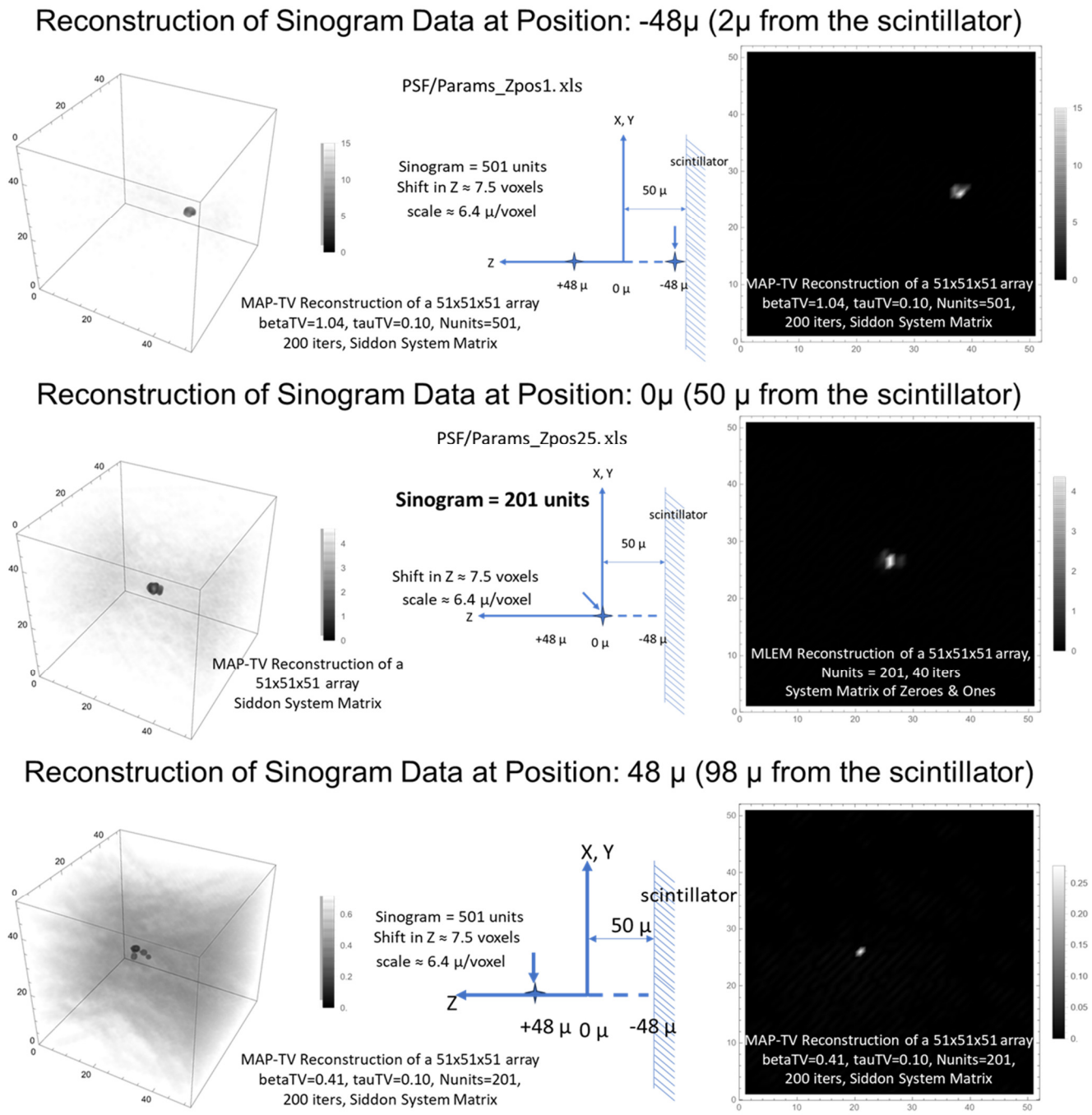


Figure 7. MAP-TV reconstructions of a 51×51×51 array using Siddon system matrix. Reconstruction of 22000 simulated events are shown for point sources: (a) 2μ from the detector with data stored in a sinogram 501×501-unit sinogram; (b) 50μ from the detector with data stored in a 201×201 sinogram; (c) 98μ from the detector with data stored in a 501×501 sinogram. All reconstructions were obtained with 200 iterations of the MAP-TV algorithm using a Siddon system matrix to jointly estimate the reconstruction values and the strength of the prior β . The reconstructions were obtained from simulated events that were accumulated into sinograms of 501×501 or 201×201 units corresponding to projection coordinates (u, v). The resolution is approximately 6.4μ.

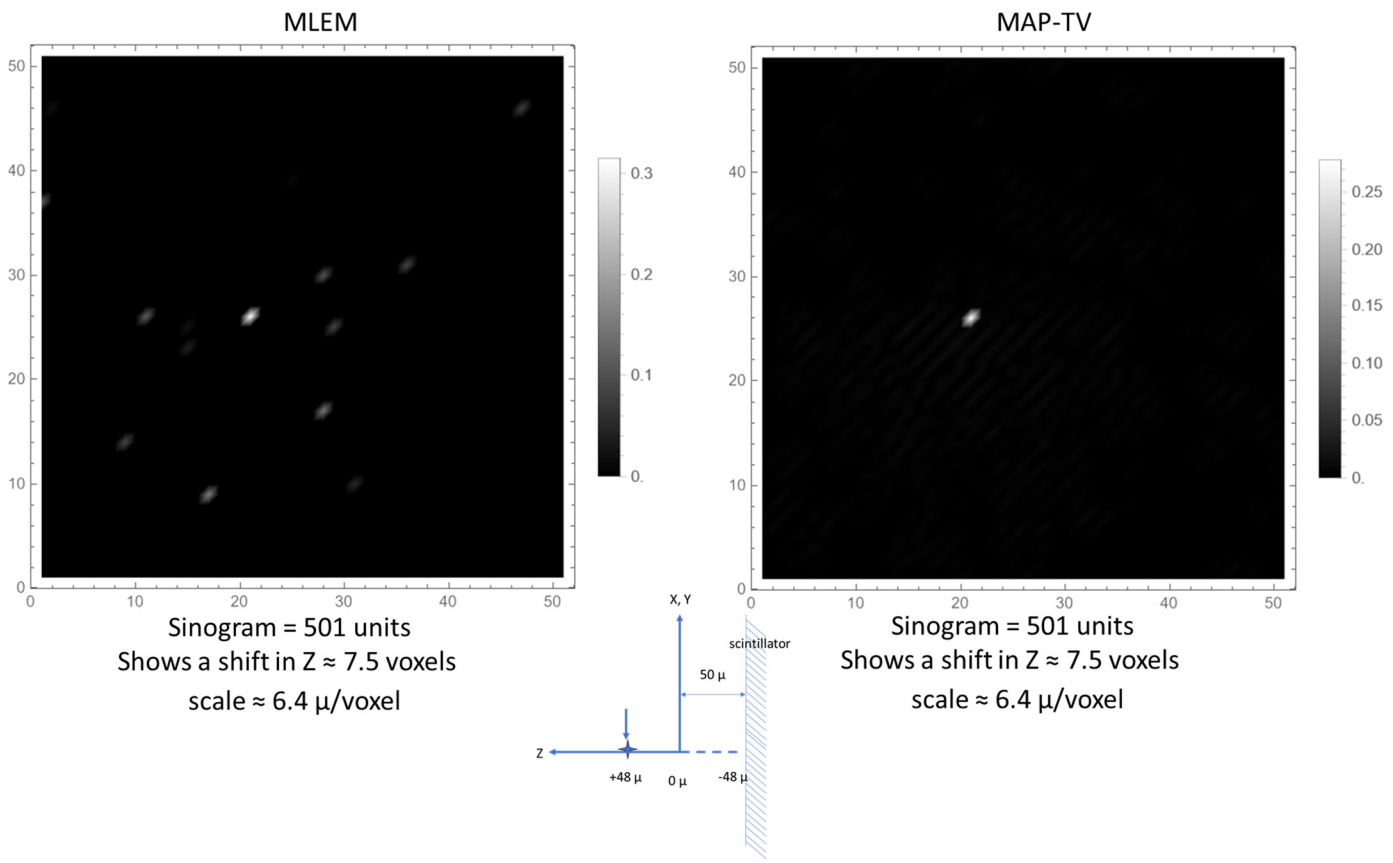


Figure 8. Reconstruction of 22000 simulated events comparing MLEM algorithm with a MAP-TV algorithm for a point source 98μ from the detector. The algorithms performed reconstructions a 51×51×51 arrays from data stored in a 501×501 sinogram using Siddon system matrix. The resolution is approximately 6.4μ. Notice the MAP-TV cleans up many background artifacts.

5. Discussion

We present a Maximum A Posteriori (MAP) - Total Variation (TV) reconstruction algorithm to process RLM data. Geant4 simulated data intended to mimic the acquisition of RLM data with a miroscope assembly were reconstructed using a MAP-TV algorithm that optimizes for both the reconstructed image values and the weighting factor β. The Geant4 simulation provides the 3D orientation on the crystal surface of each event, including the 3D position of the detected event and the angle of the detected photon. The generation of data in this way for the RLM problem was organized into a sinogram of limited angle projections for processing by an iterative MAP-TV Bayesian algorithm. The generation of data in this way is also ideal for implementation using a list mode (LM) iterative reconstruction algorithm [33] or a convoluton backprojection algorithm [34]. In our case we used Mathematica software (Wolfram Research, Champaign, Illinois) to perform the reconstruction of simulated point sources at various distances from the detector.

Hierarchical formulation. We first formulated the emission image reconstruction as a **hierarchical** maximum a posteriori (MAP) problem under a Poisson data model with a total variation (TV) prior. For fixed regularization parameter β, the reconstruction is obtained by minimizing a convex objective function consisting of the negative Poisson log-likelihood and a TV regularization:

$$\Phi(x,\beta) = \sum_i \sqrt{y_i log[(Fx)_i] - (Fx)_i} + \beta\, TV(x) \quad ,$$

$$\text{subject to } x \geq 0$$

For fixed regularization parameter β, the image is obtained by minimizing the sum of the negative Poisson log-likelihood and a total variation penalty, whereas to jointly estimate the image and regularization strength, β is treated as a hyperparameter with a prior distribution, leading to a joint objective in $\Phi(x,\beta)$. Since the Poisson negative log-likelihood is convex in x and if the total variation functional $TV(x)$ is also convex, $\Phi(x;\beta)$ is convex for fixed $\beta \geq 0$. The resulting optimization is performed via alternating minimization, in which the image is updated using a MAP-TV reconstruction step and β is updated in closed form from the current TV magnitude. (Comment: with a Gamma prior $p(\beta) \propto \beta^{a-1}e^{-b\beta}$, minimization with respect to β gives the closed-form update $\beta = \frac{a-1}{\mathrm{TV}(x)+b}$.). This leads naturally to an alternating optimization scheme in which x is updated by MAP-TV reconstruction and β is updated from the current image total variation. The algorithm uses a gradiant approach whereby defining the direction that minimizes the log likelihood and the total variatrion function. The hierarchical MAP estimation problem for Poisson emission data describes a sophisticated, probabilistic method for creating images from sensor data (like PET scans, SPECT, or astronomical imaging). It means that the reconstruction process is modeled as finding the "most likely" underlying image, based on both the noisy data collected and prior knowledge about what the image *should* look like. In general, hierarchical MAP–TV mathematical models are optimization frameworks that combine multi-scale structural decomposition with edge-preserving regularization. They are used to decompose images or signals into coarse, intermediate, and fine-level components. By applying TV minimization at different levels of a hierarchy, these models effectively separate structural information from noise or texture, enabling tasks like image restoration, segmentation, and inpainting.

Poisson deviance and discrepancy. Previously [11] we presented a dynamic regularization (DR) algorithm [7] where the targeted optimized variable and the regularization parameter β are both updated within each iteration. The parameter β is selected in a way to preserve a balance between the fidelity term and the prior. This underlying mathematical approach was originally developed by Ito et al. [17]. This report further investigates the development of MAP-TV algorithms with joint estimation of a reconstructed image

from Poisson data and a total variation prior with a regularization parameter β. Our developed MAP-EM-TV algorithms with an adaptive β update has evolved into something that sits **between** the following two ideas:

- It uses the **Poisson deviance** (equivalently, twice the KL divergence) as the feedback signal.
- It uses a **Morozov-style discrepancy target** to regulate the amount of TV regularization.

That combination is one of the more distinctive aspects of our reconstruction algorithm. It is not a standard textbook implementation of MAP-EM-TV, but rather a practical adaptive regularization strategy built on well-established statistical ideas. A key result presented demonstrates that minimizing the generalized Kullback-Leibler (KL) divergence is mathematically equivalent to finding the maximum likelihood estimate (MLE) under Poisson noise.

Duality. In the future we plan to investgate te developent of an algorithm using duality, where we consider emission image reconstruction under a Poisson noise model formulated as a convex optimization problem with a total variation (TV) constraint. Specifically, the image is obtained by minimizing the negative Poisson log-likelihood subject to a bound on the total variation, which enforces spatial regularity while preserving edges. This constrained formulation admits a saddle-point representation in which the TV term is expressed via its dual, leading to a coupled primal–dual optimization problem. The resulting algorithm alternates between updates of the image (primal variable), driven by the data fidelity term, and updates of a dual field associated with the TV constraint, implemented through gradient ascent and projection onto a unit ball. Such primal–dual schemes, including variants of the Chambolle–Pock algorithm [35, 36], provide efficient and convergent methods for large-scale tomographic reconstruction and naturally balance noise suppression and resolution without explicit tuning of a penalty parameter.

Acknowledgment. This work was supported by the National Institutes of Health under Grant NIH 1 R21 EB035205-01. We used ChaptGPT (OpenAI) to assist with generating Wolfram Mathematica code (Wolfram Research, Champaign, Illinois) for implementing the reconstruction algorithms.

6. References

[1] G. Pratx, K. Chen, C. Sun, L. Martin, C. M. Carpenter, P. D. Olcott, L. Xing. Radioluminescence microscopy: measuring the heterogeneous uptake of radiotracers in single living cells. PLoS One (2012) Oct 3;7(10):e46285. DOI: 10.1371/journal.pone.0046285.

[2] S. Khan, J. H. Shin, V. Ferri, N. Cheng, J. E. Noel, C. Kuo, J. B. Sunwoo, G. Pratx. High-resolution positron emission microscopy of patient-derived tumor organoids. Nature Communications. 2021;12(1):1–11.

[3] G. Pratx, K. Chen, C. Sun, M. Axente, L. Sasportas, C. Carpenter, L. Xing. High-resolution radioluminescence microscopy of ^{18}F-FDG uptake by reconstructing the β-ionization track. Journal of Nuclear Medicine. (2013), 54 (10) pp. 1841-1846; DOI: https: //doi.org/10.2967/jnumed.112.113365.

[4] Y. Sung. Snapshot projection optical tomography. Physic Review Applied. 13(5): May 20 (2020), 054048; DOI: https: //doi.org/10.1103/PhysRevApplied.13.054048

[5] Y. Sung. Snapshot three-dimensional absorption imaging of microscopic specimens. Physic Review Applied. (2021);15(6):064065; DOI: https://doi.org/10.1103/PhysRevApplied.15.064065

[6] G.L. Zeng, R. Clack. Three-Dimensional Reconstruction Techniques in Medical Imaging. Class Notes: Bioengineering 652, Winter 1992, University of Utah.

[7] R.H. Huesman, G.T. Gullberg, W.L. Greenberg, T.F. Budinger. User's Manual - Donner Algorithms for Reconstruction Tomography. Lawrence Berkeley Laboratory Publication PUB-214, 1977.

[8] G.T. Gullberg, G.L. Zeng, B.M.W. Tsui, J.T. Hagius. An iterative reconstruction algorithm for single photon emission computed tomography with cone beam geometry. Int J of Imag Sys & Techn 1:169-186, 1989.

[9] R.L. Siddon. Fast calculation of the exact radiological path for a three-dimensional CT array. Med Phys. 1985 Mar-Apr;12(2):252-5.

[10] V.Y. Panin, G.L. Zeng, G.T. Gullberg. Total variation regulated EM algorithm. IEEE Trans on Nucl Sci 46:2202-2210, 1999.

[11] M. Abdalah, D. Mitra, S. Chen, R. Boutchko, G.T. Gullberg. Optimization of regularization parameter in a reconstruction algorithm. In: 2013 IEEE Nuclear Science Symposium and Medical Imaging Conference Record (NSS/MIC), Oct. 27 – Nov. 2, 2013, Seoul, Korea. **4 pages DOI:** 10.1109/NSSMIC.2013.6829217

[12] P.C. Hansen. Analysis of discrete ill-posed problems by means of the L-curve, SIAM Rev., Vol. 34, No. 4, pp. 561580,1992.

[13] G.D. Rodriguez. Thesis, An algorithm for estimating the optimal regularization parameter by the L-curve, Rend. Mat. 25, pp. 69-84, 2005.

[14] J. Chamorro-Servent, J. Aguirre, J. Ripoll, J. Vaquero, M. Desco. Feasibility of U-curve method to select the regularization parameter for fluorescence diffuse optical tomography in phantom and small animal studies, Opt. Express 19, 11490-11506, 2011.

[15] S. W. Anzengruber, R. Ramlau. Morozov's discrepancy principle for Tikhonov-type functionals with nonlinear operators, Inverse Problems, 25, 0266-5611, 2009.

[16] B. Jin, Y. Zhao, J. Zou. Iterative parameter choice by discrepancy principle, IMA Journal of Numerical Analysis, doi: 10.1093/imanum/drr051, 2012.

[17] K. Ito, B. Jin, T. Takeuchi. A regularization parameter for nonsmoothed Tikhonov regularization. SIAM Journal on Scientific Computing, 33:3, 1415-1438, 2011.

[18] E.Y. Sidky, J.H. Jorgensen, X. Pa. Convex optimization problem prototyping for image reconstruction in computed tomography with the Chambolle-Pock algorithm, Phys Med Biol: 2012 May 21.

[19] E. Van Den Berg, M. P. Friedlander. Probing the Pareto frontier for basis pursuit solutions, SIAM J. Sci. Comput., Vol. 31, No. 2, pp. 890912, 2008.

[20] K. Ito, B. Jin, T. Takeuchi. Multi-parameter Tikhonov regularization, eprint arXiv:1102.1173, 2011.

[21] M. Bertero, P. Boccacci, G. Talenti, R. Zanella, L. Zanni. A discrepancy principle for Poisson data, Inverse Problems, 26(10), 105004, 2010.

[22] F. Bevilacqua, A. Lanza, M. Pragliola, F. Sgallari. Nearly exact discrepancy principle for low-count Poisson image restoration. Journal of Imaging, 8(1), 2022.
[23] B. Sixou. Adaptative regularization parameter for Poisson noise with a bilevel approach: application to spectral computerized tomography. Inverse Problems in Science and Engineering, 29:11, 1519-1536, DOI: 10.1080/17415977.2020.1864348
[24] L. Calatroni, A. Lanza, M. Pragliola, F. Sgallari. Adaptive parameter selection for weighted-TV image reconstruction problems. J. Phys.: Conf. Ser. 1476 012003, 2020.
[25] S.W. Anzengruber, R. Ramlau, Morozov's discrepancy principle for Tikhonov-type functionals with nonlinear operators. Inverse Problems 26 025001, 2010. **DOI**: 10.1088/0266-5611/26/2/025001.
[26] S. Kullback, R.A. Leibler. On information and sufficiency. The Annals of Mathematical Statistics, 22(1), 79–86. http://www.jstor.org/stable/2236703
[27] I. Csiszár. Why least squares and maximum entropy? An axiomatic approach to inference for linear inverse problems. Ann. Statist. 19 (4) 2032 - 2066, December 1991. https://doi.org/10.1214/aos/1176348385
[28] L. A. Shepp, Y. Vardi. Maximum likelihood reconstruction for emission tomography. IEEE Trans Med Imaging. 1982;1(2):113-22. doi: 10.1109/TMI.1982.4307558
[29] W. H. Richardson. Bayesian-based iterative method of image restoration. J Opt Soc Am 62:55-59, 1972.
[30] L.B. Lucy. An iterative technique for the rectification of observed distributions. Astronomical J, 79:75, 1974.
[31] V.A. Morozov. On the solution of functional equations by the method of regularization. Soviet Mathematics Doklady 7, 414-417, 1966.
[32] T. Hohage, F. Werner. Inverse problems with Poisson data: statistical and regularization theory. Inverse Problems. 32 (9), 093001 (56pp) 2016.
[33] L. Parra, H. H. Barrett, List-mode likelihood: EM algorithm and image quality estimation demonstrated on 2-D PET, IEEE Trans Med Imag, vol. 17, no. 2, pp. 228-235, April 1998, doi: 10.1109/42.700734.
[34] G.T. Gullberg, G.L. Zeng, Y. Seo, G. Pratx, Y. Sung. List mode processing of RLM data for fully 3D reconstruction using convolution filtering of 2D projections. Proceedings, 18th International Meeting on Fully Three-Dimensional Image Reconstruction in Radiology and Nuclear Medicine; (2025), Shanghai, China.
[35] A. Chambolle, T. Pock (2011-05-01). A first-order primal-dual algorithm for convex problems with applications to imaging. Journal of Mathematical Imaging and Vision. 40 (1): 120–145, doi:10.1007/s10851-010-0251-1
[36] Z Zhang, B Chen, D Xia, E. Y Sidky, X Pan. Directional-TV algorithm for image reconstruction from limited-angular-range data. Med Image Anal 2021 May 102030 doi 10.1016/j. media 201.102030.

Appendix I (Abstract and Two-Page Summary)

3D Reconstruction of Radioluminescence Microscopy Data via MAP with Optimized Total Variation Regularization

A new microscopic nuclear imaging instrument using radioluminescence microscopy (RLM) has been developed to image living cells at high resolution with radiotracers tagged with positron emitters. These are used in biological samples to mimic clinical applications to evaluate their effectiveness to target receptors, antigens, and proteins that are overexpressed during growth of cancerous cells. The methodology has demonstrated impressive results for planar applications using optical reconstruction of β-ionization tracks (ORBIT); however, 3D imaging presents a challenging problem of reconstructing the 3D distribution of the source requiring backtracking of the ionization track of the positron in the scintillator to determine its position and 3D orientation at the detector surface of the instrument. Four microlens arrays are used to record projections, limiting the angle of projection measurements to 360° sampling in the azimuthal angle ϕ and 40° sampling in the polar angle θ. The angle information acts as a collimator, but the limited angle acquisition presents a challenging reconstruction problem which increases in difficulty for sources further from the detector. We consider reconstruction as a hierarchical MAP estimation problem for Poisson emission data. For fixed regularization parameter β, the image is obtained by minimizing the sum of the negative Poisson log-likelihood and a total variation penalty, whereas to estimate β jointly with the image, we introduce a hyperprior function of β, yielding a joint objective function in terms of the reconstruction variables and the regulation parameter β. This leads naturally to an alternating optimization scheme in which x is updated by MAP-TV reconstruction and β is updated from the current image total variation. Good results are demonstrated using Geant4 simulations of point sources as far as 98 μ from the face of the detector. The novelty is the development of a microscopy application to image individual cells three-dimensionally with radiotracers intended for human imaging and therapeutic applications.
Support from the National Institutes of Health under Grant NIH 1 R21 EB035205-01.

Panin, VY, Zeng, GL, Gullberg GT 1999, 'Total variation regulated EM algorithm', *IEEE Trans on Nucl Sci* 46:2202-2210.
Sung, Y 2021, Snapshot three-dimensional absorption imaging of microscopic specimens,' *Physic Review Applied* 15(6):064065.
Huesman, RH, Gullberg, GT, Greenberg, WL, Budinger, TF 1977 'Users Manual - Donner Algorithms for Reconstruction Tomography,' PUB-214, Lawrence Berkeley Laboratory Publication.
Siddon, RL 1985, Fast calculation of the exact radiological path for a three-dimensional CT array,' *Med Phys* 12:252-255.
Abdalah, M, Mitra, D, Chen, S, Boutchko, R, Gullberg, GT 2013 'Optimization of regularization parameter in a reconstruction algorithm,' In: 2013 IEEE NSS/MIC Record, Oct. 27 – Nov. 2, 2013, Seoul, Korea.

Key Words: limited angle reconstruction, Radioluminescence Microscopy

3D Reconstruction of Radioluminescence Microscopy Data via MAP with Optimized Total Variation Regularization

Grant T. Gullberg,[1] *Life Fellow, IEEE,* Qihui Lyu,[2] Guillem Pratx,[3] *Member of IEEE*, Youngho Seo,[1] *Senior Member, IEEE,* Yongjin Sung,[4] *Member of IEEE*

[1]Department of Radiology and Biomedical Imaging, University of California, San Francisco, CA, USA
[2]Department of Radiation Oncology, University of California, San Francisco, CA, USA
[3]Department of Radiation Oncology, Stanford University, CA, USA
[4]Department of Biomedical and Mechanical Engineering, University of Wisconsin-Milwaukee, Milwaukee, WI, USA

1 Introduction

We have developed a Maximum a Posteriori (MAP) with Total Variation (TV) [1] reconstruction algorithm to process 3D radioluminescence microscopy (RLM) [2] data, enabling the characterization of radionuclide distributions in thick living biological specimens using the same clinical radiotracers used in clinical imaging and therapy. The data processing for 3D RLM starts with calculating the 3D orientations of scintillation tracks from four projection images of micro-lens arrays (MLAs) (Fig. 1). Backtracking the scintillation tracks provides the position (p_x, p_y, p_z) and the angle (u_x, u_y, u_z) that the positron enters the scintillation detector (Fig. 2b). Events are stored into a sinogram of coordinates (ϕ, θ, u, v) (Fig. 2c), forming limited angle projections. The goal is to develop appropriate priors that suppress noise and artifacts while maintaining accurate localization of radiotracer accumulation. The reconstruction is an iterative method that obtains optimized image values and a regularization parameter. We evaluate our method by reconstructing events simulated using Geant4.

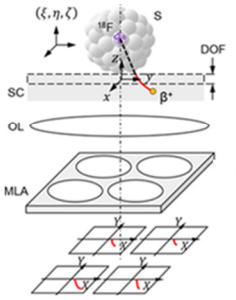


Fig. 1. Simulation geometry. S: Sample; SC: scintillation crystal; OL: objective lens; MLA:micro-lens array; DOF:depth of field.

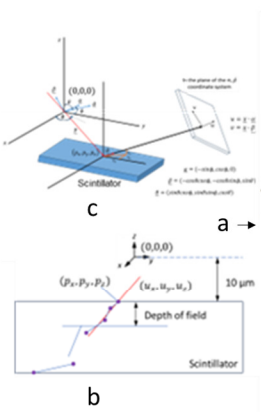
2 Methods

2.1 Data Input

A Geant4 simulation was performed to provide RLM projection data for evaluating our joint estimation algorithm. Geant4 simulated data provided the 3D position and angle of the detected positron on the detector surface (Fig. 2). The ^{18}F point source was positioned at varying distances from the detector surface, thereby simulating radiotracer distributions in a thick specimen. The results of the Geant4 simulation were included in an Excel file (Fig. 2). The task was to reconstruct the 3D distribution of point sources positioned at 2, 50, and 98 μms from the detector surface.

2.2 Sinogram

Each projection event was extracted from the Excel file and binned into a sinogram. For the angle θ, the bin index was determined by $\mathrm{I}_\theta = \mathrm{Int}\left[\frac{9}{45.}\theta(\mathrm{I})\right] + 1$, for I=1,…,9. For the angle ϕ the bin index was determined by $\mathrm{I}_\phi = \mathrm{Int}\left[\frac{18.}{360.}\phi(\mathrm{I})\right] + 1$, for $\mathrm{I} = 1, \ldots, 18$. If we let $\mathrm{umax} = \mathrm{Max}[\mathrm{u(i)}, \mathrm{i} = 1, \ldots, 22000$ events] and $\mathrm{vmax} = \mathrm{Max}[\mathrm{v(i)}, \mathrm{i} = 1, \ldots, 22000$ events], where i is the index number of the simulated events, then the bin indices for u and v were determined by $\mathrm{I}_\mathrm{u} = \mathrm{Int}\left[\frac{\mathrm{Nbins}}{2\,\mathrm{Maxuv}}\mathrm{u(i)} + \frac{\mathrm{Nbins}}{2} + 1\right]$, $\mathrm{I}_\mathrm{v} = \mathrm{Int}\left[\frac{\mathrm{Nbins}}{2\,\mathrm{Maxuv}}\mathrm{v(i)} + \frac{\mathrm{Nbins}}{2} + 1\right]$, where $\mathrm{Maxuv} = \max[\mathrm{umax}, \mathrm{vmax}] + 1$.



px	py	pz	ux	uy	uz
-6.6869	-4.55169	10	-0.49148215	-0.283579273	0.823424613
-7.16131	-4.8801	10	-0.442679566	-0.167026624	0.880986327
-4.33504	8.89787	10	-0.22176683	0.602658076	0.766565533
1.03895	-8.02056	10	0.093476932	-0.519691683	0.849224716
-12.8715	-6.914	10	-0.475733798	-0.376854621	0.794769116
-2.0348	-5.62691	10	-0.247797176	-0.469645882	0.847366098
1.4596	-0.0622373	10	-0.044336068	0.259171266	0.964813229
-2.83432	2.99764	10	0.148953763	0.334714288	0.930472526
12.5718	5.32271	10	0.422621511	0.24598376	0.872286105
6.46098	-3.04908	10	0.157700258	-0.485930422	0.859652403
-13.2006	-183.704	10	-0.272233475	-0.228548627	0.934694849
-121.05	142.106	10	-0.300932326	0.31164161	0.901287547
-198.544	-581.019	10	0.059525976	-0.580661132	0.811966322
-0.525323	4.5085	10	0.012283154	0.429559081	0.902955215
-7.98275	-3.77677	10	-0.554888661	-0.28872934	0.780214037
0.275827	-4.28141	10	0.169855386	-0.394757425	0.90294835
4.88445	0.340886	10	0.507783984	0.179961334	0.842478097
5.55379	4.50009	10	0.494706929	0.376457808	0.783290861
2.11716	-0.81613	10	0.192227747	-0.314837254	0.929476195
8.13453	3.16254	10	0.501189081	0.251783452	0.827897698
-4.34978	0.765566	10	-0.518493545	-0.05879772	0.853057602
98.3586	-193.493	10	-0.571068214	0.237945749	0.785660815
-111.892	130.115	10	-0.40269678	0.277888575	0.872131437
9.46601	-2.06311	10	0.370989185	-0.176320645	0.911744512
-5.18696	-6.17182	10	-0.444477294	-0.203947705	0.872264449
-2.78902	-9.33404	10	-0.006641684	-0.578047327	0.815976721
0.105605	3.45717	10	0.037694929	0.306634311	0.951080697
10.1213	-3.28258	10	0.535596733	-0.108055141	0.837532224
0.628075	7.91237	10	0.039833974	0.636981293	0.769849392
-3.31912	-0.774675	10	-0.213543251	0.095524664	0.972252189
11.4083	8.98358	10	0.517331901	0.138940763	0.844430677

Fig. 2. Example of Geant4 simulated positron trails for a ^{18}F point source located 10 μm from the crystal surface. (a) Geant simulated events were compiled in Excel files to be processed for sources 2, 50, 98 μm from the detector. (b) Each row represents one scintillation trail, and the columns (p_x, p_y, p_z) are the coordinates of the starting point of a scintillation trail (unit: μm), and (u_x, u_y, u_z) is the direction of the positron trail at the surface: $\phi = \arcsin\left[u_y/\sqrt{u_x^2 + u_y^2}\right]$, $\theta = \arccos\left[u_z/\sqrt{u_x^2 + u_y^2 + u_z^2}\right]$. (c) The coordinate (0,0,0) is the center of the imaging coordinate system, 10 μm from the detector surface.

2.3 Algorithm

In our algorithm we treat β as a variable and jointly estimate x and β by alternating the minimization of: $\mathrm{J(x,\beta)} = \mathrm{L(x)} + \beta\,\mathrm{TV(x)} - \log \mathrm{p(\beta)}$, so $x^{(k+1)} = \arg\min_x J(x, \beta^k)$ and $\beta^{(k+1)} = \arg\min_\beta J(x^{k+1}, \beta)$; where $\mathrm{TV(x)} = \sum_j \sqrt{\left(\nabla_x x_j\right)^2 + \left(\nabla_y x_j\right)^2 + \left(\nabla_z x_j\right)^2}$ is the total variation prior, $\mathrm{L(x)} = \sum_i [(Fx)_i - y_i \log((Fx)_i)]$ is the negative Poisson log-likelihood, $y = Fx$ are projection data, x is the image matrix, and F is the system matrix with line length weighting [3,4]. A deviance-based adaptive beta update is used which aims to keep the Poisson deviance D near a target value. Poisson deviance measures how well predicted counts match measured counts under Poisson noise, or how much worse is the model than a model that fits every data point perfectly. TV is applied after the EM update as an additive correction.

For Iteration k:

1. x update:

$$x_{EM}^{(k)} = \prod_{x\geq 0}\left[x^{(k)} \cdot \frac{F^T\left(\frac{y}{(Fx^{(k)}+\epsilon_{proj})}\right)}{F^T 1+\epsilon_{proj}}\right]$$

$$x^{(k+1)} = \prod_{x\geq 0}\left[x_{EM}^{(k)} - \frac{\tau_{TV}\beta^{(k)}}{max\left|\nabla TV_\epsilon\left(x_{EM}^{(k)}\right)\right| + 10^{-12}}\nabla TV_\epsilon\left(x_{EM}^{(k)}\right)\right]$$

2. β update:

$$\beta^{(k+1)} = 0.8\beta^{(k)} + 0.2clip\left(\beta^{(k)}\left(\frac{D_{target}}{D^{(k+1)}}\right)^\gamma, \beta_{min}, \beta_{max}\right)$$

$$D(x^k) = 2\sum_i [y_i\, log\left[\frac{y_i}{(F\,x^k)_i}\right] - y_i + (F\,x^k)_i]$$

3 Results

Figure 3 shows the results of reconstructing 22000 Geant4 simulated events using Mathematica (Wolfram Research, Champaign, Illinois); comparing the MAP-TV with prior

optimization reconstruction of a 51×51×51array 2, 50, and 98 μm away from the detector using the Siddon system matrix. The sinogram was constructed so that the simulated events for projection angles (ϕ, θ) were accumulated into a matrix of 201×201 units corresponding to projection coordinates (u, v). The central 51×51 bins of the sinogram were used for the sampled projections. The reconstruction of the point source 2 μm from the detector (Fig. 5a) showed a shift to the right of approximately 7.5 voxels which gives a resolution of approximately 48/7.5 = 6.4 μm per voxel. The reconstruction was obtained with 1000 iterations initially starting β at: (a) 2.0 converging to 0.85; (b) 2.0 converging to 0.50; and (c) 0.10 converging to 0.41. The reconstruction of the point source at 50 μm from the detector (Fig. 3b) shows lower resolution than that of the point source at 2 μm from the detector. The reconstruction of the point source 98 μm from the detector (Fig. 5c) shows good resoution in the transaxial slice but somewhat more dispersed in the 3D view. The total variation provides less dispersion but even still the result does not obtain a complete resolve of the point source as can be seen by a maximum grayscale amplitude of **0.49** compared to **4.5** for 50 μm from the detector and **23.5** for 2 μm from the detector.

4 Discussion

We present an algorithm for 3D reconstruction of RLM data via Maximum a Posteriori with optimized TV regularization. Geant4 simulated data were intended to mimic the acquisition of 3DRLM data obtained with a miroscope assembly. The Geant4 simulation provides the 3D orientation on the crystal surface of each event, including the 3D position of the detected event and the angle of the detected positron. The generation of the simulated 3D RLM data is organized into a sinogram of limited angle projections to be processed by an iterative MAP-TV Bayesian algorithm. The reconstruction was formulated as a hierarchical maximum a posteriori (MAP) problem under a Poisson data model with a TV prior. The optimization was performed via alternating minimization, in which the image is updated using a MAP reconstruction step and β is updated using a deviance-based adaptive beta update which aims to keep the Poisson deviance D near a target value.

In the future we plan to consider the image reconstruction under a Poisson noise model as a convex optimization problem with a TV constraint. This constrained formulation admits a saddle-point representation in which the TV term is expressed via its dual, leading to a coupled primal–dual optimization [5]. The algorithm alternates between updates of the image (primal variable), and updates of a dual variable associated with the TV constraint. Such primal–dual schemes provide efficient and convergent methods for large-scale tomographic reconstruction and naturally balance noise suppression and resolution without explicit tuning of a penalty parameter.

Table 1. Iteration updates for Fig. 3a show the algorithm maximizes the objective function as both the data fidelity term given by the deviation and the TV prior strength β decreases.

Map-TV Iteration	β	Deviation	Image min/mean/max
1	1.91034	1162940	{0., 0.0182435, 0.371918}
50	0.99332	481231	{0., 0.00830965, 23.3693}
100	0.86874	426549	{0., 0.00733767, 23.5103}
150	0.85616	420927	{0., 0.00723617, 23.4975}
200	0.85478	420312	{0., 0.00722503, 23.5115}

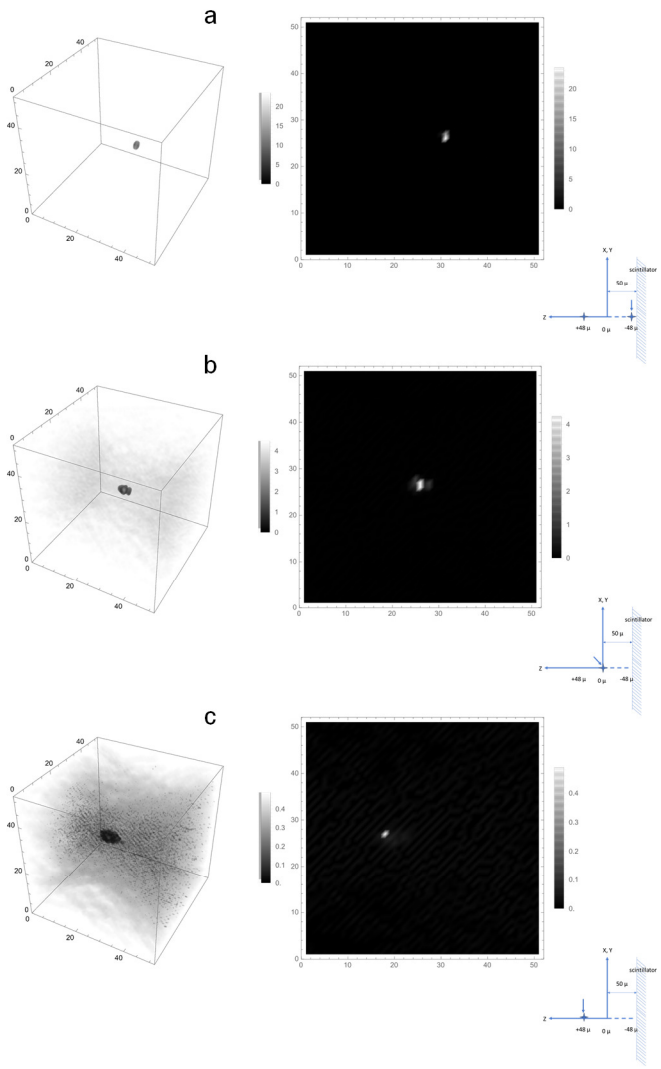


Fig. 3. MAP-TV reconstructions of a 51×51×51 array using Siddon system matrix, 1000 iterations, and 201×201-unit sinogram: (a) 2 μ from the detector, (b) 50 μ from the detector, and (c) 98 μ from the detector. A shift in z of ≈ 7.5 voxels give a scale of ≈ 6.4 μ,

Acknowledgment. This work was supported by the National Institutes of Health under Grant NIH 1 R21 EB035205-01. We used ChaptGPT (OpenAI) to assist with generating Wolfram Mathematica code (Wolfram Research, Champaign, Illinois) for implementing the reconstruction algorithms.

References

[1] V Y Panin, G L Zeng, G T Gullberg. Total variation regulated EM algorithm. IEEE Trans on Nucl Sci 46:2202-2210, 1999.
[2] Y Sung. Snapshot three-dimensional absorption imaging of microscopic specimens Physic Review Applied. 15(6):064065, 2021.
[3] R H Huesman, G T Gullberg, W L Greenberg, T F Budinger. "Users Manual - Donner Algorithms for Reconstruction Tomography." Lawrence Berkeley Laboratory Publication PUB-214, 1977.
[4] R L Siddon. "Fast calculation of the exact radiological path for a three-dimensional CT array." Med Phys 12:252-255, 1985.
[5] M Abdalah, D Mitra, S Chen, R Boutchko, G T Gullberg. Optimization of regularization parameter in a reconstruction algorithm. In: 2013 IEEE NSS/MIC Record, Oct. 27 – Nov. 2, 2013, Seoul, Korea.

Appendix II (Poster)

Appendix III (Mathematica Code)

3D Reconstruction of Radioluminescence Microscopy Data via MAP with Optimized Total Variation Regularization

Grant T. Gullberg[1], Qihui Lyu[2], Guillem Pratx[3], Youngho Seo[1,4], and Yongjin Sung[5]

[1]Department of Radiology and Biomedical Imaging, University of California, San Francisco, San Francisco, CA, USA
[2]Department of Radiation Oncology, University of California, San Francisco, San Francisco, CA, USA
[3]Department of Radiation Oncology, Stanford University, Stanford, CA, USA
[4]Department of Nuclear Engineering, University of California, Berkeley, Berkeley, CA, USA
[5]Department of Biomedical and Mechanical Engineering, University of Wisconsin-Milwaukee, Milwaukee, WI, USA

1 Introduction

We have developed a Maximum a Posteriori (MAP) with Total Variation (TV) [1] reconstruction algorithm to process 3D radioluminescence microscopy (RLM) [2] data, enabling the characterization of radionuclide distributions in thick living biological specimens using positron radiotracers used in clinical imaging. The data processing for 3D RLM starts with calculating the 3D orientations of scintillation tracks from four projection images of a micro lens array (MLA) (Fig. 1). Backtracking the scintillation tracks provides the position (p_x, p_y, p_z) and the angle (u_x, u_y, u_z) that the positron enters the scintillation detector (Fig. 2b). Events are stored in a sinogram (Fig. 2a) with coordinates (ϕ, θ, u, v) forming limited angle projections. The goal is to develop appropriate priors that suppress noise and artifacts while maintaining accurate localization of radiotracer accumulation. **The reconstruction is an iterative method that obtains optimized image values and an optimized regularization parameter.**

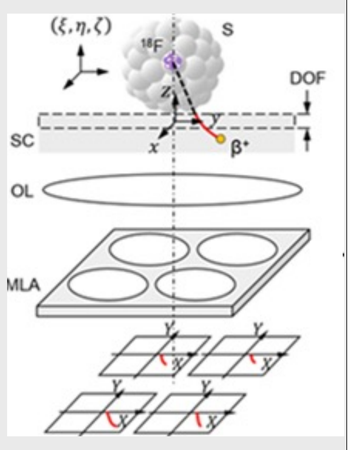


Figure 1. Simulation geometry. S: Sample; SC: scintillation crystal; OL: objective lens; MLA: micro lens array; DOF: depth of field.

2 Methods

2.1 Data Input

A Geant4 simulation was performed to provide RLM projection data [2] for evaluating our joint estimation algorithm. Geant4 simulated data provided the 3D position and angle of the detected positron on the detector surface (Fig. 2b). The ^{18}F point source was positioned at varying distances from the detector surface, thereby simulating radiotracer distributions in a thick specimen. The results of the Geant4 simulation were included in an Excel file (Fig. 2a). The task was to reconstruct the 3D distribution of point sources positioned at 2, 50, and 98 μms from the detector surface..

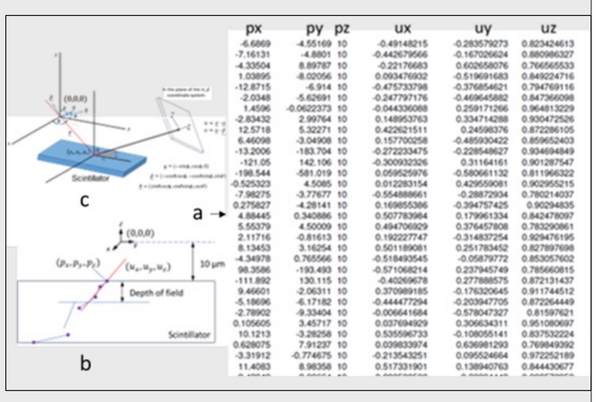


Figure 2. Example of Geant4 simulated positron trails for a ^{18}F point source located 10 μm from the crystal surface. (a) Events were compiled in Excel files for sources 2, 50, 98 μm from the detector. (b) Each row represents one scintillation trail, and the columns (p_x, p_y, p_z) are the coordinates of the starting point of a scintillation trail (unit: μm), and (u_x, u_y, u_z) is the direction of the positron trail at the surface: $\phi = \arcsin[u_y/\sqrt{u_x^2+u_y^2}]$, $\theta = \arccos[u_z/\sqrt{u_x^2+u_y^2+u_z^2}]$. (c) The coordinate (0,0,0) is the center of the imaging coordinate system, 10 μm from the detector

2.2 Sinogram

Each projection event was extracted from the Excel file and binned into a sinogram. For the angle θ, the bin index was determined by $I_\theta = \text{Int}\left[\frac{9.}{45.}\theta(I)\right] + 1$, for I=1,…,9. For the angle ϕ the bin index was determined by $I_\phi = \text{Int}\left[\frac{18.}{360.}\phi(I)\right] + 1$, for $I = 1, \ldots, 18$. If we let $\text{umax} = \text{Max}[u(i), i = 1, \ldots, 22000 \text{ events}]$ and $\text{vmax} = \text{Max}[v(i), i = 1, \ldots, 22000 \text{ events}]$, where i is the index number of the simulated events, then the bin indices for u and v were determined by $I_u = \text{Int}\left[\frac{\text{Nbins}}{2\,\text{Maxuv}}u(i) + \frac{\text{Nbins}}{2} + 1\right]$, $I_v = \text{Int}\left[\frac{\text{Nbins}}{2\,\text{Maxuv}}v(i) + \frac{\text{Nbins}}{2} + 1\right]$, where $\text{Maxuv} = \max[\text{umax}, \text{vmax}] + 1$.

2.3 Algorithm

In our algorithm we treat β as a variable and jointly estimate x and β by alternating the minimization [3] of: $J(x, \beta) = L(x) + \beta\, TV(x) - \log p(\beta)$, so $x^{(k+1)} = \arg\min_x J(x, \beta^k)$ and $\beta^{(k+1)} = \arg\min_\beta J(x^{k+1}, \beta)$; where $TV(x) = \sum_j \sqrt{(\nabla_x x_j)^2 + (\nabla_y x_j)^2 + (\nabla_z x_j)^2}$ is the total variation prior [1], $L(x) = \sum_i \left[(Fx)_i - y_i \log((Fx)_i)\right]$ is the negative Poisson log-likelihood, $y = Fx$ are projection data, x is the image matrix, and F is the system matrix with line length weighting [4, 5]. A deviance-based adaptive beta update [6] is used which aims to keep the Poisson deviance D near a target value. Poisson deviance measures how well predicted counts match measured counts under Poisson noise, or how much worse is the model than a model that fits every data point perfectly. TV is applied after the EM update as an additive correction.

For iteration k

1. **Step 1 – x update:**

$$x_{EM}^k = \prod\nolimits_{x\geq 0} x^{(k)} \cdot \frac{F^T\left(\frac{y}{(Fx^{(k)} + \epsilon_{proj})}\right)}{F^T 1 + \epsilon_{proj}}$$

$$\nabla TV_\in(x_{EM}^k) = -\nabla \cdot \left(\frac{\nabla x}{\sqrt{|\nabla x|^2 + \epsilon_{TV}^2}}\right)$$

$$x^{(k+1)} = \prod\nolimits_{x\geq 0}\left[x_{EM}^{(k)} - \frac{\tau_{TV}\,\beta^{(k)}}{max|\nabla TV_\epsilon(x_{EM}^{(k)})| + 10^{-12}}\nabla TV_\epsilon(x_{EM}^{(k)})\right]$$

2. **Step 2 – β update:**

$$\beta^{(k+1)} = 0.8\beta^{(k)} + 0.2\; clip\left[\beta^k\left(\frac{D_{target}}{D(x^{k+1})}\right)^\gamma, \beta_{min}, \beta_{max}\right]$$

Deviance

$$D(x^{k+1}) = 2\sum_i\left[y_i log\left[\frac{y_i}{(Fx^{k+1})_i}\right] - y_i + (Fx^{k+1})_i\right]$$

where ϵ is a very small number to prevent division by zero in the algorithm update. Notice each iteration involves first updating x, which is used to update β.

3 Results

Figure 3 compares MAP-TV and prior optimization for reconstructing 22,000 Geant4-simulated events (51×51×51 array) at **2, 50, and 98 μm** from the detector using Mathematica and a Siddon system matrix. Angles (ϕ, θ) were binned into a 201×201 sinogram matrix (u, v), with the central 51×51 bins sampled for each projection.

Reconstruction Results

Reconstructions (1,000 iterations) converged β from: 2.0 to 0.85 (Fig. 3a); 2.0 to 0.50 (Fig. 3b); and 0.10 to 0.41 (Fig. 3c).

- **At 2 μm (Fig. 3a):** Shows a 7.5-voxel rightward shift, yielding a resolution of ~ 6.4 μm/voxel (48/7.5) with peak grayscale amplitude of 23.5.
- **At 50 μm (Fig. 3b):** Shows lower resolution than at 2 μm, with a peak grayscale amplitude of 4.5.
- **At 98 μm (Fig. 3c):** Shows good transaxial resolution but appears dispersed in 3D. Total variation minimized dispersion, but failed to resolve the source, yielding a low peak grayscale amplitude of 0.49.

Deviation decreases → improves data fidelity

Map-TV Iteration	β	Deviation	Image min/mean/max
1	1.91034	1162940	{0., 0.0182435 , 0.371918}
50	0.99332	481231	{0., 0.00830965 , 23.3693}
100	0.86874	426549	{0., 0.00733767 , 23.5103 }
150	0.85616	420927	{0., 0.0072 3617 , 23.4975 }
200	0.85478	420312	{0., 0.00722 503 , 23.5115}

Table 1. Iteration updates for Fig. 3a show the algorithm maximizes the objective function as both the data fidelity term given by the deviation and the TV prior strength β decreases.

TV prior β decreases → improves balance between data and model

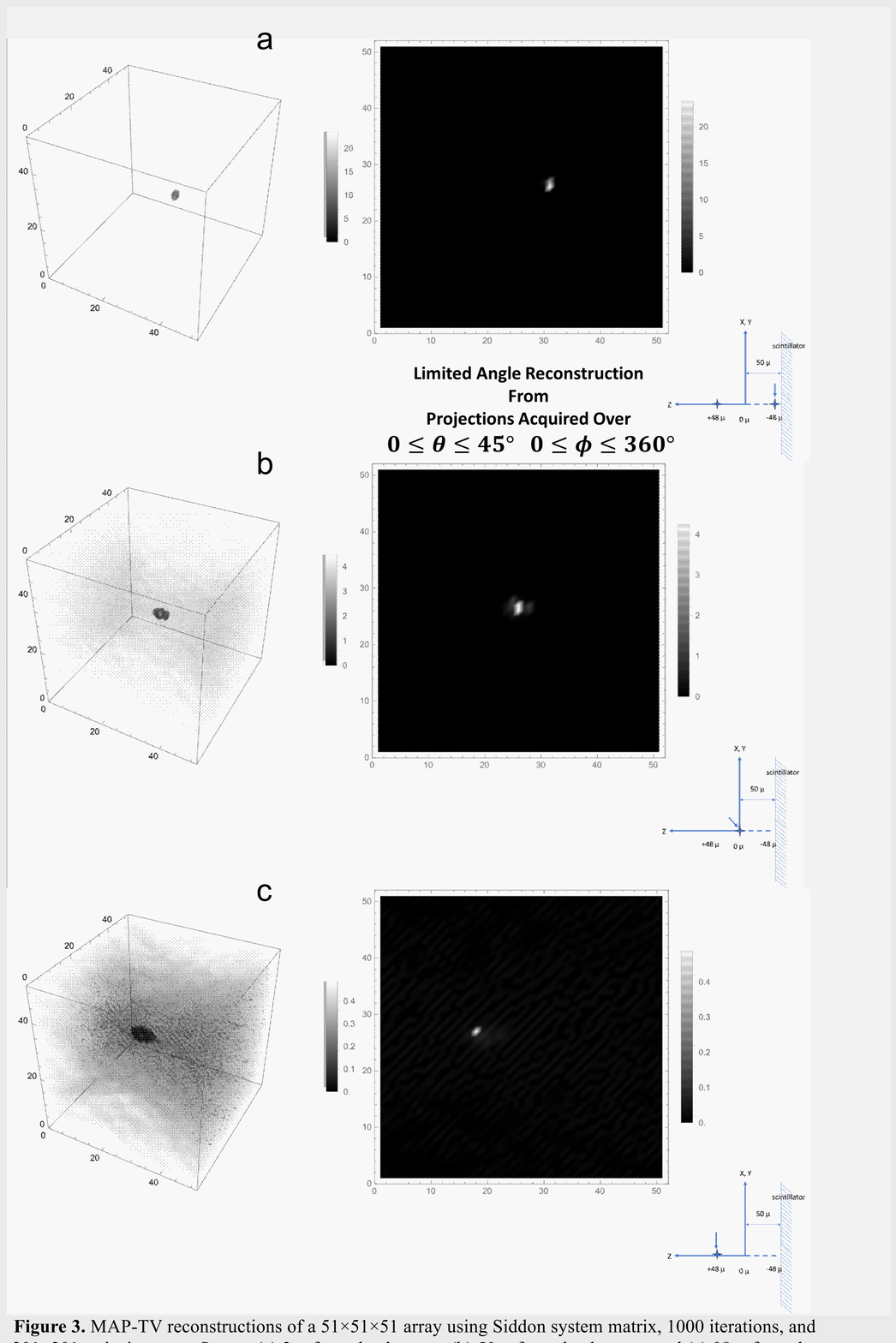


Figure 3. MAP-TV reconstructions of a 51×51×51 array using Siddon system matrix, 1000 iterations, and 201×201-unit sinogram: Source (a) 2 μ from the detector, (b) 50 μ from the detector, and (c) 98 μ from the detector. A shift in z of ≈ 7.5 voxels give a scale of ≈ 6.4 μ,

Acknowledgment. This work was supported by the National Institutes of Health under grant R21EB035205. We used ChatGPT (OpenAI) to assist with generating Mathematica code (Wolfram Research, Champaign, Illinois) for implementing the reconstruction algorithms.

References

[1] V Y Panin, G L Zeng, G T Gullberg. Total variation regulated EM algorithm. IEEE Trans on Nucl Sci 46:2202-2210, 1999.
[2] Y Sung. Snapshot three-dimensional absorption imaging of microscopic specimens Physic Review Applied. 15(6):064065, 2021.
[3] M Abdalah, D Mitra, S Chen, R Boutchko, G T Gullberg. Optimization of regularization parameter in a reconstruction algorithm. In: 2013 IEEE NSS/MIC Record, Oct. 27 – Nov. 2, 2013, Seoul, Korea.
[4] R H Huesman, G T Gullberg, W L Greenberg, T F Budinger. Users Manual - Donner Algorithms for Reconstruction Tomography. Lawrence Berkeley Laboratory Publication PUB-214, 1977.
[5] R L Siddon. Fast calculation of the exact radiological path for a three-dimensional CT array. Med Phys 12:252-255, 1985.
[6] M Bertero, P Boccacci, G Talenti, R Zanella, L Zanni. A discrepancy principle for Poisson data, Inverse Problems, 26(10), 105004, 2010.

In[ ]:=

```
$HistoryLength = 0;     (*avoid hidden copies in Out[]*)
Unprotect[Out]; Clear[Out]; Protect[Out];
Print["Nsamples = ", Nsamples = 22000];(*Print["Nsamples = ",Nsamples=24931];*)
Print["Nunits = ", Nunits = 201];
Print["Nprjsize = ", Nprjsize = 51];
Print["n = ", n = 51];
Print["Nimage = ", Nimage = 51];
suv = 1.0
L = 2.0 n;
Print["Noffset = (Nimage-n)/2 = ", Noffset = (Nimage - n) / 2];
Nstart = 1;
Nfinish = Nstart + Nsamples - 1;

Unprotect[First];
ClearAll[First];
Protect[First];

Unprotect[PositionLargest];
ClearAll[PositionLargest];
Protect[PositionLargest];

ux =
  Import["Desktop/Yongjin/PSF/Params_Zpos1.xls", {"Data", 1, Range[1, Nsamples], 4}];
uy =
  Import["Desktop/Yongjin/PSF/Params_Zpos1.xls", {"Data", 1, Range[1, Nsamples], 5}];
uz =
  Import["Desktop/Yongjin/PSF/Params_Zpos1.xls", {"Data", 1, Range[1, Nsamples], 6}];
px =
  Import["Desktop/Yongjin/PSF/Params_Zpos1.xls", {"Data", 1, Range[1, Nsamples], 1}];
py =
  Import["Desktop/Yongjin/PSF/Params_Zpos1.xls", {"Data", 1, Range[1, Nsamples], 2}];
pz =
  Import["Desktop/Yongjin/PSF/Params_Zpos1.xls", {"Data", 1, Range[1, Nsamples], 3}];
(********)
(*Calculate the angles for each sample*)
(********)
Print["theta"];
theta = Table[0., {II, 1, Nsamples}];
theta = Table[ArcSin[Sqrt[ux[[II]]^2 + uy[[II]]^2] /
   Sqrt[ux[[II]]^2 + uy[[II]]^2 + uz[[II]]^2]] / Degree, {II, 1, Nsamples}];
theta = Table[0., {II, 1, Nsamples}];
theta = Table[ArcSin[Sqrt[ux[[II]]^2 + uy[[II]]^2] /
   Sqrt[ux[[II]]^2 + uy[[II]]^2 + uz[[II]]^2]], {II, 1, Nsamples}];
Print["Xmaxtheta = ", Xmaxtheta = Max[theta] / Degree];
Print["Xmintheta = ", Xmintheta = Min[theta] / Degree];
```

```
Print["phi"];
Clear[x]
Clear[y]
phix[x_, y_] := Which[y > 0. && x > 0., ArcTan[y / x], y > 0. && x < 0., π - ArcTan[y / -x],
   y < 0. && x > 0., 2 π - ArcTan[-y / x], y < 0. && x < 0., ArcTan[-y / -x] + π,
   x == 0. && y > 0., π / 2, x == 0. && y < 0., 3 π / 2, y == 0. && x < 0., 0.,
   y == 0. && x > 0., 2 π, y == 0. && x == 0., 0., True, Indeterminate];
phi = Table[0., {II, 1, Nsamples}];
phi = Table[phix[ux[[II]], uy[[II]]], {II, 1, Nsamples}];
Print["Xmaxphi = ", Xmaxphi = Max[phi] / Degree];
Print["Xminphi = ", Xminphi = Min[phi] / Degree];
(********)
st = Sin /@ theta; ct = Cos /@ theta;
sp = Sin /@ phi; cp = Cos /@ phi;
ctcp = ct * cp; ctsp = ct * sp;
Print["uphysList"]
u = suv (px * sp - py * cp)
Print["vphysList"]
v = suv (px * ctcp + py * ctsp - pz * st)

Print["umax = ", umax = Max[u]];
Print["vmax = ", vmax = Max[v]];
Print["umin = ", umin = Min[u]];
Print["vmin = ", vmin = Min[v]];
Print["umaxabs = ", umaxabs = Max[Abs[umax], Abs[umin]]];
Print["vmaxabs = ", vmaxabs = Max[Abs[vmax], Abs[vmin]]];
Print["Xuvmax = ", Xuvmax = Max[umaxabs, vmaxabs] + 1];

(* ================================================*)
Print["-----------------------------------------"](*1. Form the Sinogram*)
Print["1. Form the Sinogram"];
(* ================================================*)

psum1 = Table[0., {II, 1, 9}];
psum2 = Table[0., {II, 1, 18}];
psum3 = Table[0., {II, 1, Nunits}];
psum4 = Table[0., {II, 1, Nunits}];
Clear[Sinogram];
Sinogram = Table[0., {Kx, 1, 9}, {Lx, 1, 18}, {Ix, 1, Nunits}, {Jx, 1, Nunits}];
For[i = 1, i ≤ Nsamples, i++,
 Kx = IntegerPart[(9. / 45.) * (theta[[i]] / Degree)] + 1;
  psum1[[Kx]] = psum1[[Kx]] + 1;
  Lx = IntegerPart[(18. / 360.) * (phi[[i]] / Degree) + 1.];
  psum2[[Lx]] = psum2[[Lx]] + 1;
   Ix = IntegerPart[((Nunits / 2) / Xuvmax) * u[[i]] + (Nunits / 2) + 1];
  psum3[[Ix]] = psum3[[Ix]] + 1;
  Jx = IntegerPart[((Nunits / 2) / Xuvmax) * v[[i]] + (Nunits / 2) + 1];
```

```
  psum4[[Jx]] = psum4[[Jx]] + 1;
  Sinogram[[Kx, Lx, Ix, Jx]] = Sinogram[[Kx, Lx, Ix, Jx]] + 1];
Print["theta, Kx=9"]; psum1
Print["phi, Lx=18"]; psum2
Print["u, Ix=", Nunits]
Print["psum3 for u = ", psum3];

Print["v, Jx=", Nunits]
Print["psum4 for v = ", psum4];

(*Print["Sinogram"]; MatrixForm[Sinogram]*)

Print["(Xmaxtheta-Xmintheta)"]; (Xmaxtheta - Xmintheta)
Print["(Xmaxphi-Xminphi)"]; (Xmaxphi - Xminphi)

(*Need projections for angles Kx and Lx}*)

Print["Extract Projections from the Sinogram"]
Print["Ntheta = ", Ntheta = 9];
Print["Kxx = ", Kxx = {1, 2, 3, 4, 5, 6, 7, 8, 9}];
Print["Nphi = ", Nphi = 18];
Print["Lxx = ", Lxx = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18}];
Clear[WXX];
Print["Nprjsize = ", Nprjsize];
Print["Noffset = ", Noffset = (Nunits - Nprjsize) / 2];
WXX = Table[0., {i, 1, Ntheta}, {j, 1, Nphi}, {II, 1, Nprjsize}, {JJ, 1, Nprjsize}];
For[i = 1, i ≤ Ntheta, i++, For[j = 1, j ≤ Nphi, j++,
   For[II = 1, II ≤ Nprjsize, II++, For[JJ = 1, JJ ≤ Nprjsize, JJ++,
 WXX[[i, j, II, JJ]] = Sinogram[[Kxx[[i]], Lxx[[j]], II + Noffset, JJ + Noffset]]]]];
(*Print["Projection WXX"];N[ MatrixForm[WXX]];
ListDensityPlot[Abs[WX], ColorFunction → GrayLevel, PlotRange → Automatic,
  ClippingStyle → None, PlotLegends → Automatic,
  PerformanceGoal → "Quality"]*)

(* =====BASIC PARAMETERS/HELPERS=====*)
n = 51;      (*volume size*)
nVoxels = n^3;

voxelIndex[i_, j_, k_, n_] := (i - 1) * n^2 + (j - 1) * n + k;

(* ================================================*)
Print["-----------------------------------------"]
(*2. Force all ray arrays to the same length*)
Print["2. Force all ray arrays to the same length"];
(* ================================================*)

(*Assume ux,uy,uz,px,py,pz are already imported*)
```

```
nRaysTot = Min @@ (Length /@ {ux, uy, uz, px, py, pz});
Print["nRaysTot = ", nRaysTot];

ux = ux[[1 ;; nRaysTot]];
uy = uy[[1 ;; nRaysTot]];
uz = uz[[1 ;; nRaysTot]];

px = px[[1 ;; nRaysTot]];
py = py[[1 ;; nRaysTot]];
pz = pz[[1 ;; nRaysTot]];

(* ================================================*)
Print["-----------------------------------------"]
(*3. Compute θ and φ from the same rays*)
Print["3. Compute θ and φ from the same rays"];
(* ================================================*)

theta = ArcCos[uz];     (*0..π,same length as uz*)
phi = ArcTan[ux, uy];   (*-π..π,same length as ux,uy*)

thetaDeg = N[theta * 180. / Pi];
phiDeg = N[Mod[phi * 180. / Pi, 360.]];

Print["Length[theta]  = ", Length[theta]];
Print["Length[phi]    = ", Length[phi]];
Print["Length[thetaDeg]= ", Length[thetaDeg]];
Print["Length[phiDeg]  = ", Length[phiDeg]];

(* ================================================*)
Print["-----------------------------------------"]
(*4. Build bins from those same arrays*)
Print["4. Build bins from those same arrays"];
(* ================================================*)
Ntheta = 9;
Nphi = 18;

thetaBin = Clip[Floor[thetaDeg / 5] + 1, {1, Ntheta}];
phiBin = Clip[Floor[phiDeg / 20] + 1, {1, Nphi}];

Print["Length[thetaBin] = ", Length[thetaBin]];
Print["Length[phiBin]   = ", Length[phiBin]];

(* ================================================*)
Print["-----------------------------------------"](*5. Build the angle list*)
Print["5. Build the angle list"];
(* ================================================*)
```

```
(*----Replace this whole block in your notebook----*)
(*θ-bin centers:use indices from thetaBin*)

thetaCenters = Table[Module[{idx, vals}, idx = Flatten@Position[thetaBin, i];
    (*all rays in bin i*)
    If[idx === {}, (*fallback if bin is empty*)(i - 0.5) * 5 Degree, vals = theta[[idx]];
     Mean[vals]]], {i, 1, Ntheta}];

(*φ-bin centers:use indices from phiBin*)
phiCenters = Table[Module[{idx, vals}, idx = Flatten@Position[phiBin, j];
    (*all rays in bin j*)If[idx === {}, (j - 0.5) * 20 Degree, vals = phi[[idx]];
     Mean[vals]]], {j, 1, Nphi}];


angles =
  Flatten[Table[{thetaCenters[[i]], phiCenters[[j]]}, {i, 1, Ntheta}, {j, 1, Nphi}], 1];

Print["Dimensions[angles] = ", Dimensions[angles]];
Print["First 3 angle pairs: ", angles[[1 ;; 3]]];

{Length[theta], Length[thetaBin], Length[phi], Length[phiBin]}

(* ====SIDDON CORE=====*)
ClearAll[axisT];

axisT[p0_?NumericQ, s_?NumericQ, min_?NumericQ, max_?NumericQ] := Module[{t1, t2},
   If[Abs[s] < 10.^-15, If[min ≤ p0 ≤ max, {-Infinity, +Infinity}, (*inside slab*)
     {+Infinity, -Infinity}  (*no intersection*)], t1 = (min - p0) / s;
    t2 = (max - p0) / s;
    {Min[t1, t2], Max[t1, t2]}]];

ClearAll[SiddonRay3D];

SiddonRay3D[p0_List, d_List, n_Integer?Positive] :=
  Module[{xMin = 0., xMax = n, yMin = 0., yMax = n, zMin = 0., zMax = n, dir, sx, sy,
    sz, tx, ty, tz, tEnter, tExit, t, pos, i, j, k, stepX, stepY, stepZ, tNextX,
    tNextY, tNextZ, dtx, dty, dtz, segs = {}, len, eps = 10.^-12}, dir = N[d];
   dir = dir / Max[Norm[dir], eps];
   {sx, sy, sz} = dir;
   tx = axisT[p0[[1]], sx, xMin, xMax];
   ty = axisT[p0[[2]], sy, yMin, yMax];
   tz = axisT[p0[[3]], sz, zMin, zMax];
   tEnter = Max[tx[[1]], ty[[1]], tz[[1]]];
   tExit = Min[tx[[2]], ty[[2]], tz[[2]]];
   If[tEnter ≥ tExit, Return[{}]];
   t = tEnter + eps;
   pos = p0 + t * dir;
```

```
    i = Clip[Floor[pos[[1]]] + 1, {1, n}];
    j = Clip[Floor[pos[[2]]] + 1, {1, n}];
    k = Clip[Floor[pos[[3]]] + 1, {1, n}];
    stepX = If[sx > 0, 1, If[sx < 0, -1, 0]];
    stepY = If[sy > 0, 1, If[sy < 0, -1, 0]];
    stepZ = If[sz > 0, 1, If[sz < 0, -1, 0]];
    If[stepX ≠ 0, tNextX = t + ((If[stepX > 0, i, i - 1] - pos[[1]]) / sx);
     dtx = 1. / Abs[sx], tNextX = +Infinity; dtx = +Infinity];
    If[stepY ≠ 0, tNextY = t + ((If[stepY > 0, j, j - 1] - pos[[2]]) / sy);
     dty = 1. / Abs[sy], tNextY = +Infinity; dty = +Infinity];
    If[stepZ ≠ 0, tNextZ = t + ((If[stepZ > 0, k, k - 1] - pos[[3]]) / sz);
     dtz = 1. / Abs[sz], tNextZ = +Infinity; dtz = +Infinity];
    While[t < tExit && 1 ≤ i ≤ n && 1 ≤ j ≤ n && 1 ≤ k ≤ n,
     If[tNextX ≤ tNextY && tNextX ≤ tNextZ, len = Min[tNextX, tExit] - t;
       If[len > 0, AppendTo[segs, {{i, j, k}, len}]];
       t = tNextX;
       tNextX += dtx;
       i += stepX, If[tNextY ≤ tNextZ, len = Min[tNextY, tExit] - t;
        If[len > 0, AppendTo[segs, {{i, j, k}, len}]];
        t = tNextY; tNextY += dty; j += stepY, len = Min[tNextZ, tExit] - t;
        If[len > 0, AppendTo[segs, {{i, j, k}, len}]];
        t = tNextZ; tNextZ += dtz; k += stepZ]];];
    segs];

(* ====BUILD SIDDON SYSTEM MATRIX====*)
ClearAll[makeDetectorBasis];

makeDetectorBasis[dir_?VectorQ] := Module[{d, ref, uHat, vHat}, d = N[dir / Norm[dir]];
    ref = If[Abs[d.{0., 0., 1.}] < 0.9, {0., 0., 1.}, {0., 1., 0.}];
    uHat = Cross[ref, d];
    uHat = uHat / Norm[uHat];
    vHat = Cross[d, uHat];
    vHat = vHat / Norm[vHat];
    {uHat, vHat}];

ClearAll[buildSiddonRulesOneAngle];

buildSiddonRulesOneAngle[theta_, phi_, n_Integer, rayOffset_Integer] :=
  Module[{dir, uHat, vHat, center, t0, rayIndex = 1, p0, segs, i, j, k, len, entries, u, v},
    (*Ray direction*)dir = N[{Sin[theta] Cos[phi], Sin[theta] Sin[phi], Cos[theta]}];
    dir = dir / Norm[dir];
    (*Smooth spherical detector basis*)uHat = N[{Sin[phi], -Cos[phi], 0.}];
    uHat = uHat / Norm[uHat];
    vHat = N[{-Cos[theta] Cos[phi], -Cos[theta] Sin[phi], Sin[theta]}];
    vHat = vHat / Norm[vHat];
    (*Center of cube in voxel coordinates*)center = ConstantArray[(n + 1) / 2., 3];
    (*Start detector plane upstream*)t0 = Sqrt[3.] n;
```

```
    entries = Reap[Do[Do[u = ix - (n + 1) / 2.;
          v = iy - (n + 1) / 2.;
          p0 = center + u * uHat + v * vHat - t0 * dir;
          segs = SiddonRay3D[p0, dir, n];
          Scan[Function[s, {i, j, k} = s[[1]];
            len = s[[2]];
            If[len > 0.,
             Sow[{rayOffset + rayIndex, voxelIndex[i, j, k, n]} → len]];], segs];
          rayIndex++, {iy, 1, n}], {ix, 1, n}]][[2, 1]];
    entries];

(* ================================================*)
Print["-----------------------------------------"]
(*6. Build FtotalSiddon from all angles*)
Print["6. Build FtotalSiddon from all angles"];
(* ================================================*)
(*Assumes:-n       already defined (51)-Ntheta,
Nphi defined-angles is a list of length nAngles=
 Ntheta*Nphi with angles[[k]]={theta_k,phi_k}*)

nAngles = Length[angles];  (*should be 162*)
nRaysPerAng = n^2;
nRaysTot = nAngles * nRaysPerAng;
nVoxels = n^3;

Print["Expecting FtotalSiddon dims: {", nRaysTot, ", ", nVoxels, "}"];

rulesByAngle = Table[With[{th = angles[[kk, 1]], ph = angles[[kk, 2]], offset = (kk - 1) * n^2},
     buildSiddonRulesOneAngle[th, ph, n, offset]], {kk, 1, Length[angles]}];

allSiddonRules = Join @@ rulesByAngle;
FtotalSiddon = SparseArray[allSiddonRules, {Length[angles] * n^2, n^3}];

(*FtotalSiddon=SparseArray[allSiddonRules,{nRaysTot,nVoxels}];*)

Print["FtotalSiddon dims = ", Dimensions[FtotalSiddon]];
Print["Nonzeros in FtotalSiddon = ", Length[ArrayRules[FtotalSiddon]]];

angleTotals =
  Table[Total[FtotalSiddon[[(kk - 1) * n^2 + 1 ;; kk * n^2, All]]["NonzeroValues"]],
   {kk, 1, Length[angles]}];

ListPlot[angleTotals, PlotRange → All,
 Frame → True, PlotLabel → "Total Siddon weight per angle"]
(* ================================================*)
Print["-----------------------------------------"]
```

```
(*7. Replace WXX with actual data if needed*)
Print["7. Replace WXX with actual data if needed"];
(* ================================================*)

Print["WXX size: ", Dimensions[WXX]];
Print["Projections WXX = "]; WXX;
Ptotal = Flatten[WXX];
Print["Projections Ptotal=Flatten[WXX]"]; Ptotal;
Print["Length[Ptotal] = ", Length[Ptotal]];
Dimensions[FtotalSiddon][[1]] === Length[Ptotal]
Print["Rows(FtotalSiddon)   = ", Dimensions[FtotalSiddon][[1]]];
Print["Sum[Ptotal]    = ", N@Total[Ptotal]];
Print["Nonzeros       = ", Count[Ptotal, x_ /; x ≠ 0.]];

(*Just before starting MAP EM TV*)
Print["Using real data?  Max[Ptotal] = ", Max[Ptotal]];
Print["Some entries of Ptotal: ", Take[Ptotal, 10]];

(* ================================================*)
Print["-----------------------------------------"]
(*8. Reconstruct Volume Using Iterative MLEM*)
Print["8. Reconstruct Volume Using Iterative MLEM"];
(* ================================================*)

F01 = Unitize[FtotalSiddon]; (*Siddon Weighting*)
Ftotal = F01
FT = Transpose[Ftotal];
nRaysTot = Dimensions[Ftotal][[1]];
nVoxels = Dimensions[Ftotal][[2]]; (*should equal n^3*)

onesProj = ConstantArray[1., nRaysTot];

FT1 = N[FT.onesProj];

maxIterMLEM = 40;
showEvery = 10;

xMLEM = ConstantArray[1., nVoxels]; (*initial guess*)

Do[(*forward projection*)lambda = N[Ftotal.xMLEM] + 1.*^-12;
  (*ratio y/(F x)*)rat = Ptotal / lambda;
  (*backprojection*)back = N[FT.rat];
  (*EM update*)xMLEM = xMLEM * (back / (FT1 + 1.*^-12));
  xMLEM = Clip[xMLEM, {0., Infinity}];
  If[Mod[it, showEvery] == 0, Print["MLEM iter ", it, " : min/mean/max = ",
     N@{Min[xMLEM], Mean[xMLEM], Max[xMLEM]}];];, {it, 1, maxIterMLEM}];
```

```
(*Keep Ftotal and Ptotal as they are*)
FT = Transpose[Ftotal];
nProjs = Length[Ptotal];
onesProj = ConstantArray[1., nProjs];
FT1 = N[FT.onesProj];

xMAP = xMLEM;  (*start from converged MLEM solution*)

(* ================================================*)
Print["-----------------------------------------"]
(*9. Visualize MLEM reconstruction*)
Print["9. Visualize MLEM reconstruction"];
(* ================================================*)

volMLEM = ArrayReshape[xMLEM, {n, n, n}];

(*Find voxel with maximum value in volMLEM*)
posMax = First@Ordering[Flatten[volMLEM], -1];

iMax = Quotient[posMax - 1, n^2] + 1;
r1 = Mod[posMax - 1, n^2];
jMax = Quotient[r1, n] + 1;
kMax = Mod[r1, n] + 1;

Print["Max voxel at (i,j,k) = (", iMax, ",", jMax, ",", kMax, ")"];

mid = Ceiling[n / 2];

sliceXY = volMLEM[[All, All, mid]];   (*z=mid*)
sliceXZ = volMLEM[[All, mid, All]];   (*y=mid*)
sliceYZ = volMLEM[[mid, All, All]];   (*x=mid*)

Print["Max of reconstructed volume = ", Max[volMLEM] // N];

Print["Central XY slice z = ", mid, ":"];
ArrayPlot[sliceXY, PlotRange → All, Frame → True,
 PlotLabel → Row[{"XY slice (z = ", mid, ")"}]]

Print["Central XZ slice y = ", mid, ":"];
ArrayPlot[sliceXZ, PlotRange → All, Frame → True,
 PlotLabel → Row[{"XZ slice (y = ", mid, ")"}]]

Print["Central YZ slice x = ", mid, ":"];
ArrayPlot[sliceYZ, PlotRange → All, Frame → True,
 PlotLabel → Row[{"YZ slice (x = ", mid, ")"}]]

Clear[WX]
```

```
Print["Nimage = ", Nimage];
Print["Noffset = ", Noffset = (Nimage - n) / 2];
WX = Table[0., {II, 1, Nimage}, {JJ, 1, Nimage}];
For[II = 1, II ≤ n, II++, For[JJ = 1, JJ ≤ n, JJ++,
  WX[[II + Noffset, JJ + Noffset]] = volMLEM[[II, 26, JJ]]]]
Print["volumeMLEMmax = ", volMLEMmax = Max[volMLEM]];

Print["WX = "]; N[WX];
Print["WXmax = ", WXmax = Max[WX]];
ListDensityPlot[WX, ColorFunction → GrayLevel, PlotRange → Full,
 ClippingStyle → None, PlotLegends → Automatic,
 PerformanceGoal → "Quality"]

WXX = Flatten[WX];
Print["IJ = ", IJ = PositionLargest[WXX]];
Clear[WXX]
WXX = Table[0., {II, 1, Nimage}, {JJ, 1, Nimage}, {KK, 1, Nimage}];
WXX = Table[ volMLEM[[II, JJ, KK]], {II, 1, Nimage}, {JJ, 1, Nimage}, {KK, 1, Nimage}];
ListDensityPlot3D[Abs[WXX], ColorFunction → GrayLevel,
 PlotRange → Automatic, PlotLegends → Automatic, PerformanceGoal → "Quality"]
(*-------------------------------------------------*)

(*---Smoothed isotropic TV gradient for a 3D volume---*)

(*---Utilities---*)
ClearAll[TVGrad3D];

TVGrad3D[vol_?ArrayQ, eps_ : 1.*^-6] :=
  Module[{v, nx, ny, nz, zX, zY, zZ, dx, dy, dz, mag, px, py, pz, divx, divy, divz},
   v = Developer`ToPackedArray@N@vol;
   {nx, ny, nz} = Dimensions[v];
   (*Zero slices for Neumann BC (forward diffs=0 at high edge)*)
   zX = ConstantArray[0., {1, ny, nz}];
   zY = ConstantArray[0., {nx, 1, nz}];
   zZ = ConstantArray[0., {nx, ny, 1}];
   (*Forward differences*)dx = Join[v[[2 ;;, All, All]] - v[[ ;; -2, All, All]], zX, 1];
   dy = Join[v[[All, 2 ;;, All]] - v[[All,  ;; -2, All]], zY, 2];
   dz = Join[v[[All, All, 2 ;;]] - v[[All, All,  ;; -2]], zZ, 3];
   mag = Sqrt[dx^2 + dy^2 + dz^2 + eps^2];
   px = dx / mag; py = dy / mag; pz = dz / mag;
   (*Backward divergence (Neumann BC:zero-flux at low edge)*)
   divx = Join[px[[{1}, All, All]], px[[2 ;;, All, All]] - px[[ ;; -2, All, All]], 1];
   divy = Join[py[[All, {1}, All]], py[[All, 2 ;;, All]] - py[[All,  ;; -2, All]], 2];
   divz = Join[pz[[All, All, {1}]], pz[[All, All, 2 ;;]] - pz[[All, All,  ;; -2]], 3];
   (*This returns ∇TV(u)=-div(grad/|grad|)*)
   Developer`ToPackedArray@N@(- (divx + divy + divz))];
```

```
(* ================================================*)
Print["-----------------------------------------"]
(*10. Reconstruct Volume Using Iterative EM+TV*)
Print["10. Reconstruct Volume Using Iterative EM+TV"];
(* ================================================*)

nVoxels = n^3;
nProjs = Length[Ptotal];

xMLEM = ConstantArray[1., nVoxels]; (*initial guess*)

FT = Transpose[Ftotal];
onesProj = ConstantArray[1., nProjs];

(*Precompute normalization W=F^T 1*)
FT1 = FT.onesProj // N // Developer`ToPackedArray;

(*Rules for cleaning bad values*)
badRules = {Indeterminate → 0., ComplexInfinity → 0., DirectedInfinity[_] → 0.,
   Infinity → 0., -Infinity → 0., Null → 0., Missing[__] → 0., $Failed → 0.};
badD = Dispatch[badRules];

ClearAll[PoissonDeviance];
PoissonDeviance[y_, mu_, eps_ : 10^-12] := Module[{yy, mm}, yy = N[y];
   mm = N[mu] + eps;
   2 Total@Table[If[yy[[i]] == 0., mm[[i]],
       yy[[i]] * Log[yy[[i]] / mm[[i]]] - yy[[i]] + mm[[i]]], {i, Length[yy]}]];

Print["Starting MAP EM TV iterations..."];

nVoxels = n^3;
nProjs = Length[Ptotal];

xMLEM = ConstantArray[1., nVoxels]; (*initial guess*)

FT = Transpose[Ftotal];
onesProj = ConstantArray[1., nProjs];

(*Precompute normalization W=F^T 1*)
FT1 = FT.onesProj // N // Developer`ToPackedArray;

(*Rules for cleaning bad values*)
badRules = {Indeterminate → 0., ComplexInfinity → 0., DirectedInfinity[_] → 0.,
   Infinity → 0., -Infinity → 0., Null → 0., Missing[__] → 0., $Failed → 0.};
badD = Dispatch[badRules];

(*Parameters*)
```

```
epsProj = 1.*^-12;
epsTV = 1.*^-6;
betaTV = 2.5;
Print["betaTV = ", betaTV]
tauTV = 0.10;
Print["tauTV = ", tauTV]
cTV = 51.^3;
Print["cTV = ", cTV]
epsilonBeta = 10.^-7;
Print["epsilonBeta = ", epsilonBeta]
betaMin = 0.0;
Print["betaMin = ", betaMin]
betaMax = 1.5;
Print["betaMax = ", betaMax]
gammaBeta = 0.25;     (*relaxation for adaptive beta*)
Print["gammaBeta = ", gammaBeta]
adaptiveBetaQ = True;   (*set False for fixed beta*)
Print["adaptiveBetaQ = ", adaptiveBetaQ]
nIter = 200;
Print["nIter = ", nIter]
useTVBasedBetaQ = False;
Print["useTVBasedBetaQ = ", useTVBasedBetaQ]
targetDeviance = nProjs;  (*practical first choice*)

Print["betaTV initial = ", betaTV];
Print["tauTV = ", tauTV];
Print["nIter = ", nIter];
Print["adaptiveBetaQ = ", adaptiveBetaQ];
Print["targetDeviance = ", targetDeviance];

(*Ensure numeric packed inputs*)
xMLEM = xMLEM // N // Developer`ToPackedArray;
Ptotal = Ptotal // N // Developer`ToPackedArray;

Print["Starting MAP EM TV iterations..."];

Do[Module[{projEstimate, ratio, correction,
    xEM, volEM, tTV, gTV, scale, volNext, mu, dev, tvVal, betaNew},
   (*Forward projection*)projEstimate = N[Ftotal.xMLEM];
   projEstimate = Developer`ToPackedArray[projEstimate /. badD];
   (*Ratio y/(Fx+eps)*)ratio = N[Ptotal / (projEstimate + epsProj)];
   ratio = Developer`ToPackedArray[ratio /. badD];
   (*Backprojection*)correction = N[FT.ratio];
   correction = Developer`ToPackedArray[correction /. badD];
   (*EM multiplicative update:x<-x*(F^T(y/(Fx)))/(F^T 1)*)
   xEM = xMLEM * (correction / (FT1 + epsProj));
   xEM = N[xEM];
```

```
    xEM = xEM /. badD;
    xEM = Clip[xEM, {0., Infinity}];
    xEM = Developer`ToPackedArray[xEM];
    (*Sanity check before reshape*)
    If[! (VectorQ[xEM, NumericQ] && Length[xEM] == nVoxels),
     Print["BAD xEM at iter ", iter, ": Head=", Head[xEM], " Len=", Length[xEM]];
     Break[];];
    volEM = Developer`ToPackedArray@ArrayReshape[xEM, {n, n, n}];
    (*TV gradient*)
    {tTV, gTV} = TimeConstrained[AbsoluteTiming[TVGrad3D[volEM, epsTV]],
      60, {"timeout", ConstantArray[0., {n, n, n}]}];
    If[tTV === "timeout", Print["TVGrad3D timed out after 60 seconds. Stopping."];
     Break[];];
    gTV = Developer`ToPackedArray@N[gTV /. badD];
    (*Robust step normalization*)scale = Max[Abs[gTV]] + 10.^-12;
    volNext = volEM - (tauTV * betaTV / scale) * gTV;
    volNext = Clip[volNext, {0., Infinity}];
    xMLEM = Developer`ToPackedArray@Flatten[volNext];
    (*Diagnostics*)mu = N[Ftotal.xMLEM];
    mu = Developer`ToPackedArray[mu /. badD];
    dev = PoissonDeviance[Ptotal, mu, epsProj];
    TVValue3D[vol_?ArrayQ, eps_:1.*^-6] := Module[
      {v, nx, ny, nz, zX, zY, zZ, dx, dy, dz, mag}, v = Developer`ToPackedArray@N@vol;
      {nx, ny, nz} = Dimensions[v];
      (*Zero slices for Neumann BC (forward diffs=0 at high edge)*)
      zX = ConstantArray[0., {1, ny, nz}];
      zY = ConstantArray[0., {nx, 1, nz}];
      zZ = ConstantArray[0., {nx, ny, 1}];
      (*Forward differences*)dx = Join[v[[2 ;;, All, All]] - v[[ ;; -2, All, All]], zX, 1];
      dy = Join[v[[All, 2 ;;, All]] - v[[All,  ;; -2, All]], zY, 2];
      dz = Join[v[[All, All, 2 ;;]] - v[[All, All,  ;; -2]], zZ, 3];
      mag = Sqrt[dx^2 + dy^2 + dz^2 + eps^2];
      N@Total[Flatten[mag]]];
    (*Scalar TV value*)
    tvVal = N[TVValue3D[volNext, epsTV]];
    tvVal = tvVal /. badD;
    If[adaptiveBetaQ, If[useTVBasedBetaQ, betaNew = cTV / (tvVal + epsilonBeta);
        betaNew = Clip[betaNew, {betaMin, betaMax}];
        betaTV = (1.0 - gammaBeta) * betaTV + gammaBeta * betaNew;,
        betaNew = betaTV * (targetDeviance / dev)^gammaBeta;
        betaNew = Clip[betaNew, {betaMin, betaMax}];
        betaTV = 0.8 betaTV + 0.2 betaNew;];];
    If[Mod[iter, 50] == 0 || iter == 1, Print["MAP-TV iter ", iter, " | betaTV = ",
        N[betaTV], " | dev = ", N[dev], " | tvVal = ", N[tvVal], " | min/mean/max = ",
        N@{Min[xMLEM], Mean[xMLEM], Max[xMLEM]}];];], {iter, 1, nIter}];

Print["MAP EM TV reconstruction complete."];
```

```
volumeMAPTV = ArrayReshape[xMLEM, {n, n, n}];
Print["volumeMAPTVmax ", Max[volumeMAPTV] // N];

(* ================================================*)
Print["-----------------------------------------"]
(*11. Visualize MAP TV reconstruction*)
Print["11. Visualize MAP TV reconstruction"];
(* ================================================*)

(* Reshape to Volume*)
finished = True;

Do[(*...*)If[badCondition, finished = False;
    Break[];];, {iter, 1, nIter}];

If[finished, volumeMAPTV = ArrayReshape[xMLEM, {n, n, n}],
  Print["Stopped early; not reshaping output."]];

Print["volumeMAPTVmax = ", volumeMAPTVmax = Max[volumeMAPTV]];

(* Visualize Results*)
Clear [zSlice]; Clear[xSlice]; Clear[ySlice];
zSlice = volumeMAPTV[[All, All, Ceiling[(n + 1) / 2]]]; (*XY plane at center Z*)
xSlice = volumeMAPTV[[Ceiling[n / 2], All, All]]; (*YZ plane at center X*)
ySlice = volumeMAPTV[[All, Ceiling[n / 2], All]]; (*XZ plane at center Y*)

Print["zSlicemax = Max[zSlice]", zSlicemax = Max[zSlice]];
ListDensityPlot[zSlice, PlotLabel → "XY Slice (Z = middle)",
 ColorFunction → GrayLevel, PlotRange → Full, Frame → True,
 ClippingStyle → None, PlotLegends → Automatic, PerformanceGoal → "Quality"]
Print["xSlicemax = Max[xSlice]", xSlicemax = Max[xSlice]];
ListDensityPlot[xSlice, PlotLabel → "YZ Slice (X = middle)",
 ColorFunction → GrayLevel, PlotRange → Full, Frame → True,
 ClippingStyle → None, PlotLegends → Automatic, PerformanceGoal → "Quality"]
Print["ySlicemax = Max[ySlice]", ySlicemax = Max[ySlice]];
ListDensityPlot[ySlice, PlotLabel → "XZ Slice (Y = middle)",
 ColorFunction → GrayLevel, PlotRange → Full, Frame → True,
 ClippingStyle → None, PlotLegends → Automatic, PerformanceGoal → "Quality"]

Clear[WX]
Print["Nimage = ", Nimage];
Print["Noffset = ", Noffset = (Nimage - n) / 2];
WX = Table[0., {II, 1, Nimage}, {JJ, 1, Nimage}];
For[II = 1, II ≤ n, II++, For[JJ = 1, JJ ≤ n, JJ++,
  WX[[II + Noffset, JJ + Noffset]] = volumeMAPTV[[II, JJ, 38]]]]
Print["volumeMAPTVmax = ", volumeMAPTVmax = Max[volumeMAPTV]];
Print["Position[volumeMAPTV] = ", pos = Position[volumeMAPTV, Max[volumeMAPTV]]];
```

```
Print["WX = "]; N[WX];
Print["WXmax = ", WXmax = Max[WX]];
Print["volumeMAPTV[[II,JJ,38]] = "];
ListDensityPlot[WX, ColorFunction → GrayLevel, PlotRange → Full,
 ClippingStyle → None, PlotLegends → Automatic,
 PerformanceGoal → "Quality"]

Clear[WX]
Print["Nimage = ", Nimage];
Print["Noffset = ", Noffset = (Nimage – n) / 2];
WX = Table[0., {II, 1, Nimage}, {JJ, 1, Nimage}];
For[II = 1, II ≤ n, II++, For[JJ = 1, JJ ≤ n, JJ++,
  WX[[II + Noffset, JJ + Noffset]] = volumeMAPTV[[II, 26, JJ]]]]
Print["volumeMAPTVmax = ", volumeMAPTVmax = Max[volumeMAPTV]];

Print["WX = "]; N[WX];
Print["WXmax = ", WXmax = Max[WX]];
Print["volumeMAPTV[[II,26,JJ]] = "];
ListDensityPlot[WX, ColorFunction → GrayLevel, PlotRange → Full,
 ClippingStyle → None, PlotLegends → Automatic,
 PerformanceGoal → "Quality"]

Clear[WX]
Print["Nimage = ", Nimage];
Print["Noffset = ", Noffset = (Nimage – n) / 2];
WX = Table[0., {II, 1, Nimage}, {JJ, 1, Nimage}];
For[II = 1, II ≤ n, II++, For[JJ = 1, JJ ≤ n, JJ++,
  WX[[II + Noffset, JJ + Noffset]] = volumeMAPTV[[26, II, JJ]]]]
Print["volumeMAPTVmax = ", volumeMAPTVmax = Max[volumeMAPTV]];

Print["WX = "]; N[WX];
Print["WXmax = ", WXmax = Max[WX]];
Print["volumeMAPTV[[26,II,JJ]] = "];
ListDensityPlot[WX, ColorFunction → GrayLevel, PlotRange → Full,
 ClippingStyle → None, PlotLegends → Automatic,
 PerformanceGoal → "Quality"]

WXX = Flatten[WX];
Print["IJJ = ", IJJ = PositionLargest[WXX]];

Clear[WXX]
WXX = Table[0., {II, 1, Nimage}, {JJ, 1, Nimage}, {KK, 1, Nimage}];
WXX = Table[ volumeMAPTV[[II, JJ, KK]], {II, 1, Nimage}, {JJ, 1, Nimage}, {KK, 1, Nimage}];
ListDensityPlot3D[Abs[WXX], ColorFunction → GrayLevel,
 PlotRange → Automatic, PlotLegends → Automatic, PerformanceGoal → "Quality"]
```

```
Nsamples = 22 000
```

```
Nunits = 201

Nprjsize = 51

n = 51

Nimage = 51
```

Out[•]=

```
1.
```

```
Noffset = (Nimage-n)/2 = 0

theta

Xmaxtheta = 40.54

Xmintheta = 0.825246

phi

Xmaxphi = 359.918

Xminphi = 0.0614659

uphysList
```

Out[•]=

```
{-0.270001, -0.399119, 0.0422248, 0.370341, 0.0365433, 0.272161,
 -0.0502587, -2.63421, 0.100141, 0.071053, ··· 21980 ···, -0.0424437, 0.061411,
 0.0692119, -0.0340116, -0.369738, 0.0403015, -0.505957, -335.174, 0.0152061, 0.348169}
```

Full expression not available (original memory size: 176.2 kB)

```
vphysList
```

Out[•]=

```
{25.9389, 29.7763, 29.479, 31.22, 8.21742, 29.7146, 13.3831, 23.1862, 6.37715, 28.1869, 21.3786, ··· 21978 ···,
 28.6091, 11.9584, 27.3097, 28.9274, 29.9783, 22.1965, 21.3631, 27.7531, 16.9481, 24.2936, 31.3302}
```

Full expression not available (original memory size: 176.2 kB)

```
umax = 875.868

vmax = 460.434

umin = -811.18

vmin = -1017.42

umaxabs = 875.868

vmaxabs = 1017.42

Xuvmax = 1018.42

-----------------------------------------_

1. Form the Sinogram

theta, Kx=9
```

Out[•]=

```
{340., 890., 1880., 2220., 3190., 3630., 4330., 4940., 580.}
```

```
phi, Lx=18
```

Out[ ]=

{1330., 1250., 1350., 1270., 1100., 1380., 1110., 1020.,
 1090., 1230., 1170., 1260., 1450., 1220., 1300., 1260., 1170., 1040.}

u, Ix=201

psum3 for u = {0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 10., 0.,
  0., 0., 0., 10., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
  10., 0., 30., 0., 0., 0., 0., 0., 10., 0., 10., 10., 10., 10., 20., 10., 0., 20., 20., 40.,
  0., 20., 40., 10., 40., 30., 20., 30., 40., 20., 10., 20., 30., 30., 30., 40., 30., 30.,
  30., 40., 110., 60., 90., 70., 80., 170., 40., 150., 50., 40., 100., 110., 110., 230., 300.,
  17 510., 240., 150., 80., 100., 40., 100., 80., 110., 60., 80., 40., 60., 30., 20., 110.,
  40., 40., 30., 10., 10., 20., 30., 40., 40., 60., 30., 30., 30., 30., 30., 20., 10., 20., 0.,
  20., 0., 0., 10., 20., 10., 30., 10., 0., 10., 0., 0., 0., 0., 0., 0., 0., 10., 0., 10., 0.,
  10., 0., 20., 0., 0., 0., 0., 10., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 10., 0., 0., 0.,
  0., 10., 0., 0., 0., 0., 0., 0., 10., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}

v, Jx=201

psum4 for v = {10., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 10., 0., 0., 0., 0.,
  0., 0., 0., 0., 10., 0., 0., 0., 10., 0., 0., 0., 0., 10., 10., 0., 10., 0., 10., 0., 0., 0.,
  0., 0., 20., 10., 20., 0., 0., 10., 0., 10., 10., 10., 10., 0., 20., 30., 10., 40., 20., 10.,
  10., 10., 30., 50., 20., 50., 50., 60., 50., 30., 20., 30., 20., 40., 30., 70., 40., 60.,
  30., 90., 30., 100., 60., 40., 90., 50., 130., 30., 70., 100., 130., 100., 100., 50., 150.,
  140., 180., 580., 3530., 7460., 6700., 100., 50., 110., 50., 70., 0., 60., 10., 20., 40.,
  0., 40., 40., 50., 10., 20., 50., 20., 20., 20., 20., 40., 0., 20., 10., 30., 10., 0., 50.,
  10., 20., 10., 0., 20., 0., 0., 0., 0., 10., 20., 10., 20., 0., 0., 0., 0., 0., 0., 0., 0.,
  0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
  0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}

(Xmaxtheta-Xmintheta)

Out[ ]=

39.7148

(Xmaxphi-Xminphi)

Out[ ]=

359.856

Extract Projections from the Sinogram

Ntheta = 9

Kxx = {1, 2, 3, 4, 5, 6, 7, 8, 9}

Nphi = 18

Lxx = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18}

Nprjsize = 51

Noffset = 75

---

2. Force all ray arrays to the same length

nRaysTot = 22 000

---

3. Compute θ and φ from the same rays

Length[theta] = 22 000

Length[phi] = 22 000

Length[thetaDeg] = 22 000

Length[phiDeg] = 22 000

---

4. Build bins from those same arrays

Length[thetaBin] = 22 000

Length[phiBin] = 22 000

---

5. Build the angle list

Dimensions[angles] = {162, 2}

First 3 angle pairs: {{0.0606672, 0.163406}, {0.0606672, 0.528486}, {0.0606672, 0.867731}}

Out[•]=

{22 000, 22 000, 22 000, 22 000}

---

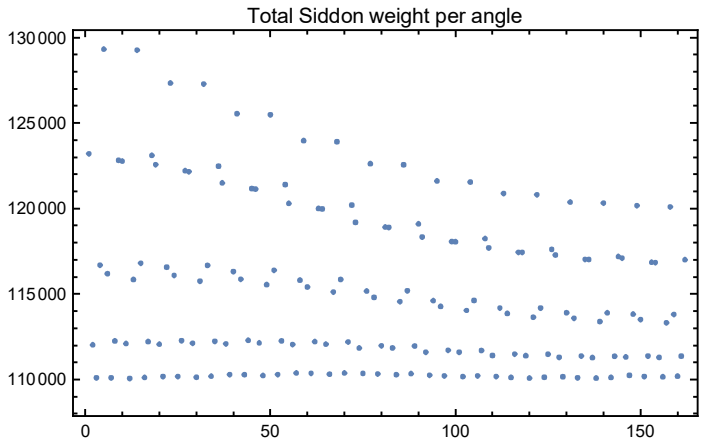
6. Build FtotalSiddon from all angles

Expecting FtotalSiddon dims: {421 362, 132 651}

FtotalSiddon dims = {421 362, 132 651}

Nonzeros in FtotalSiddon = 25 653 398

Out[ ]=



---

7. Replace WXX with actual data if needed

WXX size: {9, 18, 51, 51}

Projections WXX =

Projections Ptotal=Flatten[WXX]

Length[Ptotal] = 421362

Out[ ]=

True

Rows(FtotalSiddon) = 421362

Sum[Ptotal] = 20540.

Nonzeros = 529

Using real data? Max[Ptotal] = 290.

Some entries of Ptotal: {0., 0., 0., 0., 0., 0., 0., 0., 0., 0.}

---

8. Reconstruct Volume Using Iterative MLEM

Out[ ]=

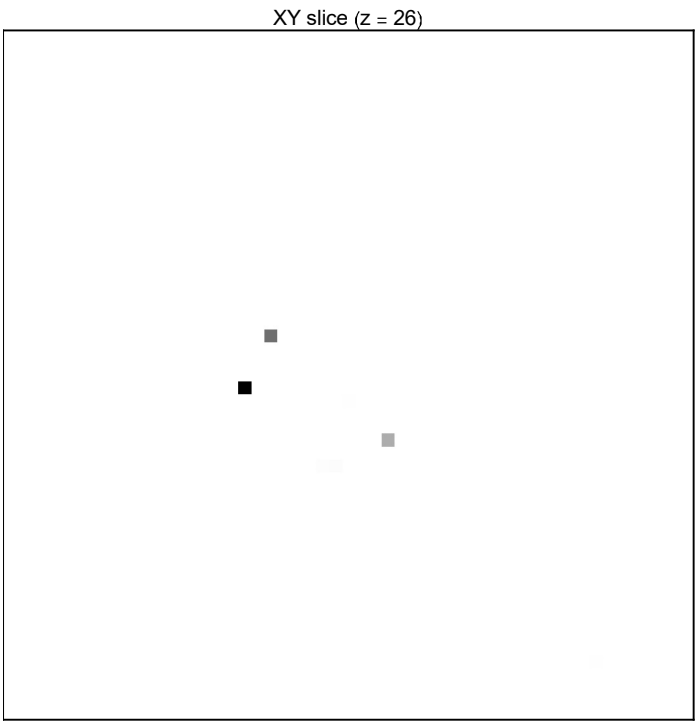
SparseArray[Specified elements: 25653397
Dimensions: {421362, 132651}]

Data not saved. Save now

```
MLEM iter 10 : min/mean/max = {0., 0.00074539, 16.5651}
MLEM iter 20 : min/mean/max = {0., 0.000745463, 18.4177}
MLEM iter 30 : min/mean/max = {0., 0.000745693, 18.9361}
MLEM iter 40 : min/mean/max = {0., 0.000745903, 19.1391}
```

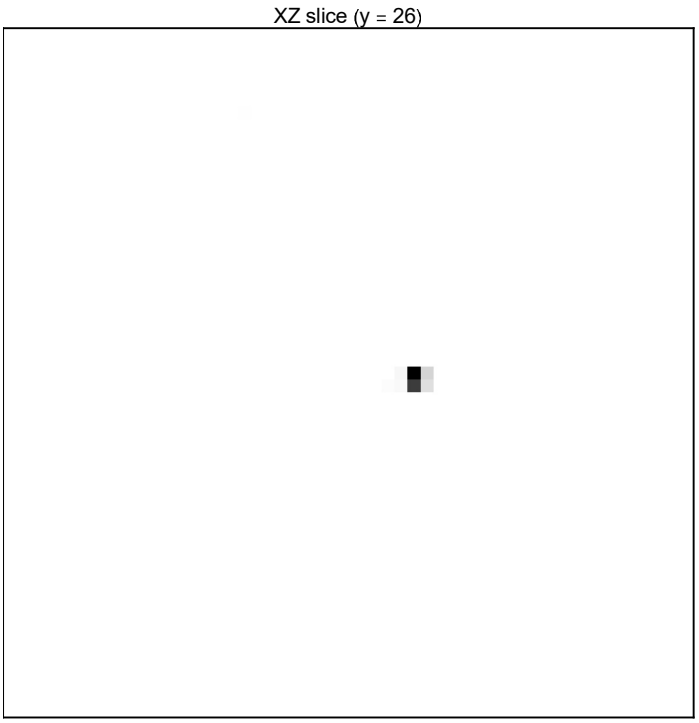
```
9. Visualize MLEM reconstruction
Max voxel at (i,j,k) = (26,26,31)
Max of reconstructed volume = 19.1391
Central XY slice z = 26:
```

Out[ ]=



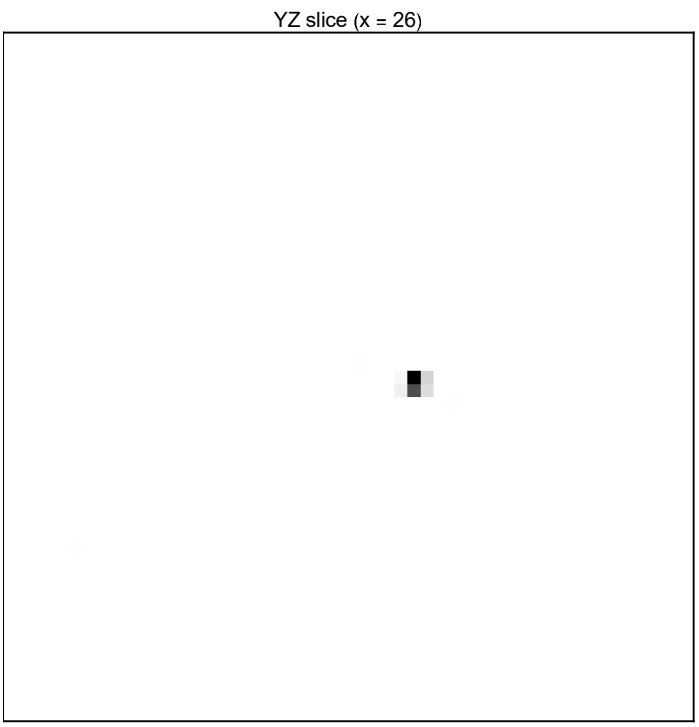

```
Central XZ slice y = 26:
```

Out[ ]=



Central YZ slice x = 26:

Out[ ]=



```
Nimage = 51

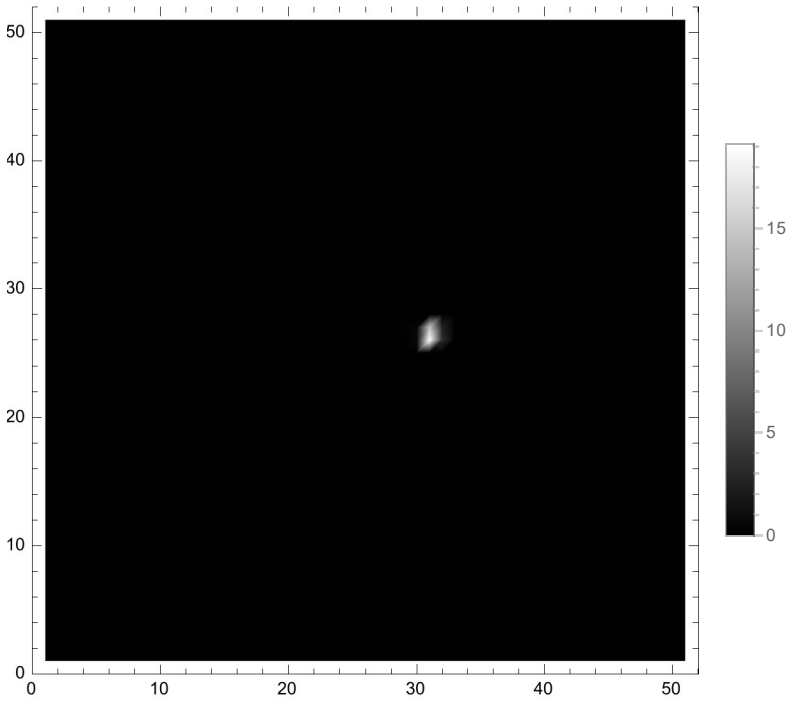
Noffset = 0

volumeMLEMmax = 19.1391

WX =

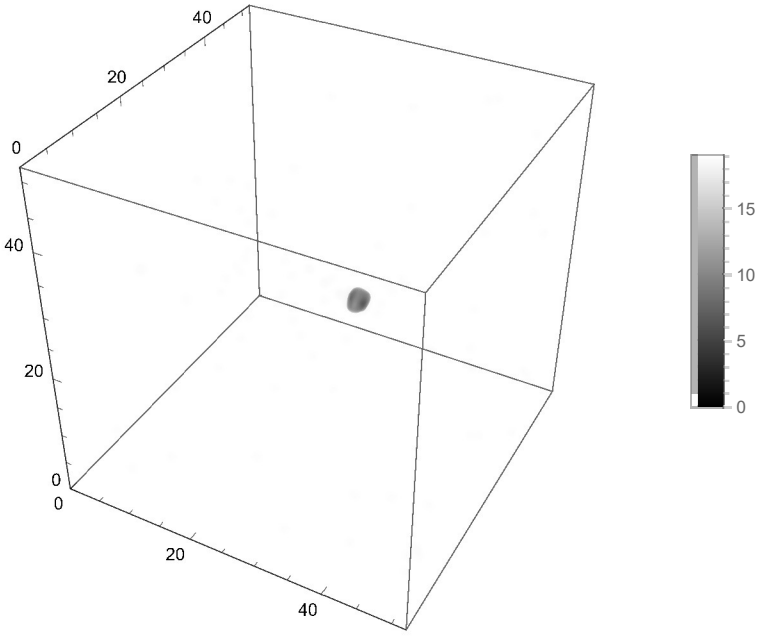
WXmax = 19.1391
```

Out[ ]=




```
IJ = {1306}
```

Out[ ]=



---

10. Reconstruct Volume Using Iterative EM+TV

Starting MAP EM TV iterations...

betaTV = 2.5

tauTV = 0.1

cTV = 132 651.

epsilonBeta = $1. \times 10^{-7}$

betaMin = 0.

betaMax = 1.5

gammaBeta = 0.25

adaptiveBetaQ = True

nIter = 200

useTVBasedBetaQ = False

betaTV initial = 2.5

tauTV = 0.1

nIter = 200

adaptiveBetaQ = True

targetDeviance = 421 362

Starting MAP EM TV iterations...

MAP-TV iter 1 | betaTV = 2.3 | dev = 1.41836×10^{6}
| tvVal = 8264.07 | min/mean/max = {0., 0.0226806, 0.327694}

MAP-TV iter 50 | betaTV = 0.978148 | dev = 476 345.
| tvVal = 3425.66 | min/mean/max = {0., 0.00813602, 19.5796}

MAP-TV iter 100 | betaTV = 0.863817 | dev = 426 327.
| tvVal = 3058.6 | min/mean/max = {0., 0.00725535, 19.7854}

MAP-TV iter 150 | betaTV = 0.851989 | dev = 421 065.
| tvVal = 3019.02 | min/mean/max = {0., 0.0071609, 19.7897}

MAP-TV iter 200 | betaTV = 0.850706 | dev = 420 489.
| tvVal = 3014.63 | min/mean/max = {0., 0.0071505, 19.7833}

MAP EM TV reconstruction complete.

volumeMAPTVmax 19.7833

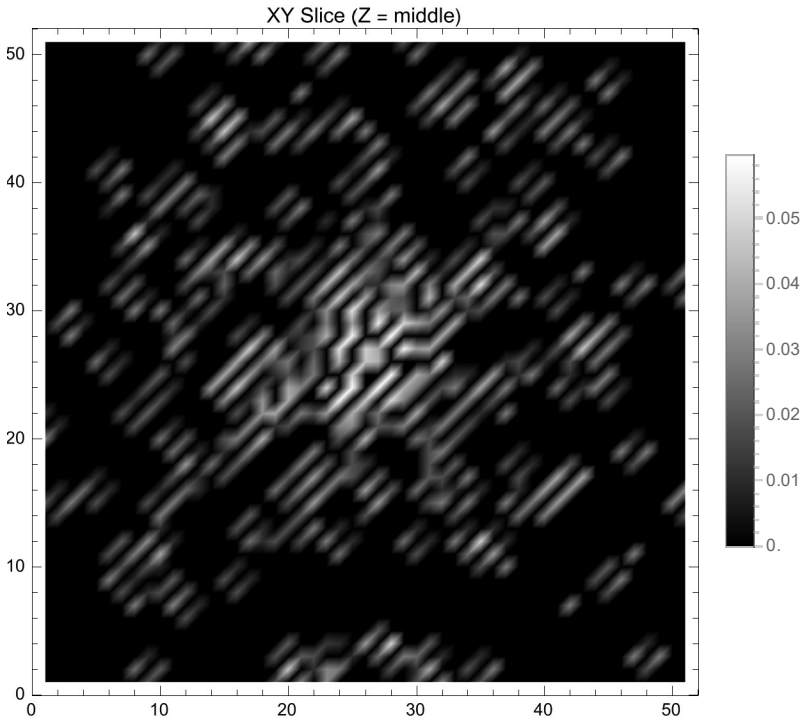
---

11. Visualize MAP TV reconstruction

volumeMAPTVmax = 19.7833

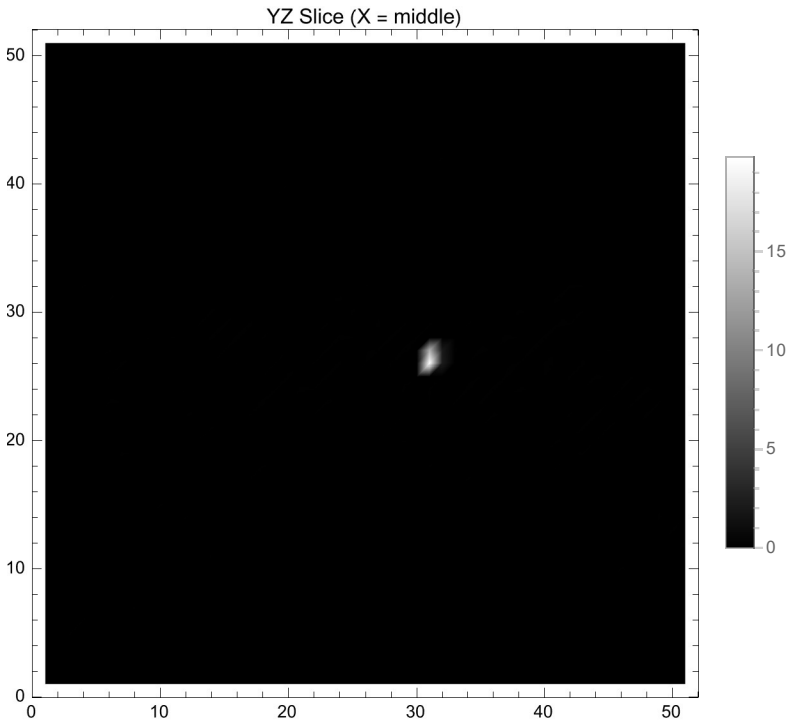
zSlicemax = Max[zSlice]0.0595857

Out[ ]=



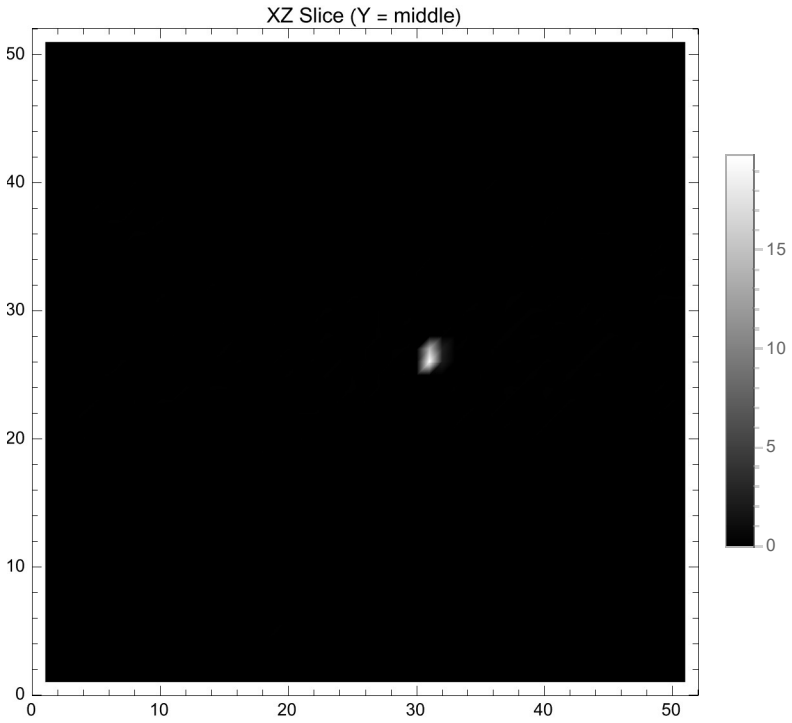

```
xSlicemax = Max[xSlice]19.7833
```

Out[ ]=




```
ySlicemax = Max[ySlice]19.7833
```

Out[ ]=



Nimage = 51

Noffset = 0

volumeMAPTVmax = 19.7833

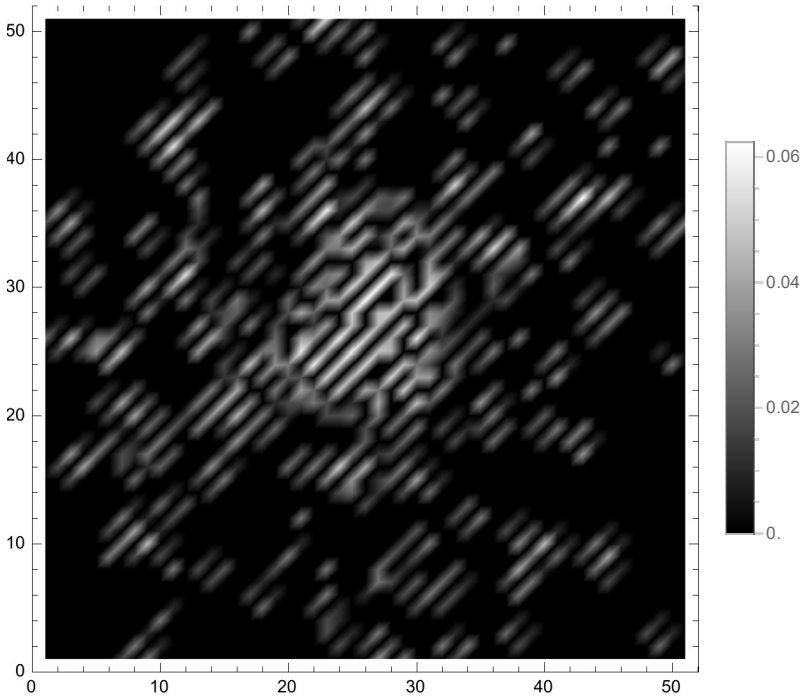
Position[volumeMAPTV] = {{26, 26, 31}}

WX =

WXmax = 0.062158

volumeMAPTV[[II,JJ,38]] =

Out[ ]=



Nimage = 51

Noffset = 0

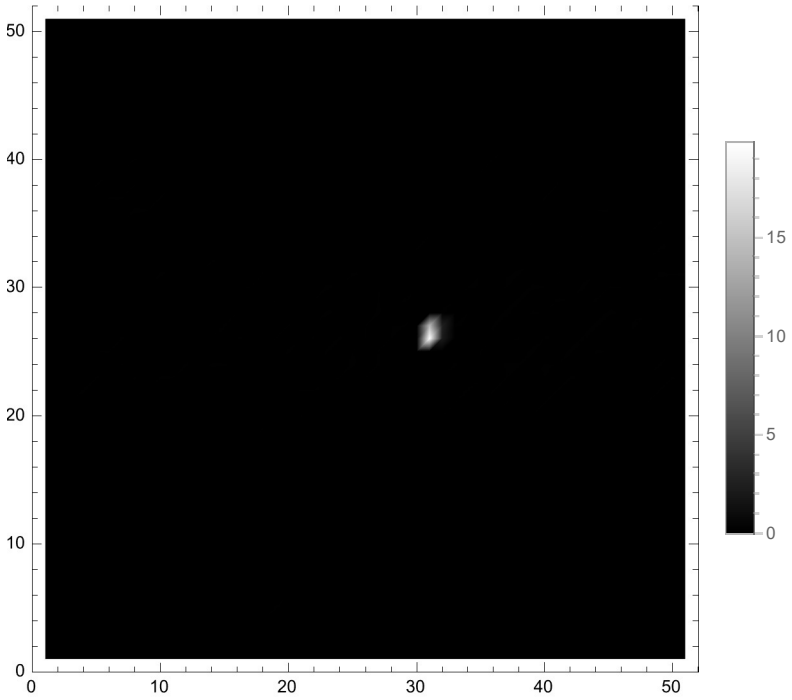
volumeMAPTVmax = 19.7833

WX =

WXmax = 19.7833

volumeMAPTV[[II,26,JJ]] =

Out[ ]=



Nimage = 51

Noffset = 0

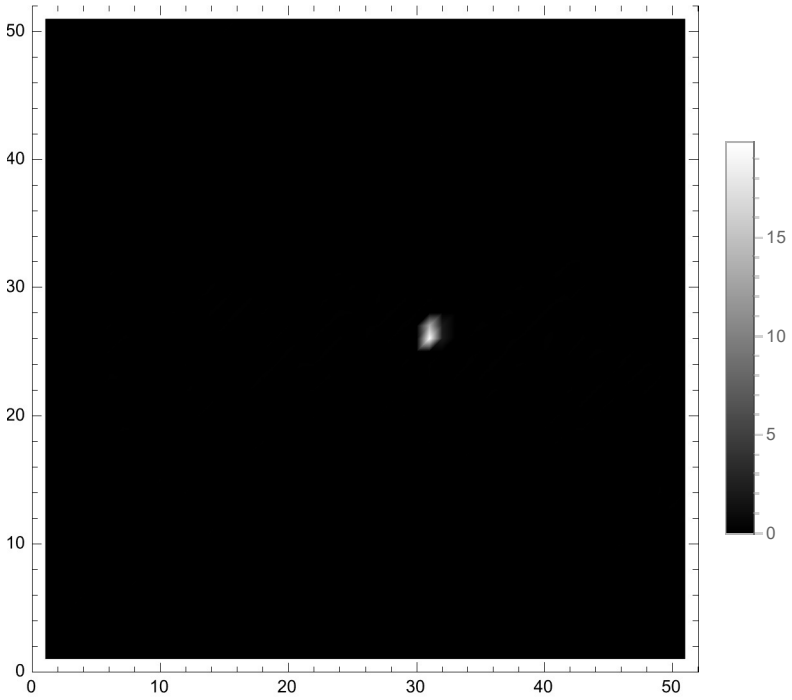
volumeMAPTVmax = 19.7833

WX =

WXmax = 19.7833

volumeMAPTV[[26,II,JJ]] =

Out[ ]=




```
IJJ = {1306}
```

Out[ ]=

