IEEE Nuclear & Plasma Sciences Society
2011 Nuclear Science Symposium and
Medical Imaging Conference

Statistical Approaches to Tomographic Reconstruction

Short Course

Sunday October 23, 2011

Program

09:30–11:00	Johan (Introduction – basic concepts)
11:00–11:30	**Coffee Break**
11:30–12:15	Johan (Algorithms in emission tomography ML, MAP)
12:15–13:00	Bruno (Iterative X-ray CT Part I)
13:00–14:30	**Lunch**
14:30–15:15	Bruno (Iterative X-ray CT Part II)
15:15–16:00	Arek (Conditional statistics in emission tomography)
16:00–16:30	**Afternoon Break**
16:30–18:00	Arek (Bayesian inference and decision making)
~18:00	**The End**

Outline:

Statistical and iterative approaches are methods of choice used for image reconstruction in emission tomography (ET) and are gaining popularity in X-Ray computed tomography (CT). The course will serve as an introduction to iterative and statistical methods of image estimation in ET and CT from projection data. The program of the course will cover fundamentals of medical tomography and common iterative and Monte Carlo methods. An introduction to general Bayesian methods in ET will also be given.

Image reconstruction in ET (Introduction to the statistical description of the data. Algorithms: ML-EM, OS-EM, MAP, etc.)
Image reconstruction in X-Ray CT (Iterative algorithms used in image reconstruction).
Bayesian statistical analysis of ET data (image creation, estimation/classification tasks using the posterior probability and Monte Carlo methods, Bayesian credible sets).
Prerequisite knowledge includes basics familiarity with the physics of emission and transmission imaging systems, statistics, and elementary algebra

Instructors:

Bruno De Man earned his B.S., M.S. and Ph.D. degrees in Electrical Engineering from the University of Leuven, where he performed research in the areas of ultrasonic tissue characterization (echocardiography) and CT iterative reconstruction for metal artifact reduction. Bruno joined GE Global Research in June 2001 and performed research in the areas of cone-beam reconstruction, iterative reconstruction, and multi-source inverse-geometry CT, among other projects. He is currently managing the CT Systems and Applications Laboratory at GE Global Research.

Johan Nuyts is research professor in the Department of Nuclear Medicine of the Katholieke Universiteit Leuven, Belgium. He received his Ph.D. in applied sciences in 1991on the subject of image reconstruction and quantification in SPECT. His research interests include iterative reconstruction in PET, SPECT and CT. Ongoing research projects focus on some multimodal imaging problems in PET/CT and PET/MRI, polychromatic CT reconstruction, multi-pinhole SPECT imaging, and motion.

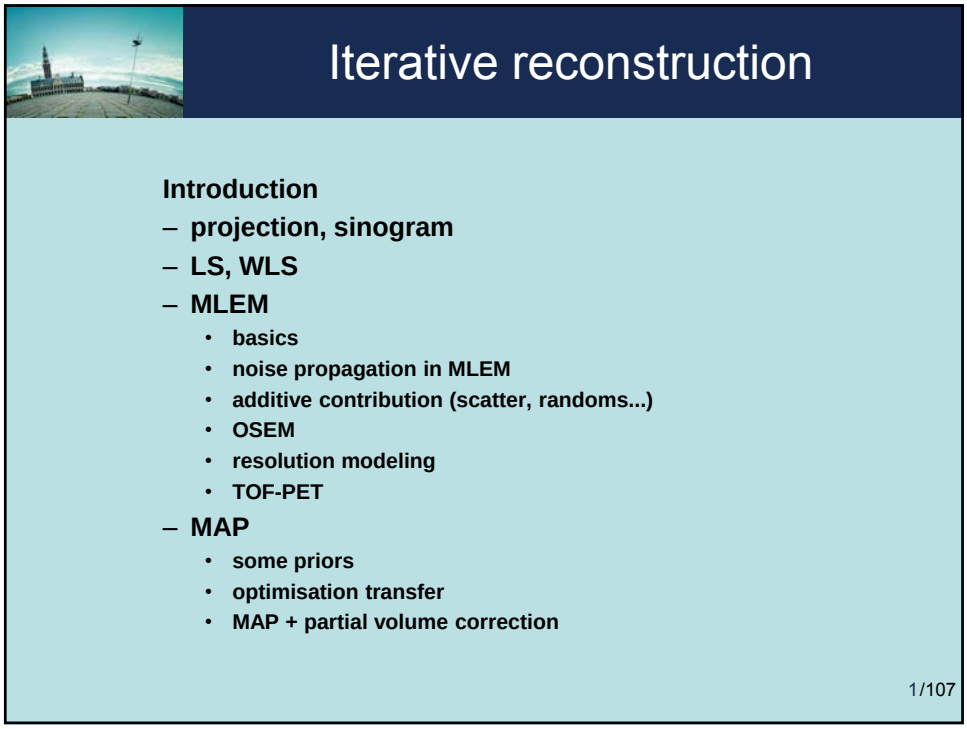
Arkadiusz Sitek is an assistant professor of Radiology at the Harvard Medical School and the Brigham and Women's Hospital in Boston. He received his M.S. degree in physics in 1994 from the University of Warsaw and Ph.D. in physics from the University of British Columbia in Vancouver, B.C. in 1998. Arek's main research interests in medical imaging include applications of statistics, high¬-performance computing, medical data visualization, and quantitation.

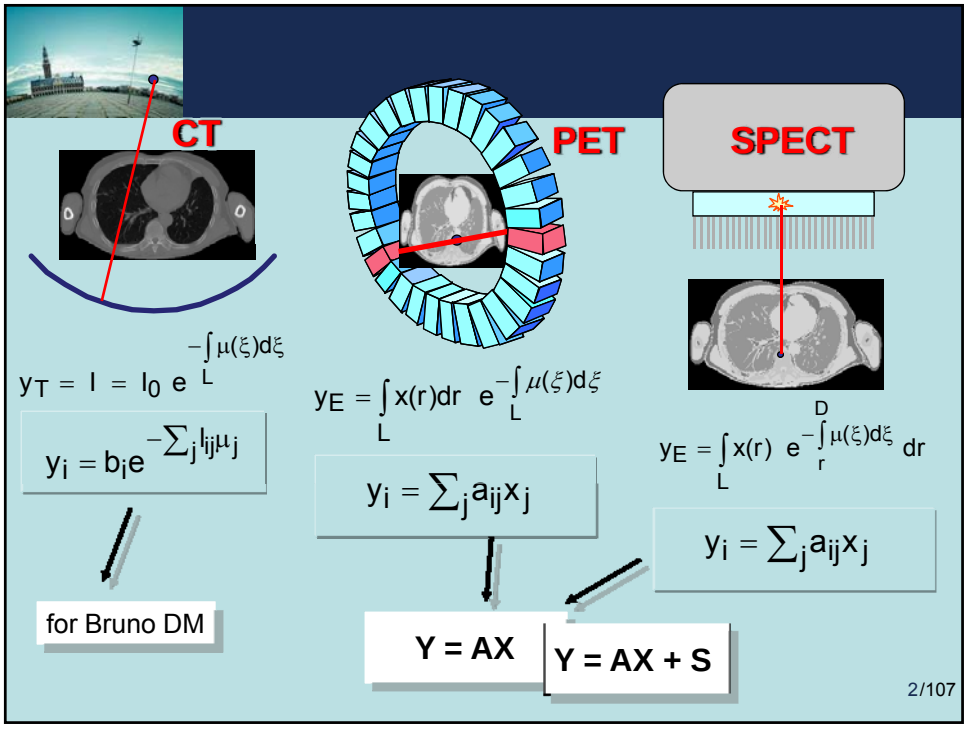
IEEE MIC Reconstruction Short Course 2011

Iterative Methods in Emission Tomography

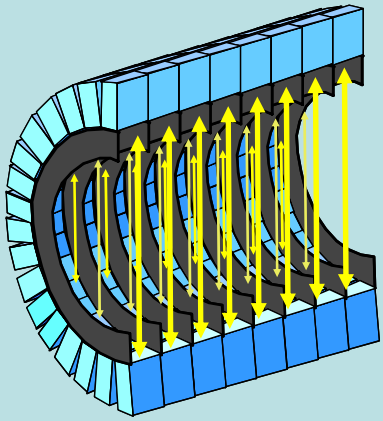
Johan Nuyts

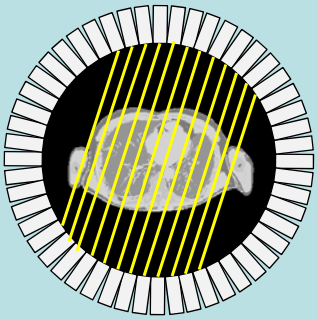
johan.nuyts@uz.kuleuven.ac.be


Iterative reconstruction
Introduction
– projection, sinogram
– LS, WLS
– MLEM
• basics
• noise propagation in MLEM
• additive contribution (scatter, randoms...)
• OSEM
• resolution modeling
• TOF-PET
– MAP
• some priors
• optimisation transfer
• MAP + partial volume correction
1/107


CT
PET
SPECT
$y_T = I = I_0 e^{-\int_L \mu(\xi)d\xi}$
$y_i = b_i e^{-\sum_j l_{ij}\mu_j}$
$y_E = \int_L x(r)dr \; e^{-\int_L \mu(\xi)d\xi}$
$y_i = \sum_j a_{ij}x_j$
$y_E = \int_L x(r) \; e^{-\int_r^D \mu(\xi)d\xi} dr$
$y_i = \sum_j a_{ij}x_j$
for Bruno DM
Y = AX
Y = AX + S
2/107

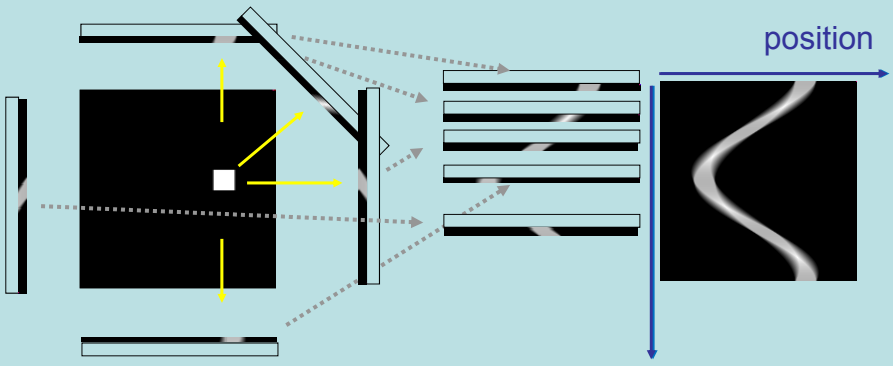
projection Y = AX





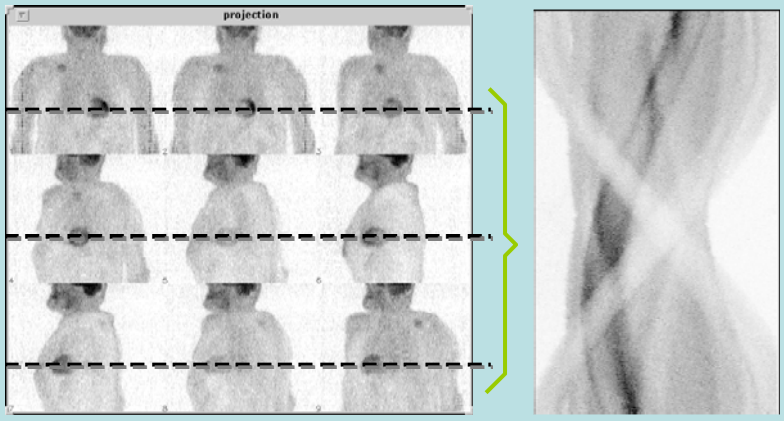
3/107

sinogram Y = AX



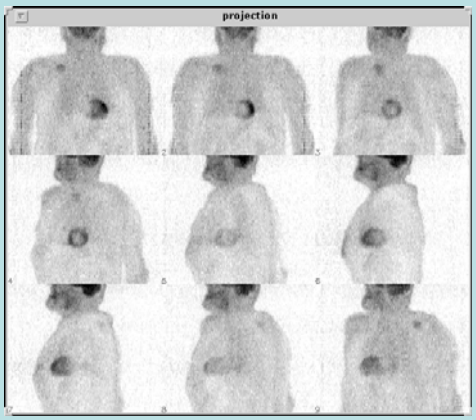
4/107

sinogram Y = AX

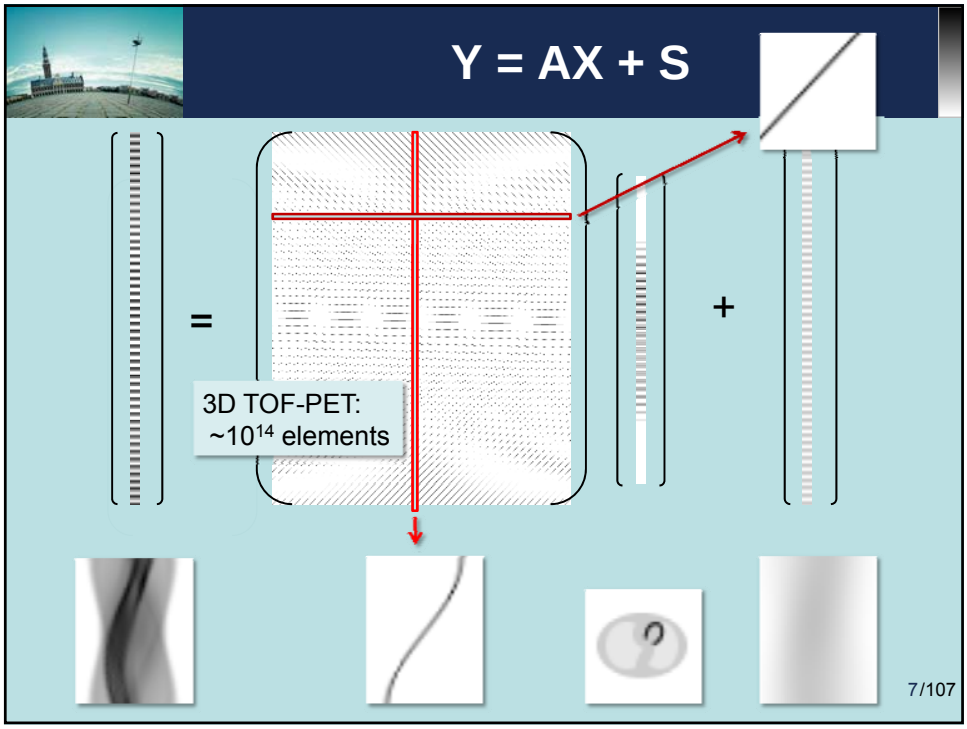


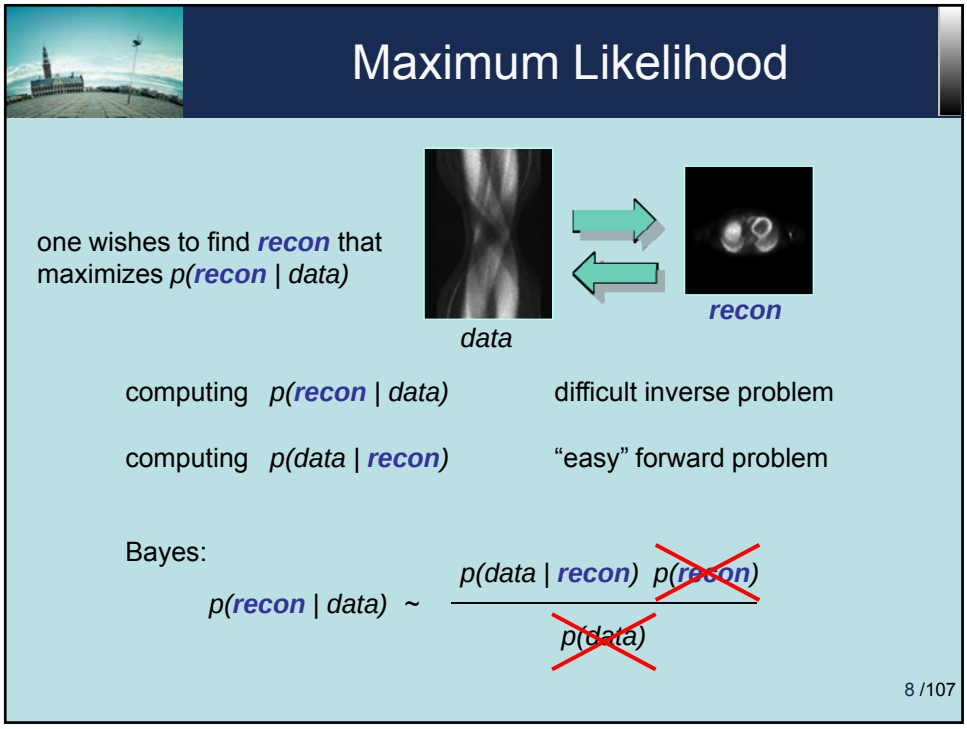
5/107

sinogram Y = AX



6/107

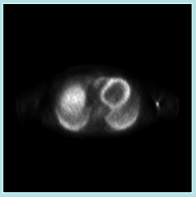

Y = AX + S
=
+
3D TOF-PET:
~10^14 elements
7/107


Maximum Likelihood
one wishes to find recon that
maximizes p(recon | data)
data
recon
computing p(recon | data)
difficult inverse problem
computing p(data | recon)
"easy" forward problem
Bayes:
p(recon | data) ~ p(data | recon) p(recon) / p(data)
8 /107

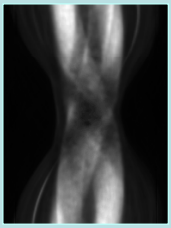
Maximum Likelihood

$p(\textbf{recon} \mid data) \sim p(data \mid \textbf{recon})$

recon

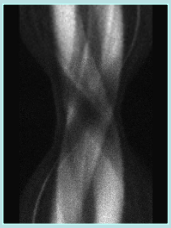


projection



noise

data



x_j

$j = 1..J$

$\hat{y}_i = \sum_j a_{ij} x_j + s_i$

$i = 1..I$

y_i

$p(data \mid \textbf{recon}) = \prod_i p(y_i \mid \hat{y}_i)$

noise models: Gaussian, Poisson...

9 /107

Maximum likelihood: Gaussian

Least squares:

maximize $p(Y \mid AX) \sim \prod_i \exp\left(-\frac{([AX]_i + s_i - y_i)^2}{2\sigma^2}\right)$ with $[AX]_i = \sum_j a_{ij} x_j$

or maximize $\ln p(Y \mid AX) = \sum_i -\frac{([AX]_i + s_i - y_i)^2}{2\sigma^2}$

or minimize $L = \sum_i ([AX]_i + s_i - y_i)^2$

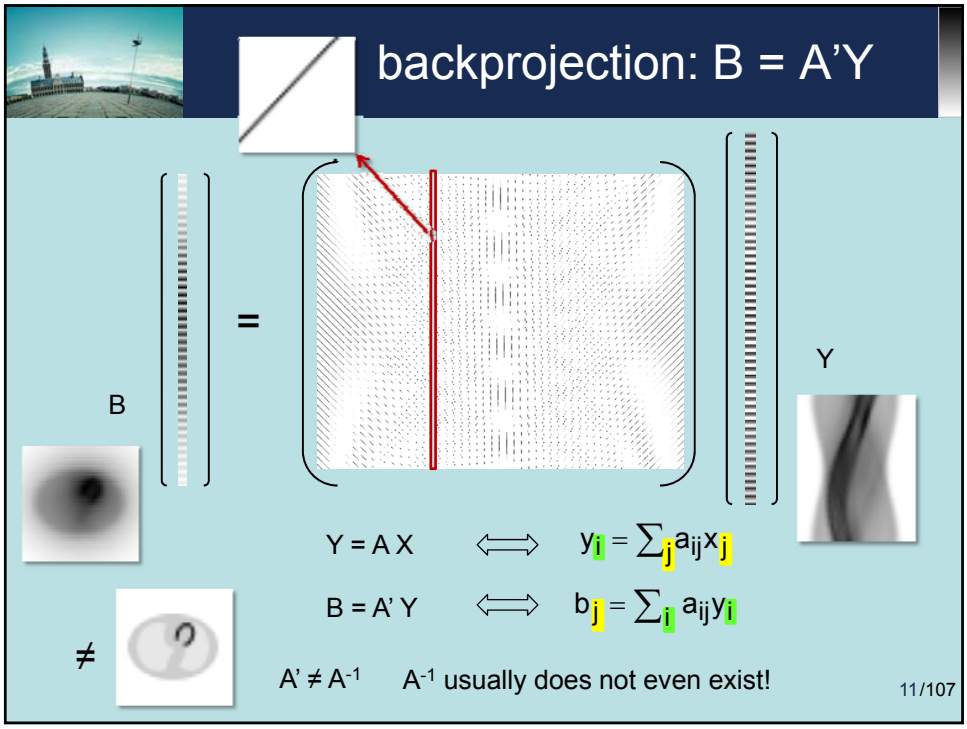
Matrix notation: $L = (AX+S-Y)'(AX+S-Y)$

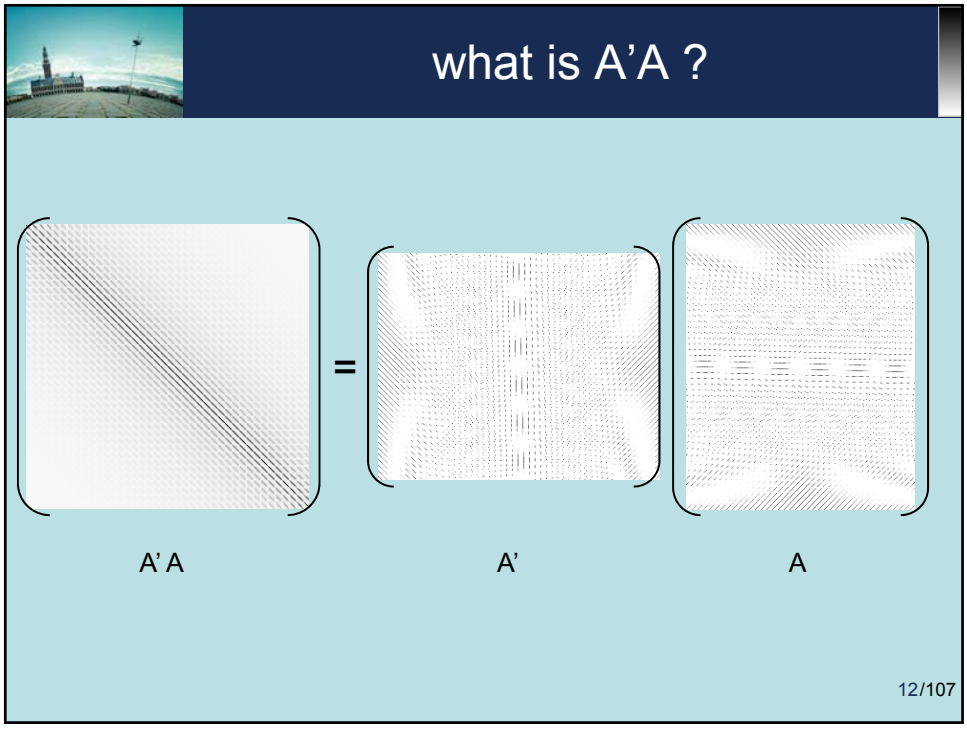
Solution: $\frac{\partial L}{\partial x_j} = 0 \Rightarrow \forall j: \sum_i a_{ij}\left(\sum_k a_{ik} x_k + s_i - y_i\right) = 0$

$A'(AX + S - Y) = 0$

$X = [A'A]^{-1} A'(Y - S) \Leftarrow A'AX - A'(Y - S) = 0$

10/107

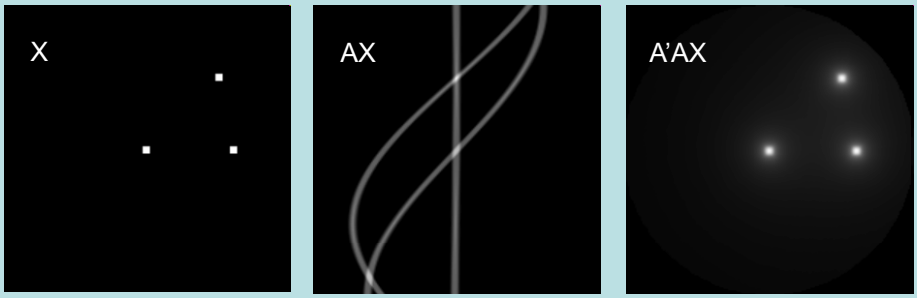

backprojection: B = A'Y
B
=
Y
Y = A X ⟺ $y_i = \sum_j a_{ij} x_j$
B = A' Y ⟺ $b_j = \sum_i a_{ij} y_i$
≠
A' ≠ A^{-1}
A^{-1} usually does not even exist!
11/107


what is A'A ?
=
A' A
A'
A
12/107

what is A'A and $[A'A]^{-1}$?

A'A is the backprojection of the projection

case 1: shift invariant A.
ideal parallel beam projection (no atten, no PSF, no scatter, no...)

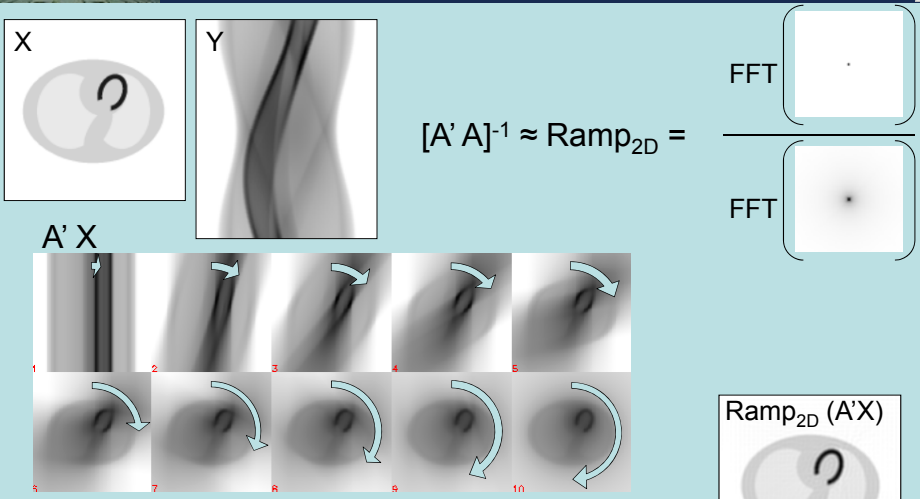


A'A is a giant matrix, but it represents simple shift invariant blurring.
$[A'A]^{-1}$ is deblurring: RAMP filter

$X = [A'A]^{-1} A'Y$ is BPF, equivalent to FBP!

13/107

BPF

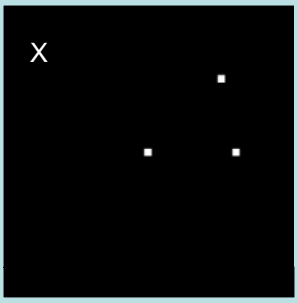


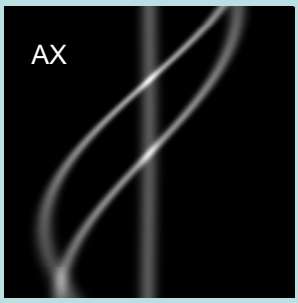
14/107

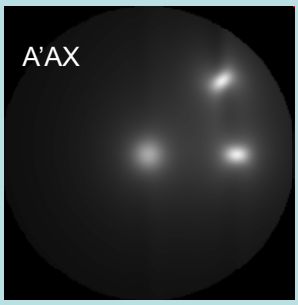
what is A'A and [A'A]$^{-1}$?

case 2: shift variant A.

SPECT projection with collimator blurring (no atten, no...)







A'A is a giant matrix, representing a position dependent filter.
[A'A]$^{-1}$ is shift variant deblurring, very complicated!!

$X = [A'A]^{-1} A'Y$ cannot be computed directly

15/107

WLS-reconstruction: solving AX = Y

Weighted least squares

minimize $L = \sum_i \frac{([AX]_i + s_i - y_i)^2}{\sigma_i^2}$ with $[AX]_i = \sum_j a_{ij} x_j$

Matrix notation:

$$L = (AX + S - Y)' C_y^{-1} (AX + S - Y)$$

$$C_y = \begin{bmatrix} \sigma_1 & 0 & 0 & & 0 \\ 0 & \sigma_2 & 0 & & \vdots \\ 0 & 0 & \sigma_3 & & 0 \\ & & & \ddots & 0 \\ 0 & \cdots & 0 & 0 & \sigma_M \end{bmatrix}$$

Solution: $\frac{\partial L}{\partial x_j} = 0 \Longrightarrow X = \left[A' C_y^{-1} A\right]^{-1} A' C_y^{-1} (Y - S)$

Even if A'A is shift invariant, A'C$_y^{-1}$A is not!

16/107

what is $A'C_y^{-1}A$ and $[A' C_y^{-1}A]^{-1}$?

Emission image

Y

approximate Poisson noise
with Gaussian: Cy ≈ Diag(Y)

$$C_y = \begin{bmatrix} y_1 & 0 & 0 & & 0 \\ 0 & y_2 & 0 & & \vdots \\ 0 & 0 & y_3 & & 0 \\ & & & \ddots & 0 \\ 0 & \cdots & 0 & 0 & y_M \end{bmatrix}$$

X

AX

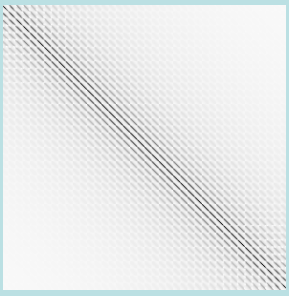
$C_y^{-1}A\,X$

$A'C_y^{-1}A\,X$

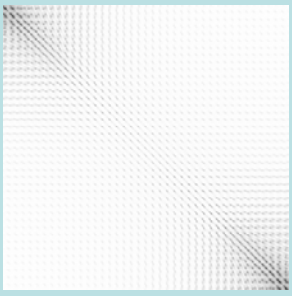
$$X = \left[A'C_y^{-1}A\right]^{-1} A'C_y^{-1}\,Y$$ cannot be computed directly!!

17/107

what is $A'C_y^{-1}A$ and $[A' C_y^{-1}A]^{-1}$?



$A' A$



$A'C_y^{-1}A$

18/107

Iterative reconstruction

Introduction
- projection, sinogram
- LS, WLS
- **MLEM**
  - **basics**
  - **noise propagation in MLEM**
  - **additive contribution (scatter, randoms...)**
  - **OSEM**
  - **resolution modeling**
  - **TOF-PET**
- MAP
  - some priors
  - optimisation transfer
  - MAP + partial volume correction

19/107

ML for Poisson data

20/107

Poisson noise

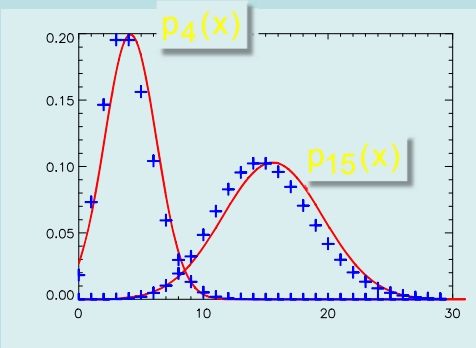
Chance of measuring n photons $p_\lambda(n) = \frac{e^{-\lambda}\lambda^n}{n!}$
when λ are expected :

$$\left(\text{note that : } e^{-\lambda}\sum_n \frac{\lambda^n}{n!} = e^{-\lambda}e^{\lambda} = 1\right)$$

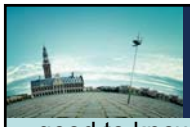
Poisson distribution resembles Gaussian:

$$p_\lambda(n) \approx \frac{1}{\sqrt{2\pi}\sqrt{\lambda}} e^{-\frac{(n-\lambda)^2}{2\lambda}}$$

$$\approx \lambda \pm \sqrt{\lambda}$$

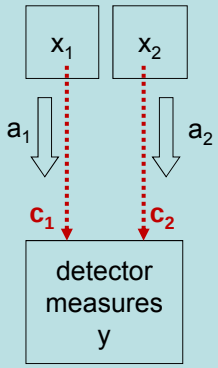


21/107



ML

good to know...



- two vials with activities x_1 and x_2
- detection sensitivities a_1 and a_2

real experiment: detector measures y

question: how many photons c_1 and c_2 were contributed by each of the vials?

answer:

1. $E(c_1) = a_1 x_1$ and $E(c_2) = a_2 x_2$
2. But we know also $y = c_1 + c_2$
   What if $y \neq a_1 x_1 + a_2 x_2$?
3. Solution: $E(c_1 \mid y) = a_1 x_1 \frac{y}{a_1 x_1 + a_2 x_2}$

22/107

ML-reconstruction: solving AX+S = Y

Poisson noise: $p(y \mid \varphi) = e^{-\varphi} \frac{\varphi^y}{y!}$

maximize p(Y | AX+S) $= \prod_i \frac{e^{-[AX]_i - s_i} ([AX]_i + s_i)^{y_i}}{y_i!}$

or maximize ln p(Y | AX+S) $= \sum_i y_i \ln([AX]_i + s_i) - [AX]_i - s_i - \ln(y_i!)$

or maximize L $= \sum_i y_i \ln([AX]_i + s_i) - [AX]_i$

Solution: $\frac{\partial L}{\partial x_j} = 0 \Rightarrow$

$$\sum_i a_{ij} \left(\frac{y_i}{[AX]_i + s_i} - 1 \right)$$

$$= \sum_i a_{ij} \left(\frac{y_i}{\sum_k a_{ik} x_k + s_i} - 1 \right) = 0$$

23/107

Expectation Maximisation, Y = AX

"complete set" $y_i = \sum_j c_{ij}$ $E(c_{ij}) = a_{ij} x_j$

log-likelihood $L_{comp}(c) = \sum_i \sum_j (c_{ij} \ln(a_{ij} x_j) - a_{ij} x_j)$

E-step $E\left(L_{comp}(x) \mid y, x^{old}\right) = \sum_i \sum_j (n_{ij} \ln(a_{ij} x_j) - a_{ij} x_j)$

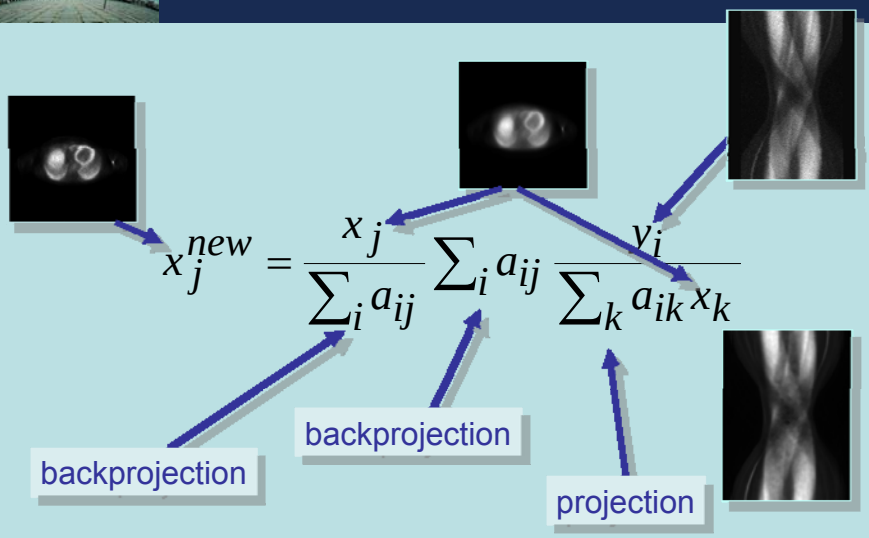
$$n_{ij} = a_{ij} x_j^{old} \frac{y_i}{\sum_k a_{ik} x_k^{old}}$$

M-step $\frac{\partial}{\partial x_j} E\left(L_x(x) \mid y, x^{old}\right) = 0$

MLEM $x_j = \frac{\sum_i n_{ij}}{\sum_i a_{ij}}$

24/107

MLEM



25/107

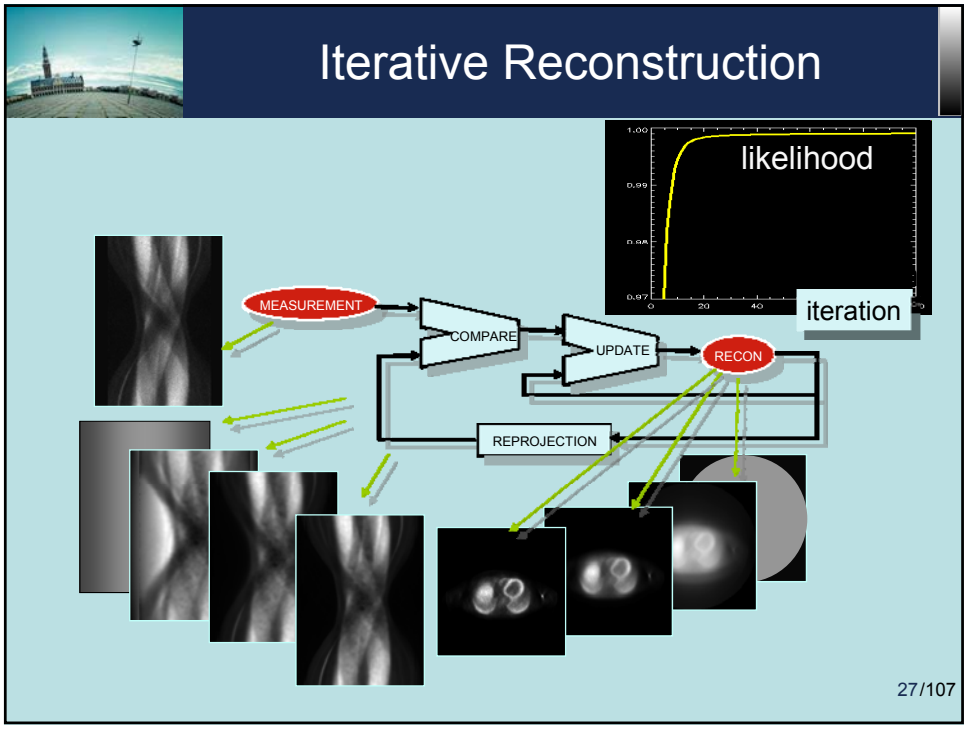
MLEM is a gradient ascent algorithm

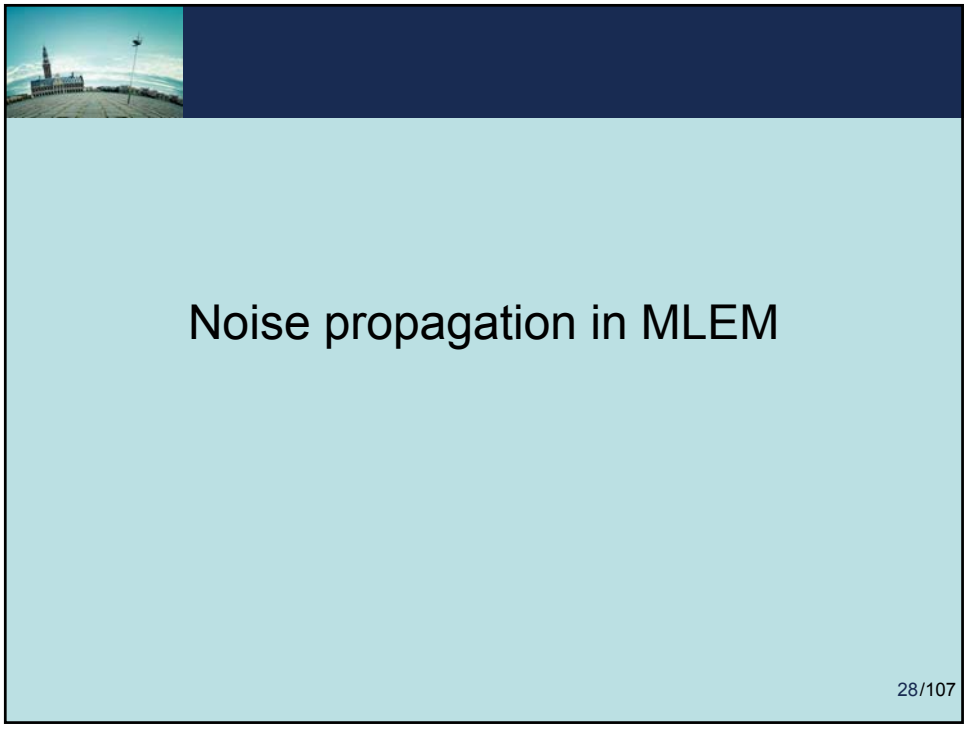
$$x_j = \frac{x_j^{old}}{\sum_i a_{ij}} \sum_i a_{ij} \frac{y_i}{\sum_k a_{ik} x_k^{old}} = x_j^{old} + \frac{x_j^{old}}{\sum_i a_{ij}} \sum_i a_{ij} \left(\frac{y_i}{\sum_k a_{ik} x_k^{old}} - 1 \right)$$

$$\frac{\partial L_y(x)}{\partial x_j} = \sum_i a_{ij} \left(\frac{y_i}{\sum_k a_{ik} x_k} - 1 \right)$$

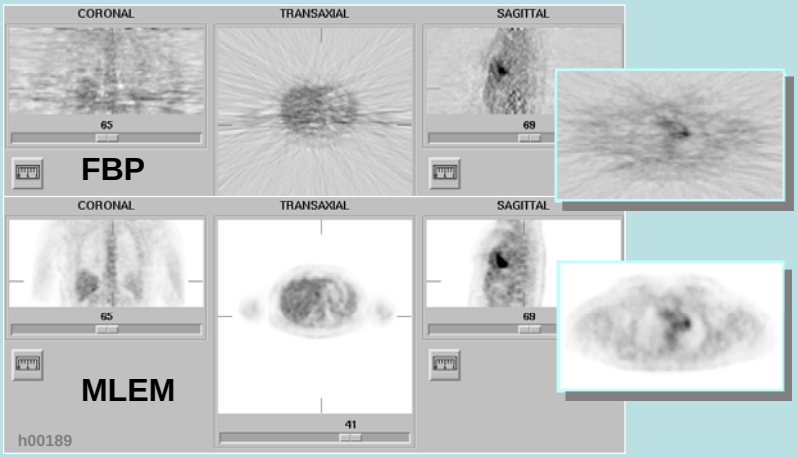
$$x_j = x_j^{old} + \frac{x_j^{old}}{\sum_i a_{ij}} \left. \frac{\partial L_y}{\partial x_j} \right|_{x^{old}}$$

26/107


Iterative Reconstruction
likelihood
iteration
MEASUREMENT
COMPARE
UPDATE
RECON
REPROJECTION
27/107


Noise propagation in MLEM
28/107

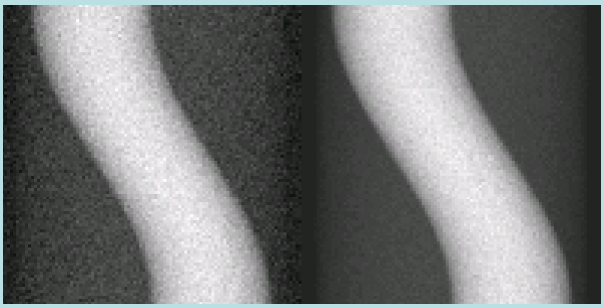
FBP vs MLEM



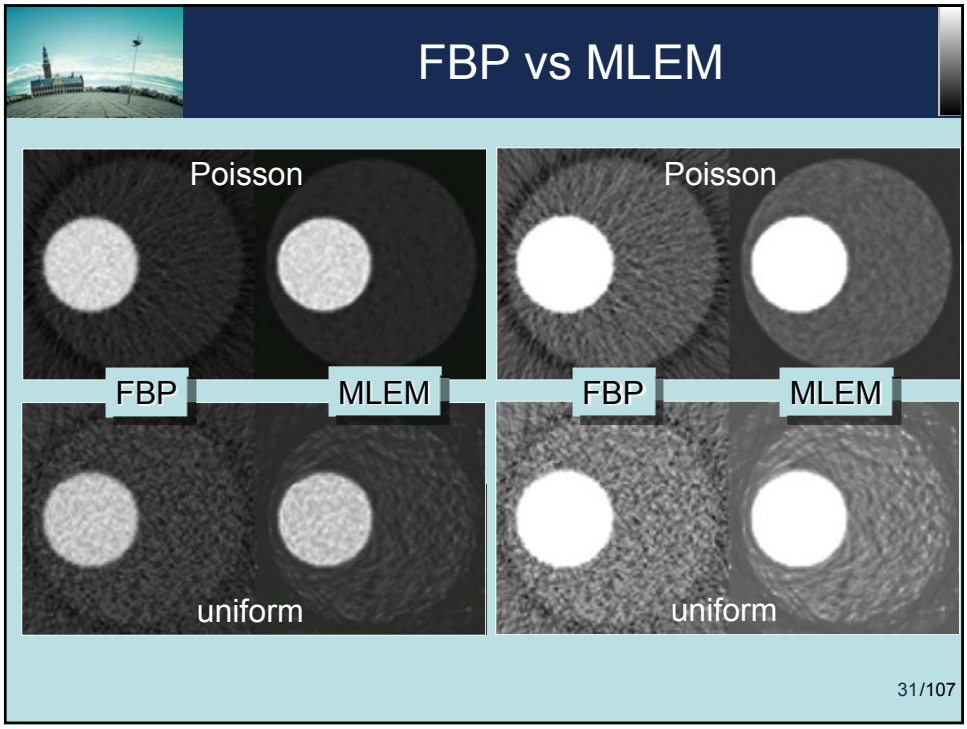
29/107

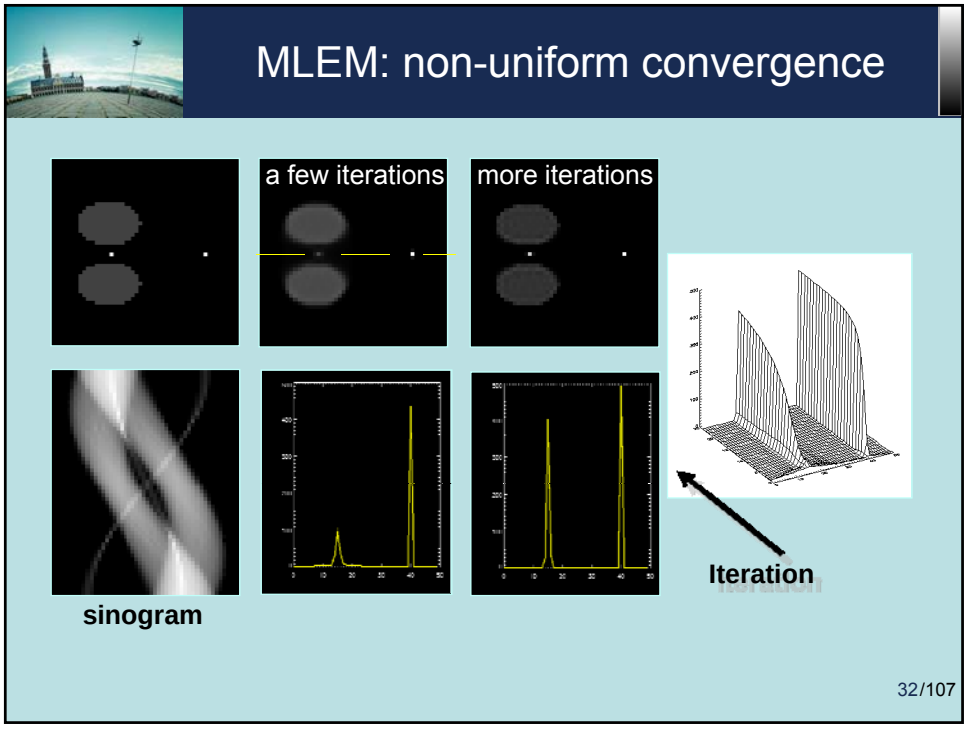
FBP vs MLEM

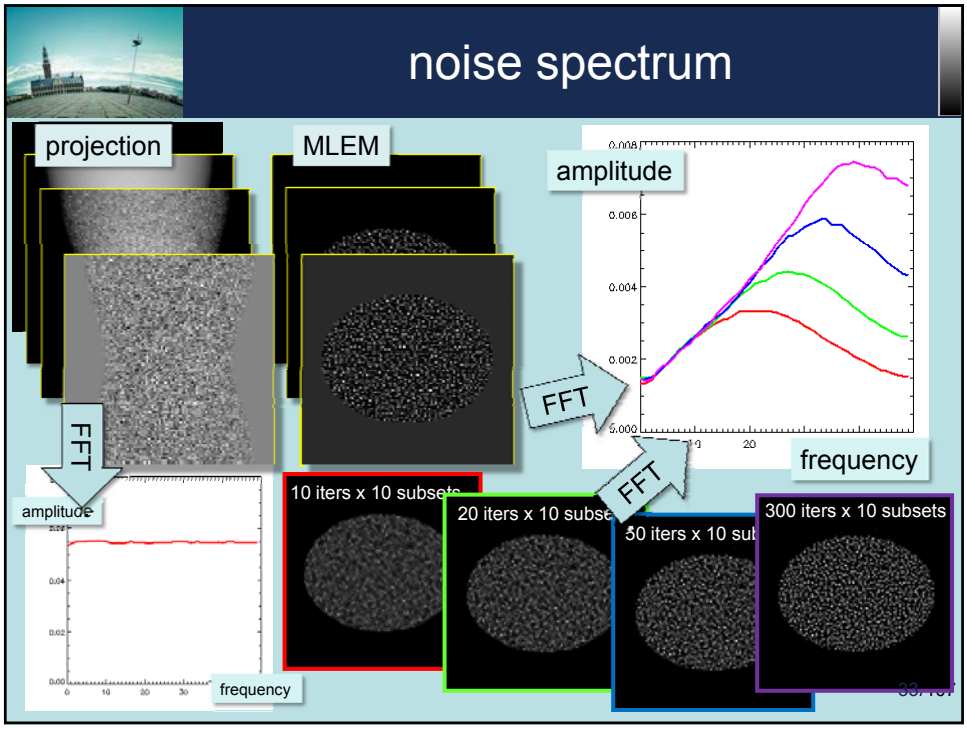
uniform Poisson

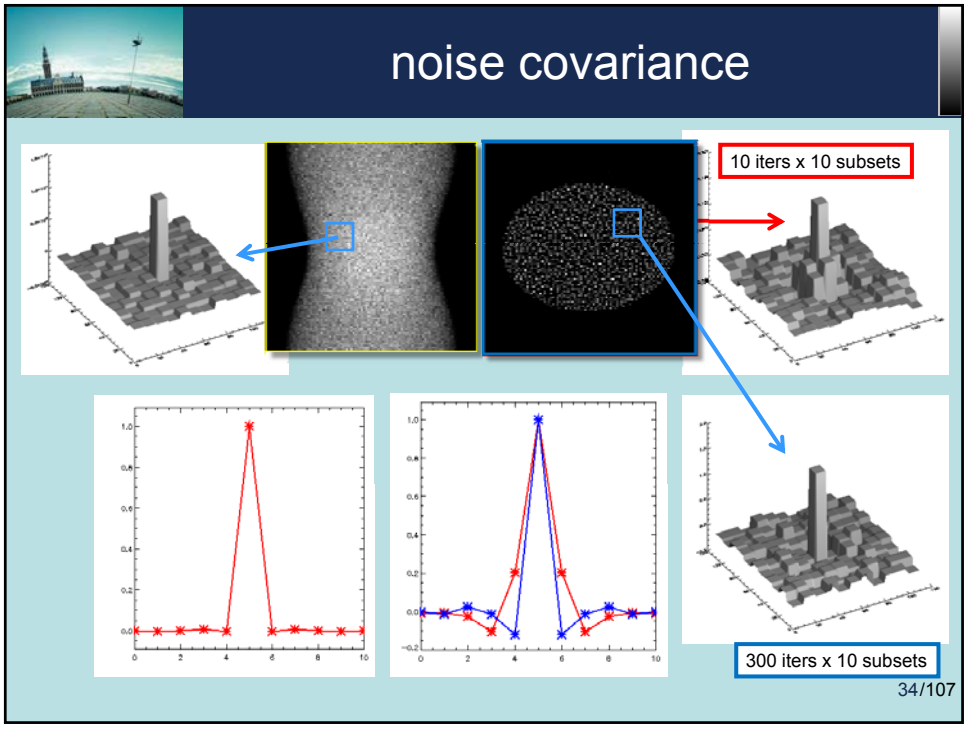


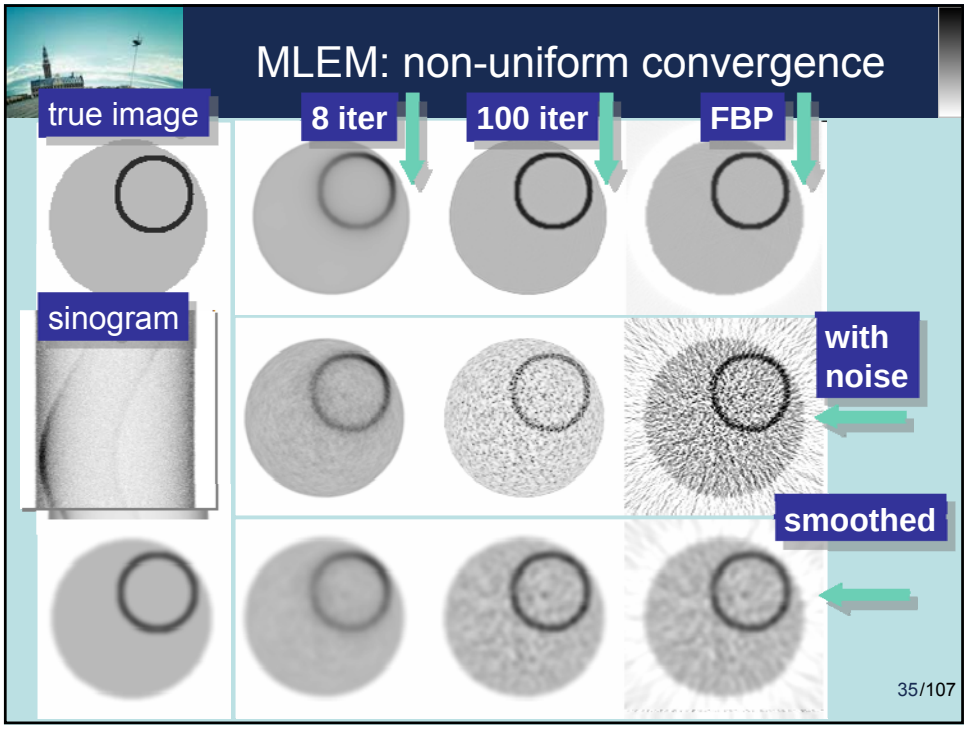
30/107

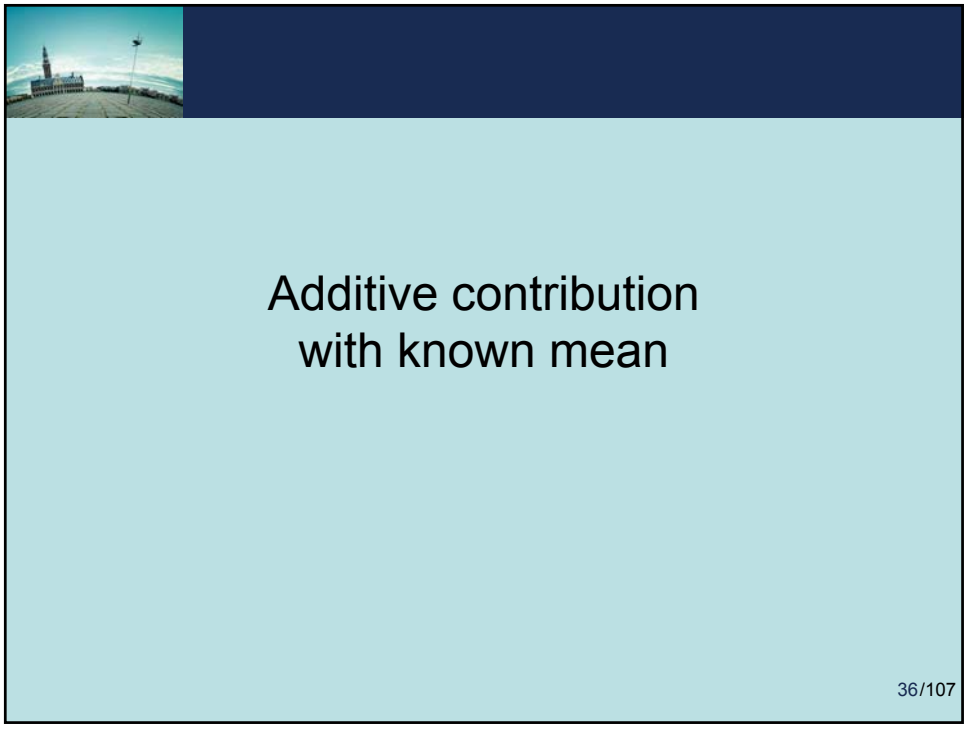

FBP vs MLEM
Poisson
Poisson
FBP
MLEM
FBP
MLEM
uniform
uniform
31/107


MLEM: non-uniform convergence
a few iterations
more iterations
Iteration
sinogram
32/107


noise spectrum
projection
MLEM
amplitude
FFT
FFT
FFT
frequency
amplitude
frequency
10 iters x 10 subsets
20 iters x 10 subse
30 iters x 10 sub
300 iters x 10 subsets


noise covariance
10 iters x 10 subsets
300 iters x 10 subsets
34/107


MLEM: non-uniform convergence
true image
8 iter
100 iter
FBP
sinogram
with noise
smoothed
35/107


Additive contribution
with known mean
36/107

MLEM, additive contamination

$y_i = \sum_j c_{ij} + d_i$ measured counts = signal + contamination

s_i separate measurement of contamination

δ ratio of scan times, so $s_i \approx \delta\, d_i$

Complete variables: c_{ij} d_i s_i

To be solved: x_j D_i

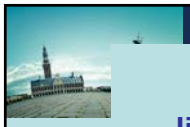
Relations:

$$c_{ij} = \text{Poisson}\,(a_{ij} x_j)$$

$$d_i = \text{Poisson}(D_i)$$

$$s_i = \text{Poisson}(\delta\, D_i)$$

37/107



likelihood:

$$\prod_{ij} \frac{e^{-a_{ij}x_j}(a_{ij}x_j)^{c_{ij}}}{c_{ij}!} \frac{e^{-D_i}D_i^{d_i}}{d_i!} \frac{e^{-\delta D_i}D_i^{s_i}}{s_i!}$$

apply E-step and M-step:

$$x_j = \frac{x_j^{old}}{\sum_i a_{ij}} \sum_i a_{ij} \frac{y_i}{\sum_k a_{ik} x_k^{old} + D_i^{old}}$$

$$D_i = \frac{1}{1+\delta}\left(D_i^{old} \frac{y_i}{\sum_k a_{ik} x_k^{old} + D_i^{old}} + s_i \right)$$

if δ >> 1:

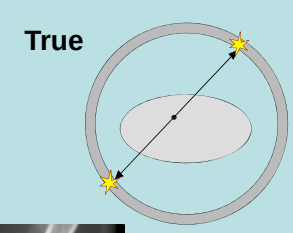
$D_i = \frac{s_i}{\delta}$ **and**

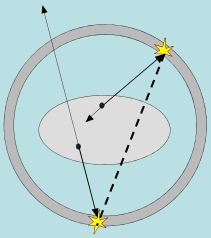
$$x_j = \frac{x_j^{old}}{\sum_i a_{ij}} \sum_i a_{ij} \frac{y_i}{\sum_k a_{ik} x_k^{old} + D_i}$$

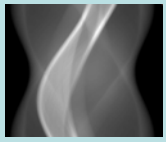
in this case, D_i is known (almost) exactly!

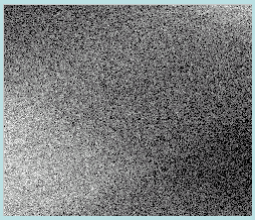
38/107

Shifted Poisson









Random or Delayed

Randoms from singles

Stearns et al., NSS-MIC IEEE 2003

39/107

Shifted Poisson

$$\begin{cases} Y = T + R \\ \hat{R} \end{cases} \qquad x_j = \frac{x_j^{old}}{\sum_i a_{ij}} \sum_i a_{ij} \frac{t_i + r_i}{\hat{t}_i(X) + \hat{r}_i} \qquad t_i + r_i = \text{Poisson}\left(\hat{t}_i + \hat{r}_i\right)$$

1. Suppose $\hat{r}_i = \bar{r}_i$ is noise-free:

$$\text{mean}(t_i + r_i) = \text{mean}\left(\hat{t}_i + \hat{r}_i\right) = \hat{t}_i + \bar{r}_i$$
$$\text{var}\left(t_i + r_i - \left(\hat{t}_i + \hat{r}_i\right)\right) = \text{var}(t_i + r_i) = \hat{t}_i + \bar{r}_i$$

OK!

2. Suppose $\hat{r}_i$ is Poisson $(\bar{r}_i)$

$$\text{mean}(t_i + r_i) = \text{mean}\left(\hat{t}_i + \hat{r}_i\right) = \hat{t}_i + \bar{r}_i$$
$$\text{var}\left(t_i + r_i - \left(\hat{t}_i + \hat{r}_i\right)\right) = \text{var}\left(t_i + r_i + \hat{r}_i\right) = \hat{t}_i + 2\bar{r}_i$$

not OK!

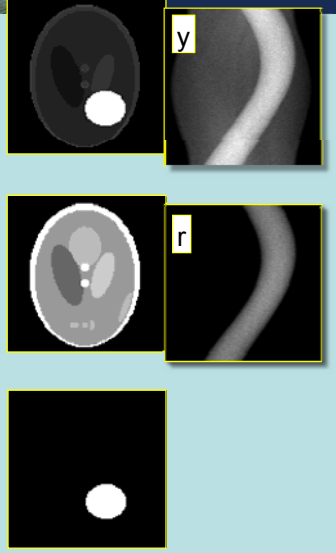
Find s such that $\text{var}\left(t_i + r_i + s_i - \left(\hat{t}_i + \hat{r}_i + s_i\right)\right) = \text{mean}\left(\hat{t}_i + \hat{r}_i + s_i\right) = \hat{t}_i + \bar{r}_i + s_i$

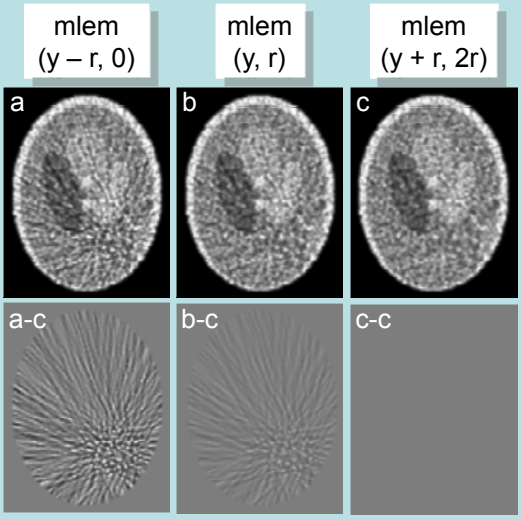
$\Rightarrow s_i = \bar{r}_i$

$$x_j = \frac{x_j^{old}}{\sum_i a_{ij}} \sum_i a_{ij} \frac{y_i + s_i}{\hat{t}_i(X) + \hat{r}_i + s_i}$$

40/107

Shifted Poisson: (extreme) example





41/107

OSEM

ordered subsets EM

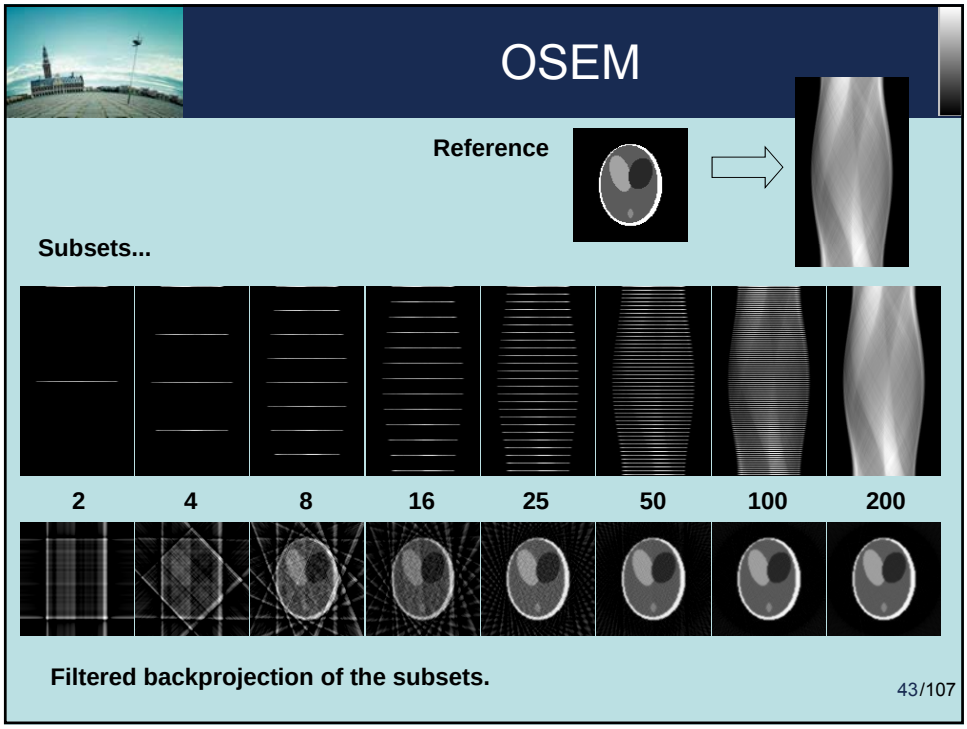
em

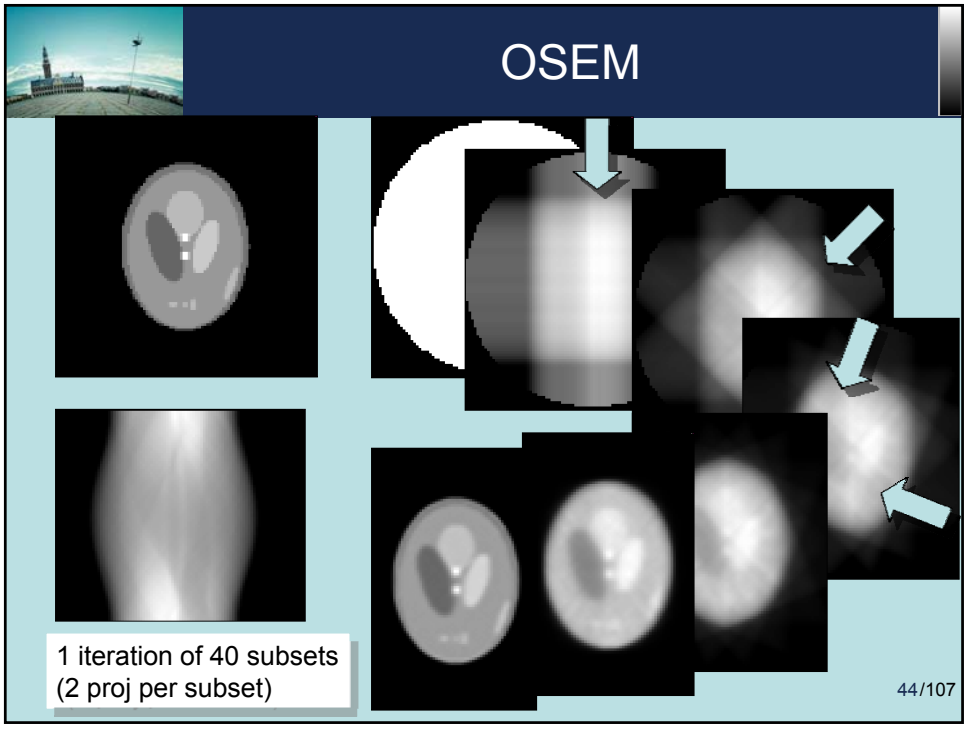
$$x_j^{(k+1)} = \frac{x_j^{(k)}}{\sum_i a_{ij}} \sum_i a_{ij} \frac{y_i}{\sum_\xi a_{i\xi} x_\xi^{(k)} + s_i}$$

osem

$$x_j^{(k+1)} = \frac{x_j^{(k)}}{\sum_{i \in S_k} a_{ij}} \sum_{i \in S_k} a_{ij} \frac{y_i}{\sum_\xi a_{i\xi} x_\xi^{(k)} + s_i}$$

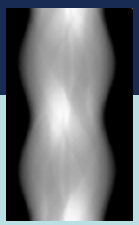
42/107


OSEM
Reference
Subsets...
2
4
8
16
25
50
100
200
Filtered backprojection of the subsets.
43/107

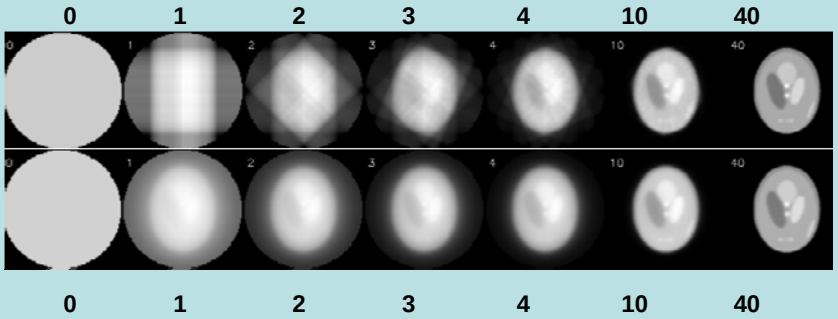

OSEM
1 iteration of 40 subsets
(2 proj per subset)
44/107

OSEM

Reference



1 OSEM iteration with 40 subsets



MLEM-iterations

45/107

OSEM

initial image

no noise (and subset balance)

with noise

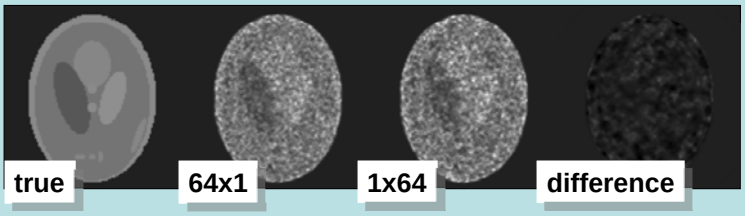
Convergence to limit cycle

Solutions:

- apply converging block-iterative algorithm:
  sacrifize some speed for guaranteed convergence
- gradually decrease the number of subsets
- ignore the problem (you may not want convergence anyway)

46/107

OSEM

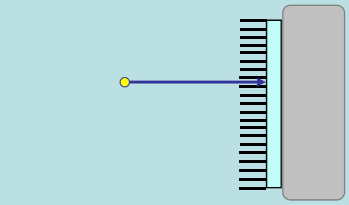


47/107

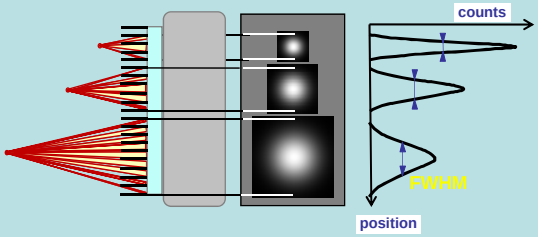
(back) projection model: model for image resolution

48/107

(back)projection in SPECT



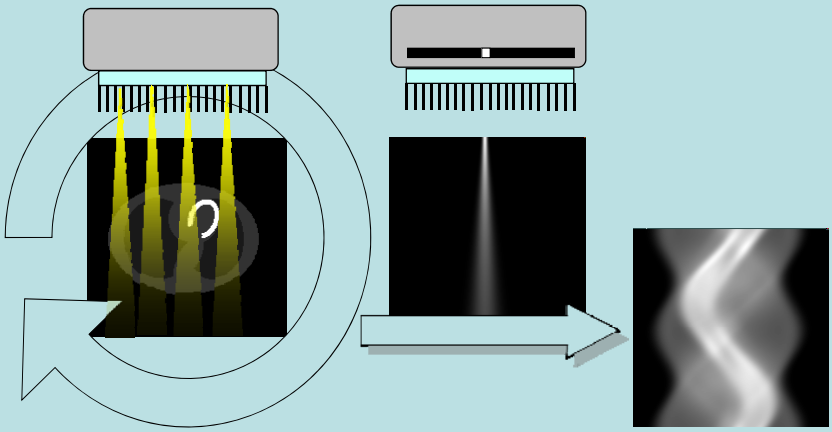
simple ray tracing



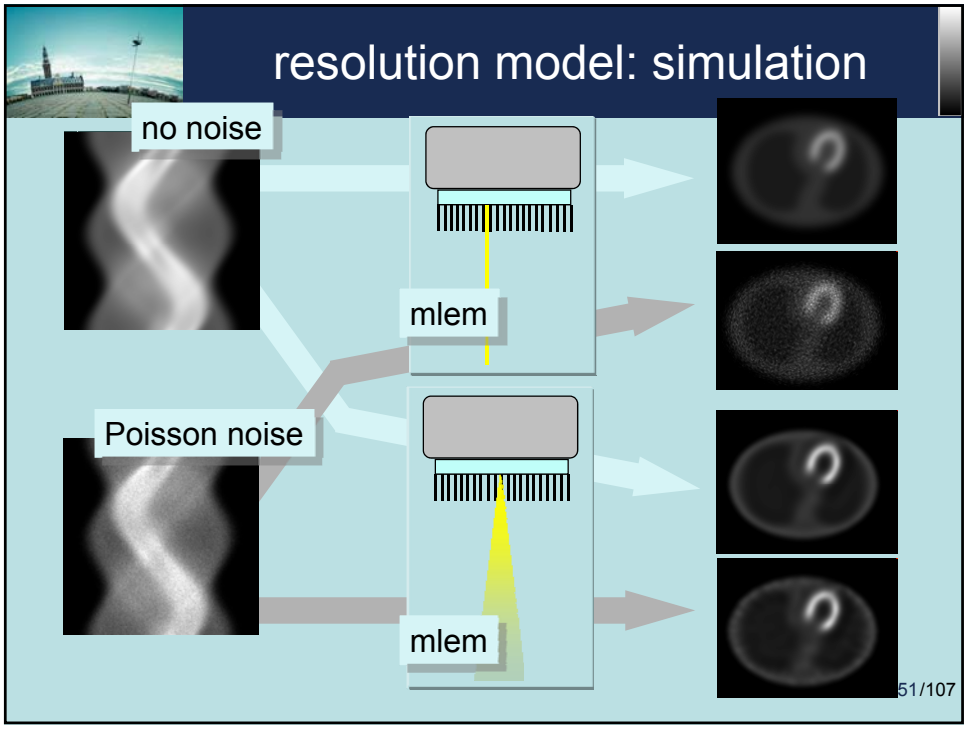
multiple rays
or
Gaussian diffusion
or
Frequency distance principle
....

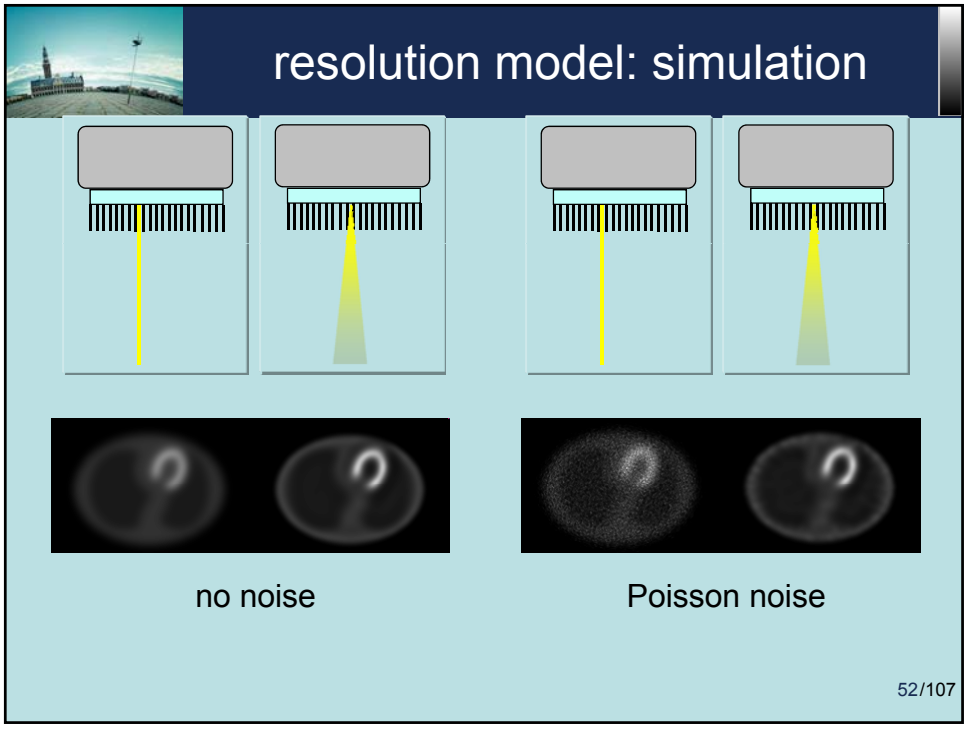
49/107

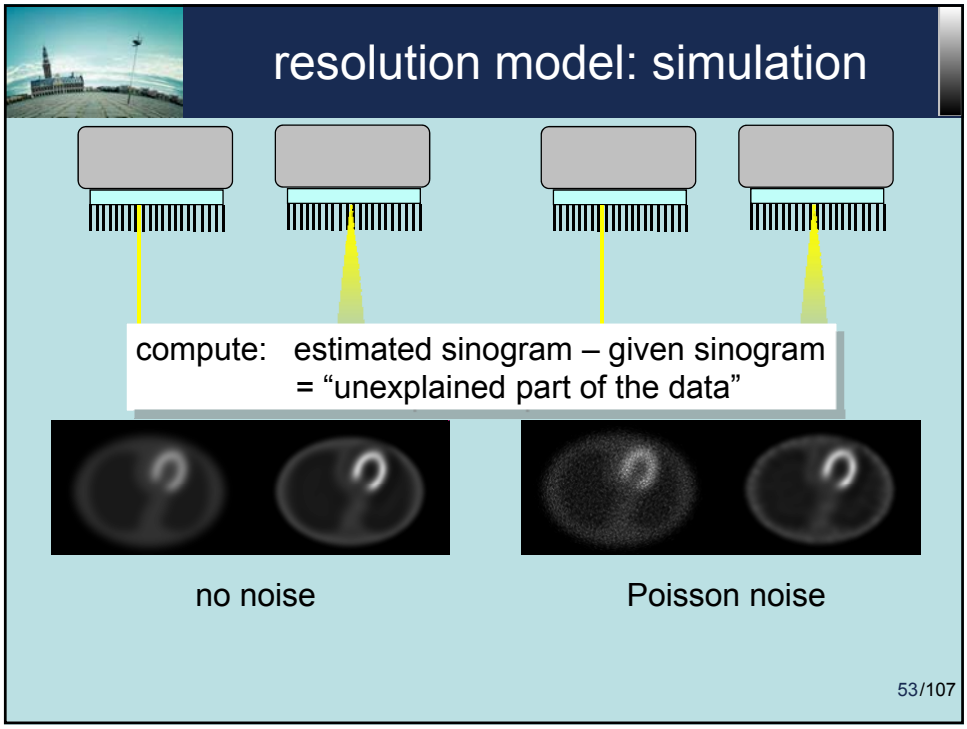
resolution model: simulation

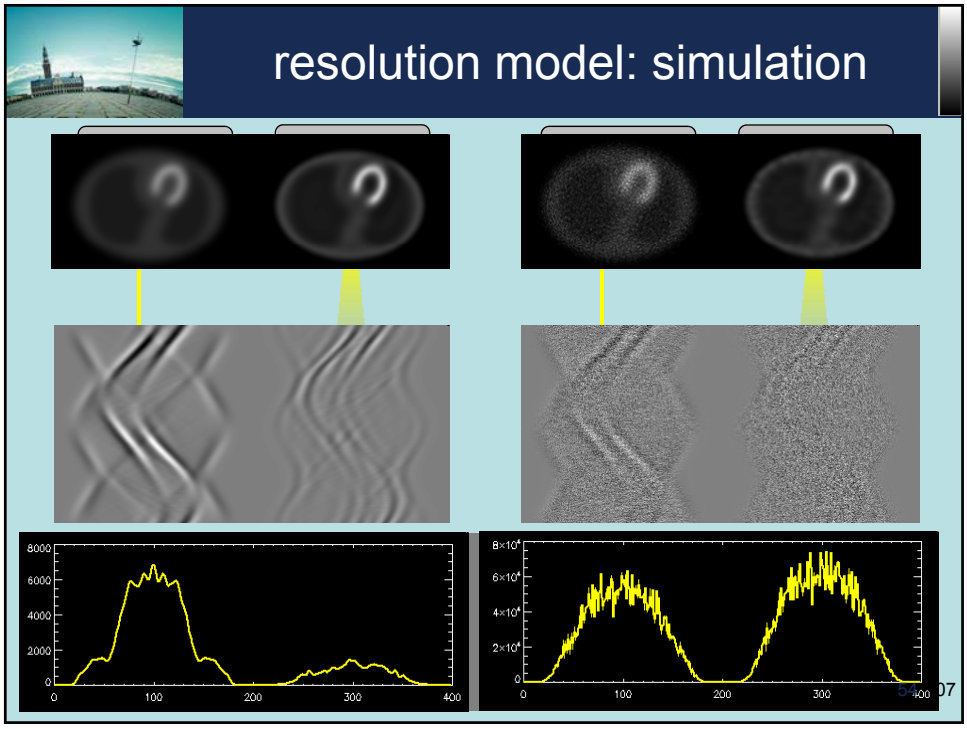


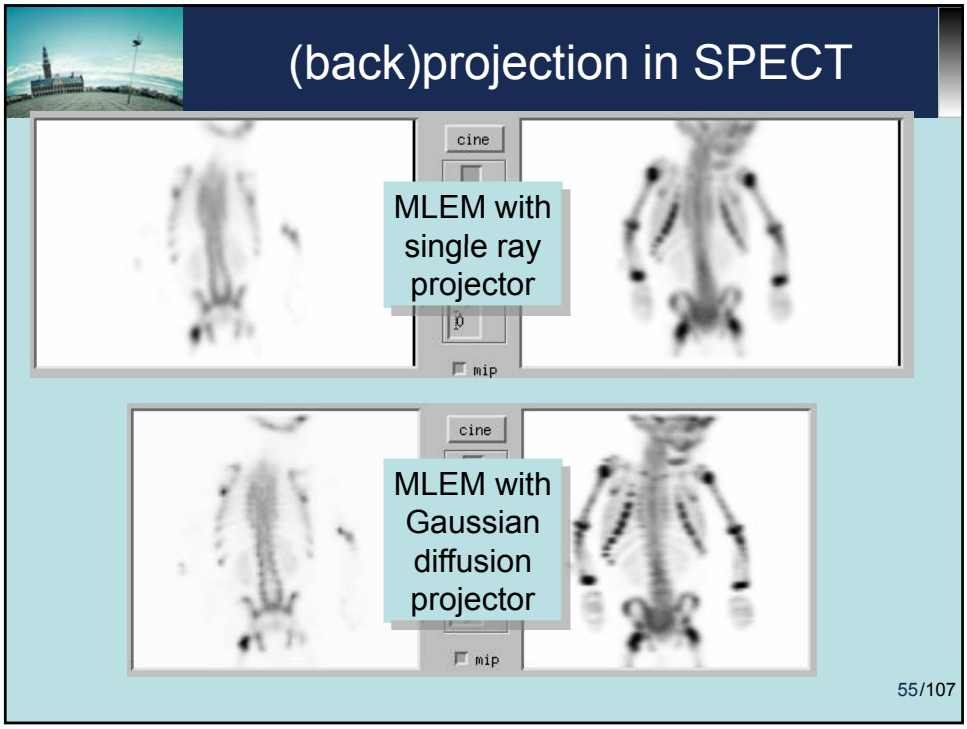
50/107

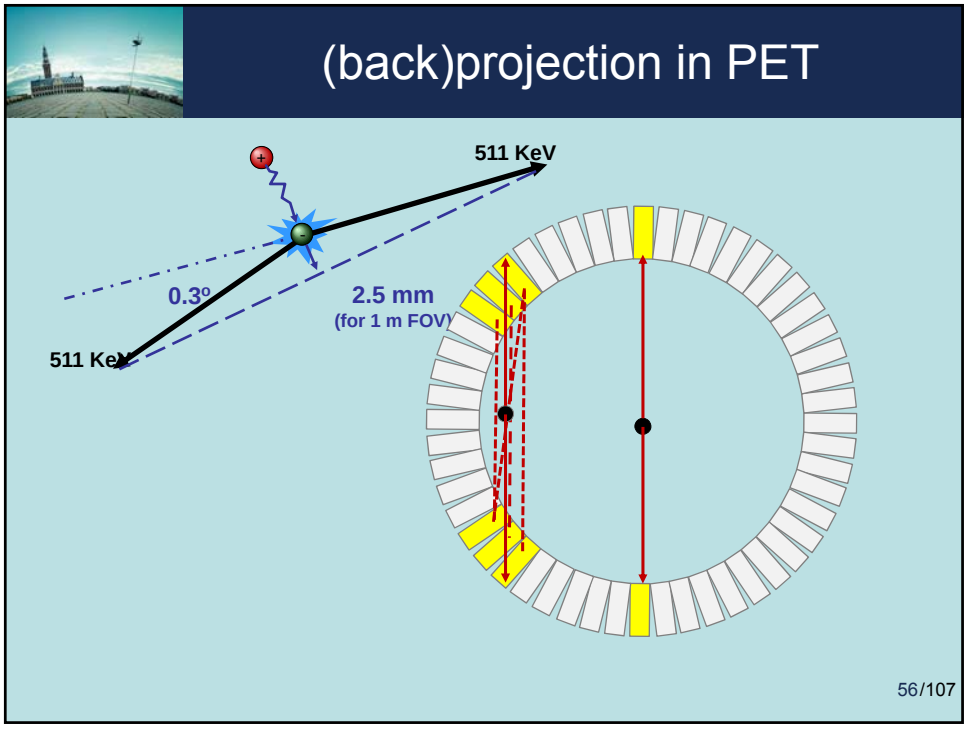

resolution model: simulation
no noise
mlem
Poisson noise
mlem
51/107


resolution model: simulation
no noise
Poisson noise
52/107


resolution model: simulation
compute: estimated sinogram – given sinogram
= “unexplained part of the data”
no noise
Poisson noise
53/107

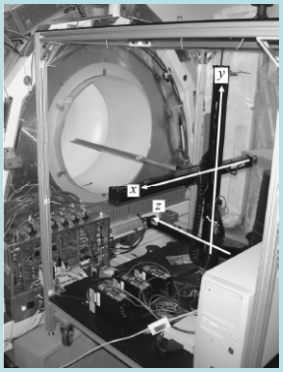

resolution model: simulation
8000
6000
4000
2000
0
0
100
200
300
400
8×10⁴
6×10⁴
4×10⁴
2×10⁴
0
0
100
200
300
400


(back)projection in SPECT
cine
MLEM with single ray projector
mip
cine
MLEM with Gaussian diffusion projector
mip
55/107


(back)projection in PET
511 KeV
0.3°
2.5 mm
(for 1 m FOV)
511 KeV
56/107

(back)projection in PET

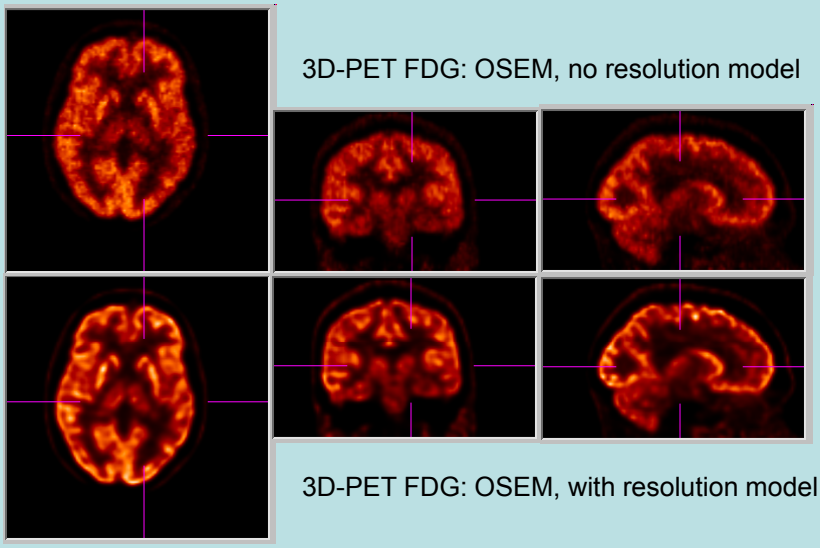
J Qi et al *PMB* 1998,
M Rafecas et al, *IEEE TNS* 2004,
V Panin et al *IEEE TMI* 2006



- measure multiple point source
- fit parametric model
- incorporate in MLEM

57/107

(back)projection in PET

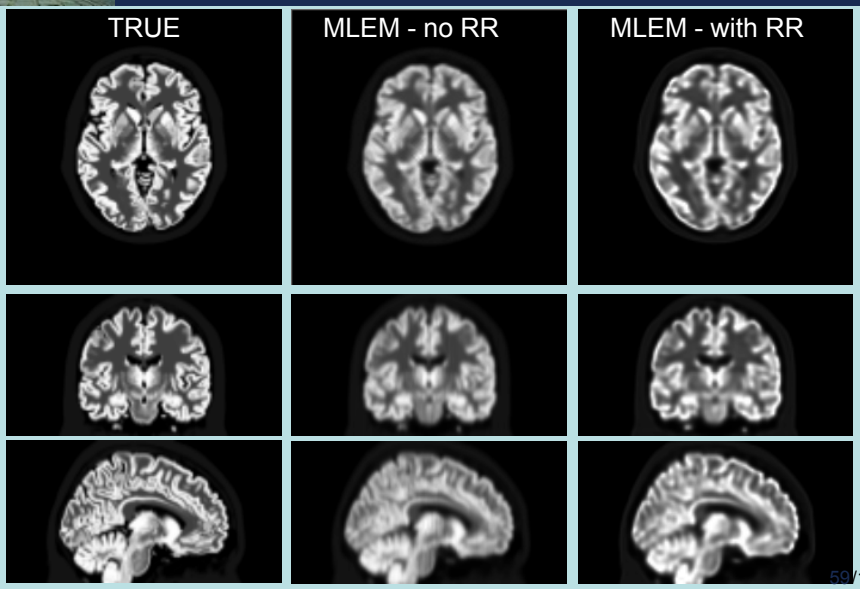


3D-PET FDG: OSEM, no resolution model

3D-PET FDG: OSEM, with resolution model

58/107

(back)projection in PET



59/107

(back) projection model

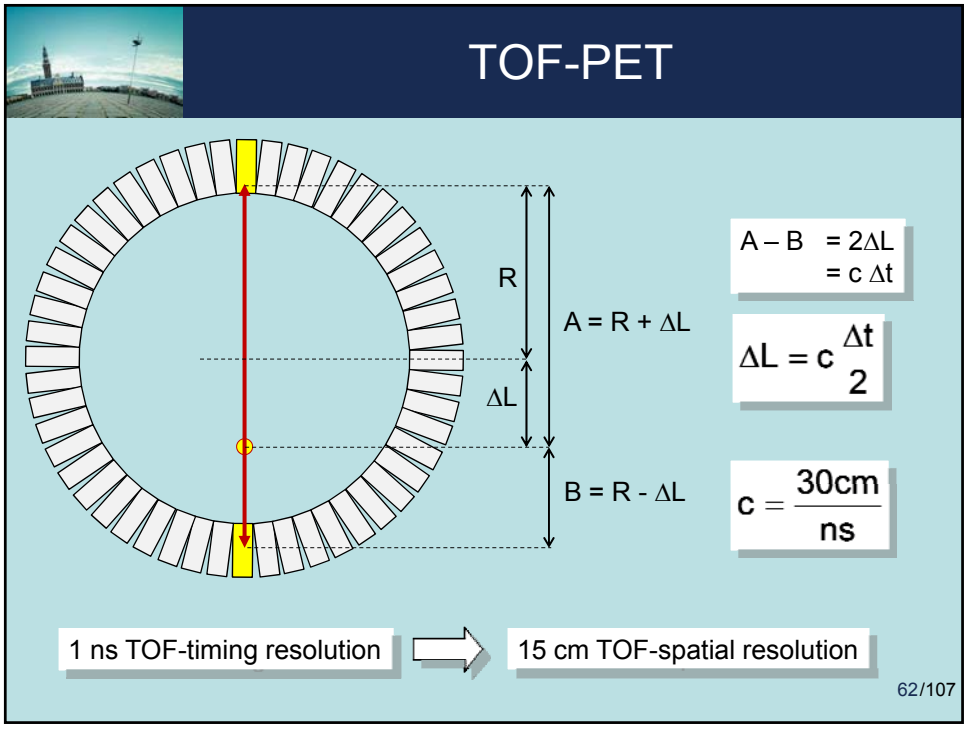
accurate modeling of the physics:

- larger fraction of the data becomes consistent
  ➔ better resolution
- larger fraction of the noise becomes inconsistent
  ➔ less noise

☺ we gain twice!
☹ but computation time goes up...

60/107


TOF-PET
61/107


TOF-PET
R
ΔL
A = R + ΔL
B = R - ΔL
A – B = 2ΔL
= c Δt
ΔL = c Δt/2
c = 30cm/ns
1 ns TOF-timing resolution
15 cm TOF-spatial resolution
62/107

TOF-PET FBP

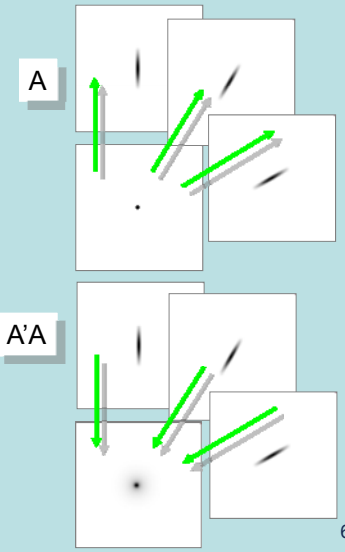
Assume shift invariance and apply LS-reconstruction:

$X = [A' A]^{-1} A' Y$

A is blurring along projection line
A' is blurring along projection line

A'A is 1/|r| x 2D Gauss

$[A'A]^{-1}$ is adjusted ramp filter

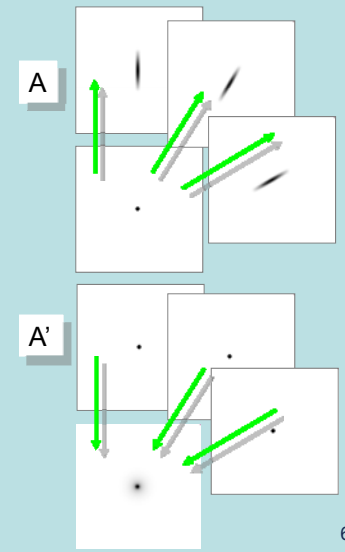


See also: Tomitani, *IEEE TNS* 1981, NS-28; 4582 ff

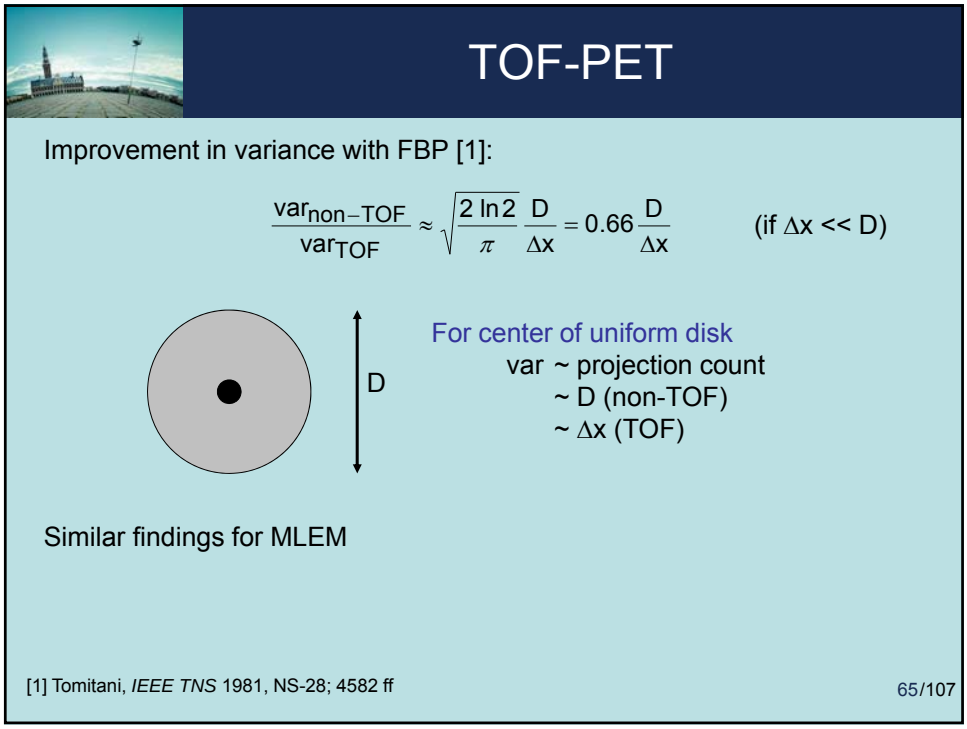
63/107

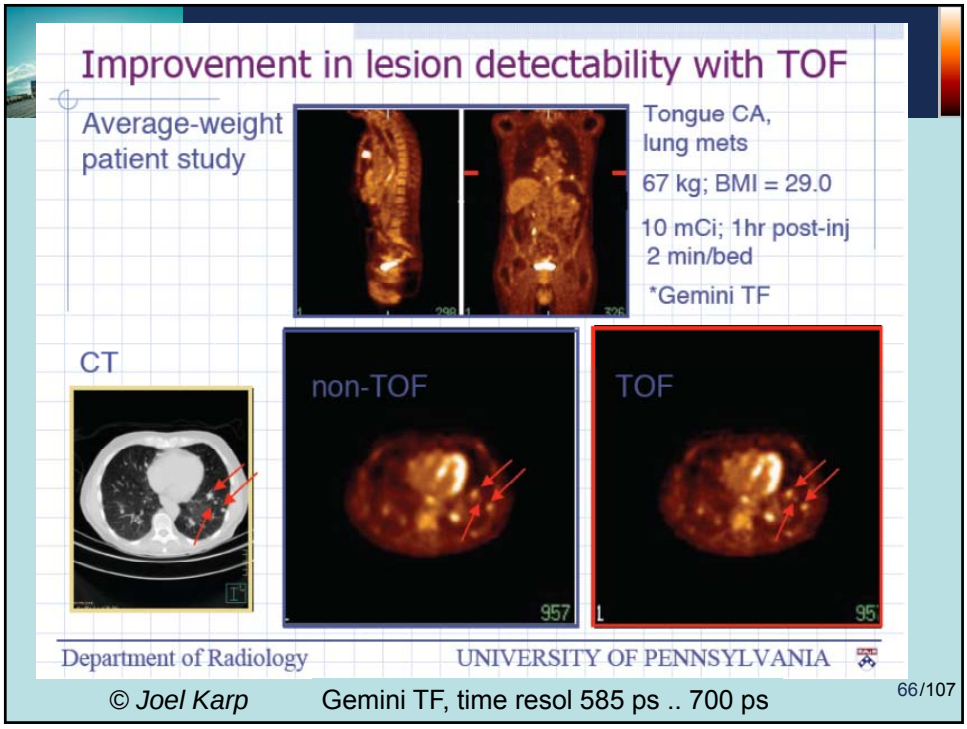
TOF-PET MLEM

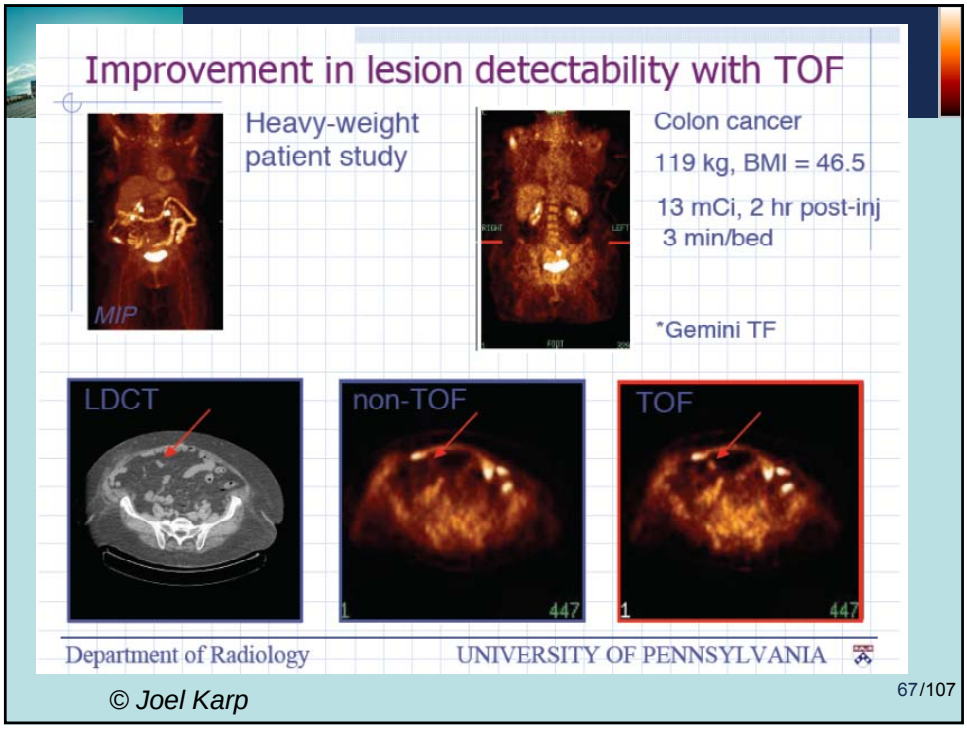
straightforward:
- insert A and A' in mlem
- attenuation same as in non-TOF

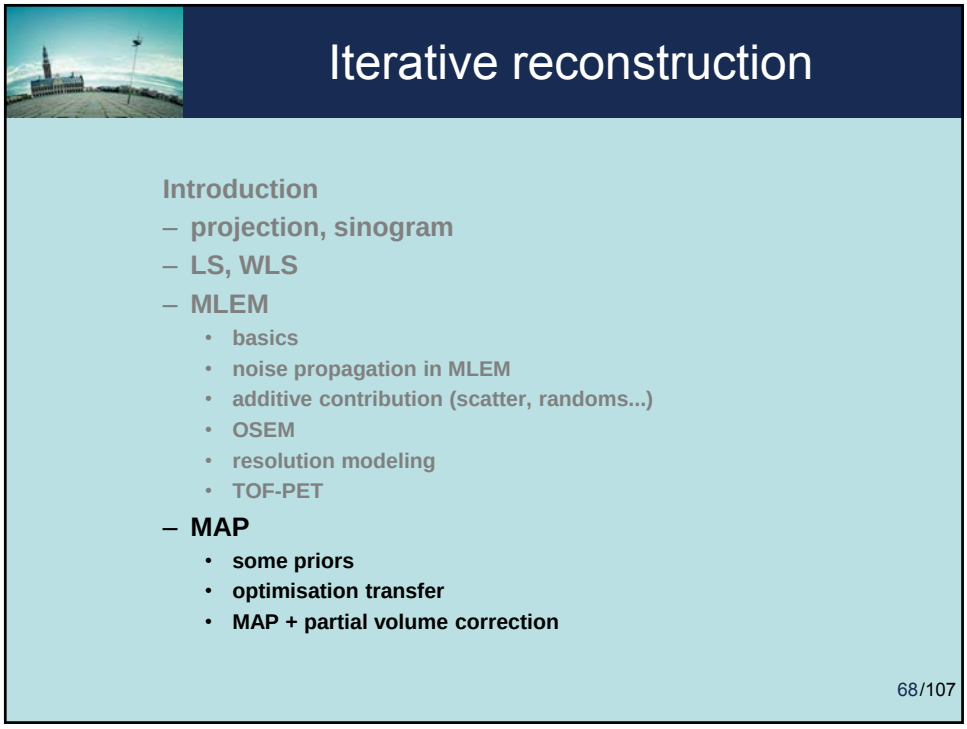


64/107


TOF-PET
Improvement in variance with FBP [1]:
$\frac{var_{non-TOF}}{var_{TOF}} \approx \sqrt{\frac{2\ln 2}{\pi}}\frac{D}{\Delta x} = 0.66\frac{D}{\Delta x}$ (if $\Delta x << D$)
D
For center of uniform disk
var ~ projection count
~ D (non-TOF)
~ Δx (TOF)
Similar findings for MLEM
[1] Tomitani, IEEE TNS 1981, NS-28; 4582 ff
65/107

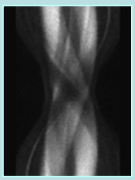

Improvement in lesion detectability with TOF
Average-weight patient study
Tongue CA, lung mets
67 kg; BMI = 29.0
10 mCi; 1hr post-inj
2 min/bed
*Gemini TF
CT
non-TOF
TOF
Department of Radiology
UNIVERSITY OF PENNSYLVANIA
© Joel Karp
Gemini TF, time resol 585 ps .. 700 ps
66/107


Improvement in lesion detectability with TOF
Heavy-weight patient study
MIP
Colon cancer
119 kg, BMI = 46.5
13 mCi, 2 hr post-inj
3 min/bed
*Gemini TF
LDCT
non-TOF
TOF
1
447
1
447
Department of Radiology
UNIVERSITY OF PENNSYLVANIA
© Joel Karp
67/107

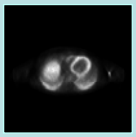

Iterative reconstruction
Introduction
– projection, sinogram
– LS, WLS
– MLEM
• basics
• noise propagation in MLEM
• additive contribution (scatter, randoms...)
• OSEM
• resolution modeling
• TOF-PET
– MAP
• some priors
• optimisation transfer
• MAP + partial volume correction
68/107

MAP

one wishes to find ***recon*** that maximizes *p(**recon** | data)*



data



recon

computing *p(**recon** | data)* difficult inverse problem

computing *p(data | **recon**)* "easy" forward problem

Bayes:

$$p(\textbf{recon} \mid data) \sim \frac{p(data \mid \textbf{recon})\ \cancel{p(\textbf{recon})}}{\cancel{p(data)}}$$

69/107

MAP

likelihood:

- agreement with data

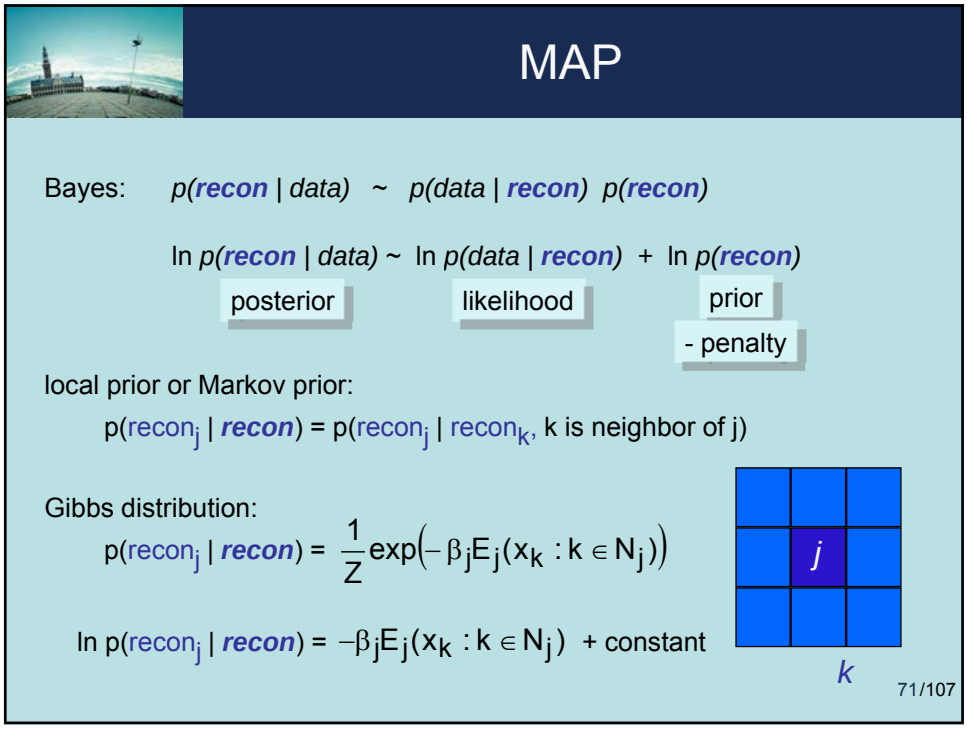
prior:

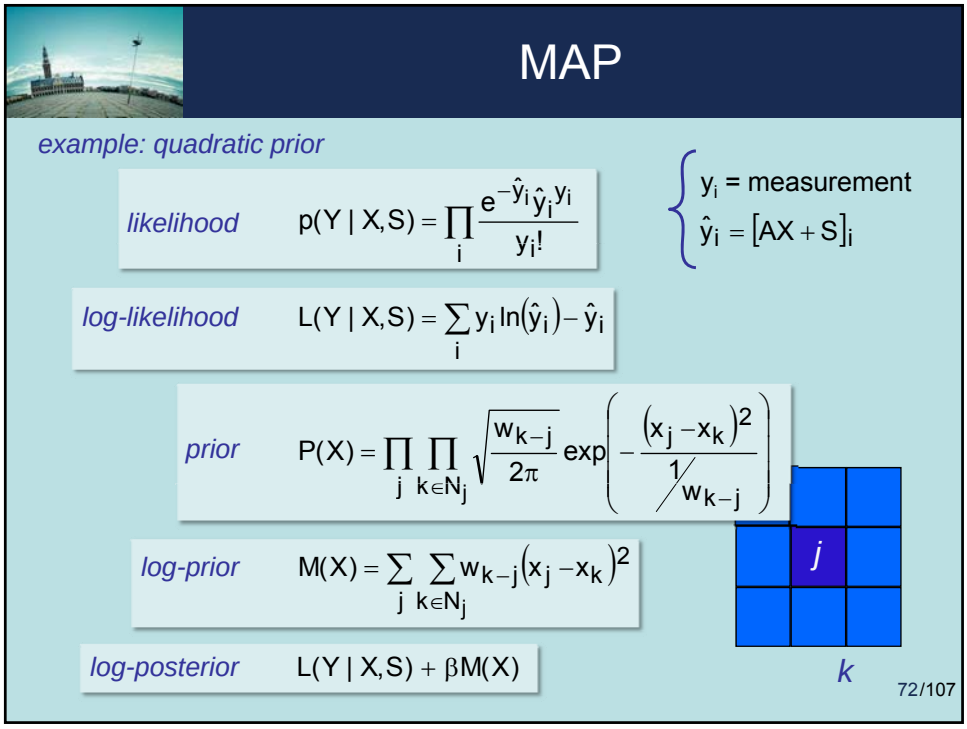
- agreement with prior expectation:
  - image should be smooth
  - image should agree with anatomical information
  - image should ...

challenge:

write the prior as a convenient mathematical function

70/107

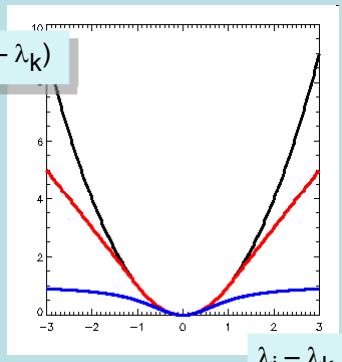

MAP
Bayes: p(recon | data) ~ p(data | recon) p(recon)
ln p(recon | data) ~ ln p(data | recon) + ln p(recon)
posterior
likelihood
prior
- penalty
local prior or Markov prior:
p(recon_j | recon) = p(recon_j | recon_k, k is neighbor of j)
Gibbs distribution:
p(recon_j | recon) = 1/Z exp(−β_j E_j(x_k : k ∈ N_j))
ln p(recon_j | recon) = −β_j E_j(x_k : k ∈ N_j) + constant
j
k
71/107

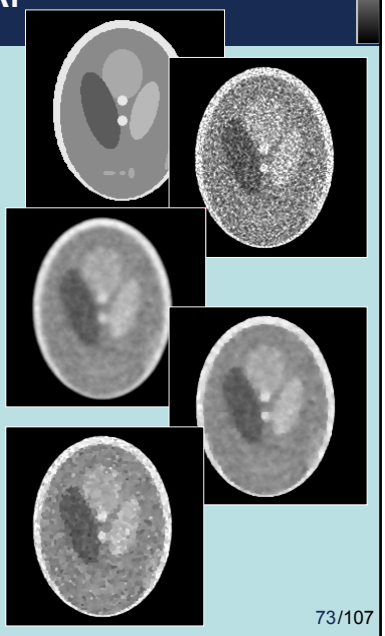

MAP
example: quadratic prior
likelihood
p(Y | X,S) = ∏_i e^{−ŷ_i} ŷ_i^{y_i} / y_i!
y_i = measurement
ŷ_i = [AX + S]_i
log-likelihood
L(Y | X,S) = Σ_i y_i ln(ŷ_i) − ŷ_i
prior
P(X) = ∏_j ∏_{k∈N_j} √(w_{k−j}/2π) exp(−(x_j − x_k)^2 / (1/w_{k−j}))
log-prior
M(X) = Σ_j Σ_{k∈N_j} w_{k−j}(x_j − x_k)^2
log-posterior
L(Y | X,S) + βM(X)
j
k
72/107

MAP

$$\ln p(\text{recon}_j \mid \textit{recon}) = -\beta_j E_j(N_j)$$

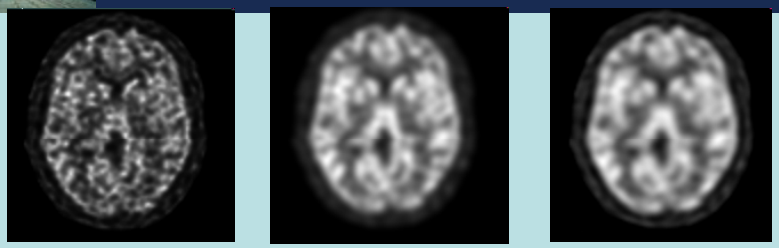
$$= -\sum_{k \in N_j} w_{k-j} M(\lambda_j - \lambda_k)$$



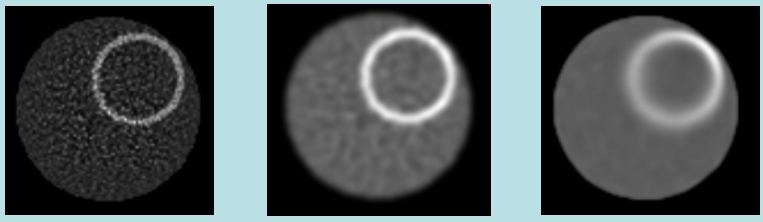


73/107

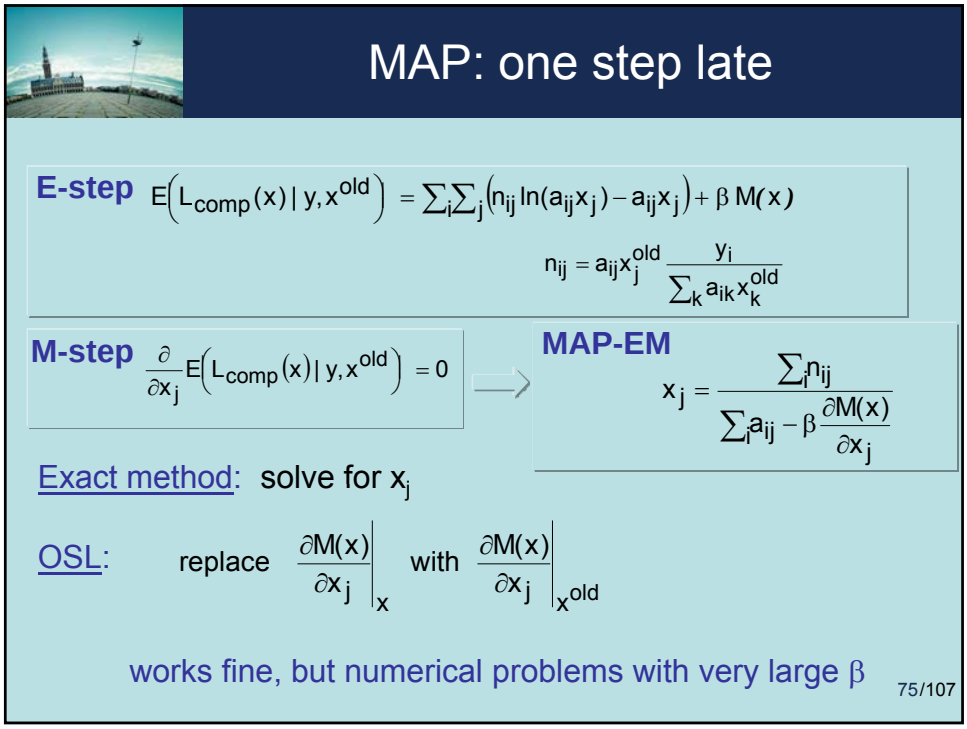
MAP vs smoothed ML

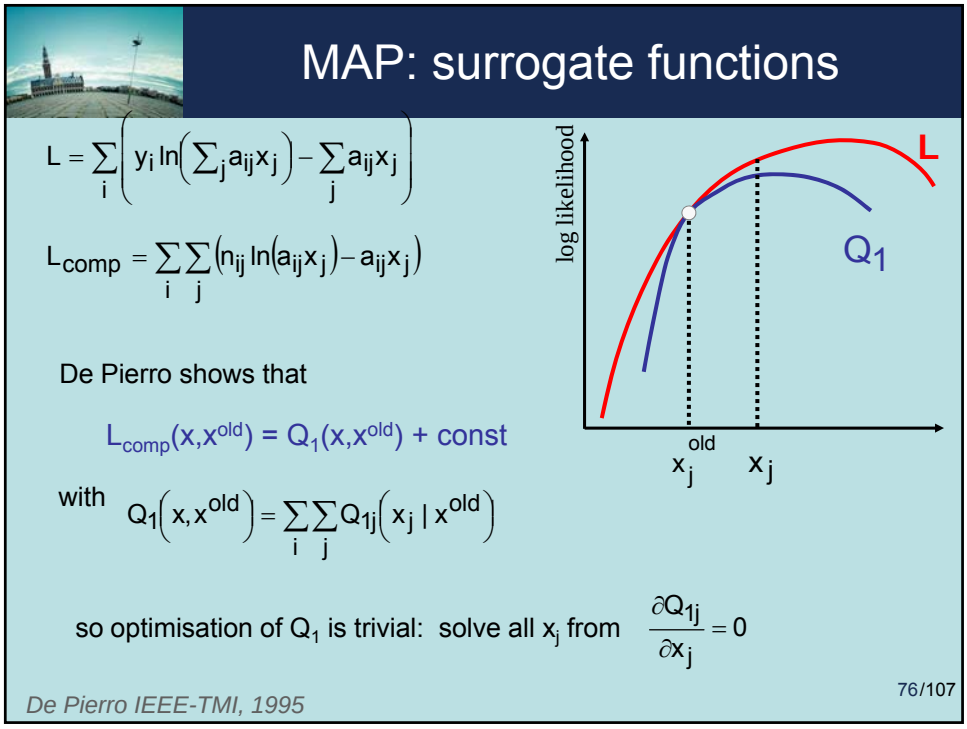


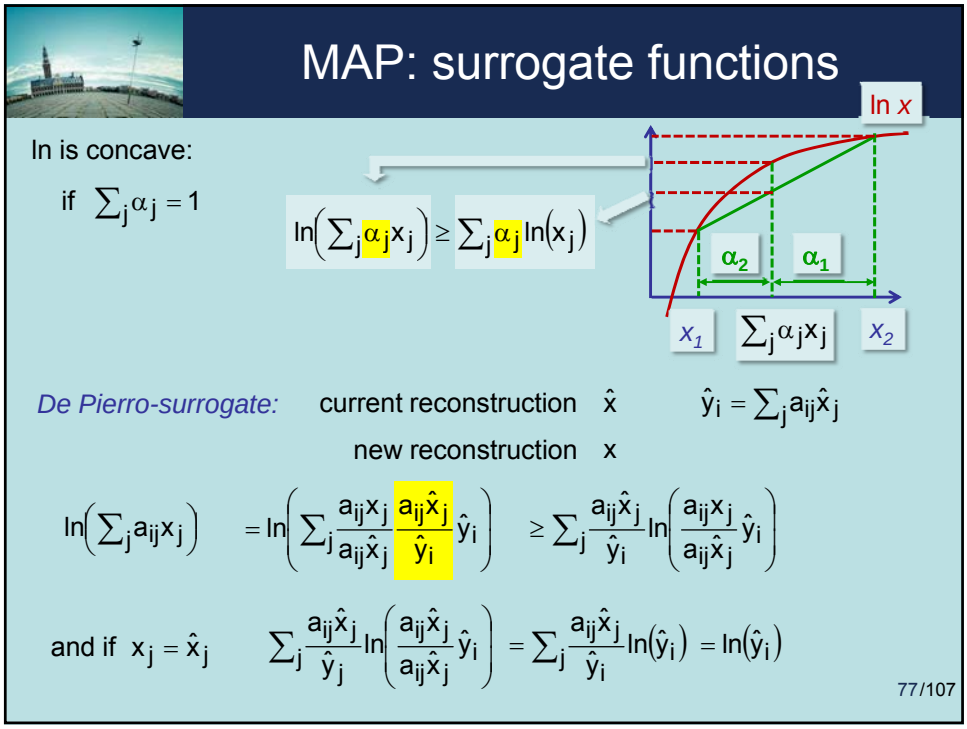
MLEM | smoothed MLEM | MAP with quadratic prior

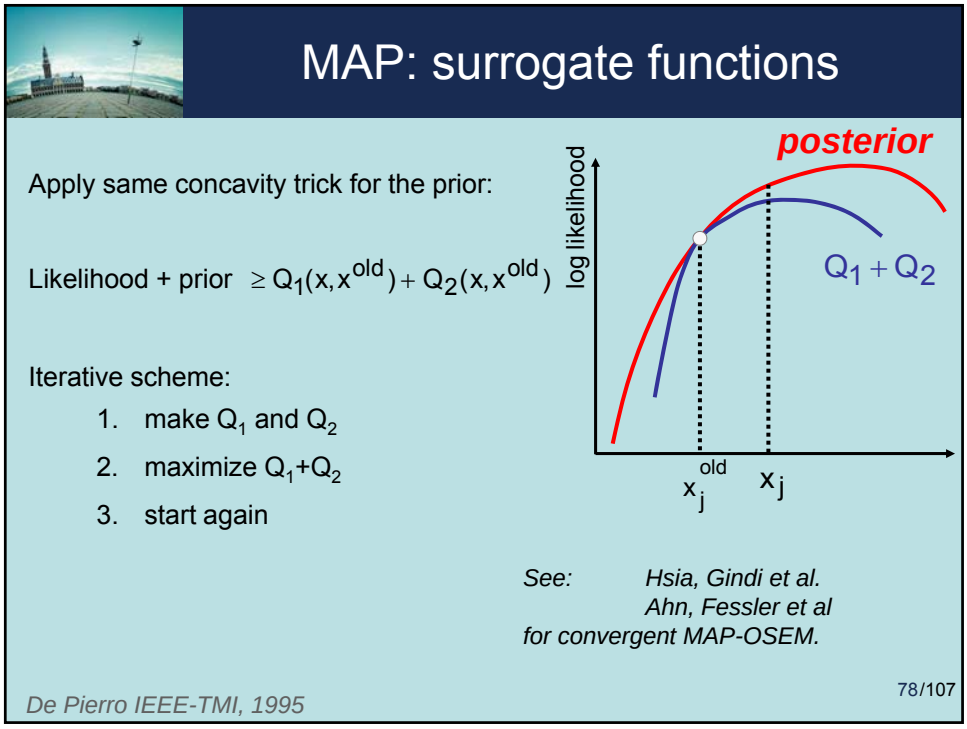


74/107


MAP: one step late
E-step $E\left(L_{comp}(x) \mid y, x^{old}\right) = \sum_i \sum_j \left(n_{ij} \ln(a_{ij}x_j) - a_{ij}x_j\right) + \beta M(x)$
$n_{ij} = a_{ij}x_j^{old} \frac{y_i}{\sum_k a_{ik}x_k^{old}}$
M-step $\frac{\partial}{\partial x_j} E\left(L_{comp}(x) \mid y, x^{old}\right) = 0$
MAP-EM
$x_j = \frac{\sum_i n_{ij}}{\sum_i a_{ij} - \beta \frac{\partial M(x)}{\partial x_j}}$
Exact method: solve for x_j
OSL: replace $\left.\frac{\partial M(x)}{\partial x_j}\right|_x$ with $\left.\frac{\partial M(x)}{\partial x_j}\right|_{x^{old}}$
works fine, but numerical problems with very large β
75/107


MAP: surrogate functions
$L = \sum_i \left(y_i \ln\left(\sum_j a_{ij}x_j\right) - \sum_j a_{ij}x_j \right)$
$L_{comp} = \sum_i \sum_j \left(n_{ij} \ln(a_{ij}x_j) - a_{ij}x_j\right)$
De Pierro shows that
$L_{comp}(x,x^{old}) = Q_1(x,x^{old}) + const$
with $Q_1\left(x, x^{old}\right) = \sum_i \sum_j Q_{1j}\left(x_j \mid x^{old}\right)$
so optimisation of Q_1 is trivial: solve all x_j from $\frac{\partial Q_{1j}}{\partial x_j} = 0$
log likelihood
L
Q_1
old
x_j
x_j
De Pierro IEEE-TMI, 1995
76/107


MAP: surrogate functions
ln x
ln is concave:
if $\sum_j \alpha_j = 1$
$\ln\left(\sum_j \alpha_j x_j\right) \geq \sum_j \alpha_j \ln(x_j)$
α_2
α_1
x_1
$\sum_j \alpha_j x_j$
x_2
De Pierro-surrogate:
current reconstruction $\hat{x}$
$\hat{y}_i = \sum_j a_{ij}\hat{x}_j$
new reconstruction x
$\ln\left(\sum_j a_{ij}x_j\right) = \ln\left(\sum_j \frac{a_{ij}x_j}{a_{ij}\hat{x}_j}\frac{a_{ij}\hat{x}_j}{\hat{y}_i}\hat{y}_i\right) \geq \sum_j \frac{a_{ij}\hat{x}_j}{\hat{y}_i}\ln\left(\frac{a_{ij}x_j}{a_{ij}\hat{x}_j}\hat{y}_i\right)$
and if $x_j = \hat{x}_j$
$\sum_j \frac{a_{ij}\hat{x}_j}{\hat{y}_j}\ln\left(\frac{a_{ij}\hat{x}_j}{a_{ij}\hat{x}_j}\hat{y}_i\right) = \sum_j \frac{a_{ij}\hat{x}_j}{\hat{y}_i}\ln(\hat{y}_i) = \ln(\hat{y}_i)$
77/107


MAP: surrogate functions
Apply same concavity trick for the prior:
Likelihood + prior $\geq Q_1(x, x^{old}) + Q_2(x, x^{old})$
Iterative scheme:
1. make Q_1 and Q_2
2. maximize Q_1+Q_2
3. start again
posterior
log likelihood
$Q_1 + Q_2$
x_j^{old}
x_j
See: Hsia, Gindi et al.
Ahn, Fessler et al
for convergent MAP-OSEM.
De Pierro IEEE-TMI, 1995
78/107

MAP: surrogate functions

surrogate for the prior: $M(x) = \sum_{jk} w_{jk} F(x_j, x_k)$

- $w_{jk} = w_{kj}$
- $F(x_j,x_k) = F(x_k,x_j)$
- F is concave

$$M(x) = M(\hat{x} + \Delta x) \quad \text{with} \quad \Delta x = x - \hat{x}$$

$$= \sum_{jk} w_{jk} F(\hat{x}_j + \Delta x_j, \hat{x}_k + \Delta x_k)$$

$$= \sum_{jk} w_{jk} F\left(\frac{1}{2}(\hat{x}_j + 2\Delta x_j, \hat{x}_k) + \frac{1}{2}(\hat{x}_j, \hat{x}_k + 2\Delta x_k)\right)$$

$$\geq \sum_{jk} \frac{w_{jk}}{2} F(\hat{x}_j + 2\Delta x_j, \hat{x}_k) + \sum_{jk} \frac{w_{jk}}{2} F(\hat{x}_j, \hat{x}_k + 2\Delta x_k)$$

$$= \sum_{jk} w_{jk} F(\hat{x}_j + 2\Delta x_j, \hat{x}_k)$$

$$= \sum_{jk} w_{jk} F(2x_j - \hat{x}_j, \hat{x}_k) \qquad = Q_2(x, \hat{x})$$

79/107

MAP: surrogate functions

example: MAP with quadratic prior

$$\varphi_{ij} = \frac{a_{ij}\hat{x}_j}{\hat{y}_i}$$

likelihood

$$L(x \mid y) = \sum_i y_i \ln\left(\sum_j a_{ij} x_j\right) - \sum_j a_{ij} x_j \geq \sum_i y_i \sum_j \varphi_{ij} \ln\left(\frac{a_{ij} x_j}{\varphi_{ij}}\right) - a_{ij} x_j = \sum_j Q_{1j}(x_j)$$

prior

$$M(x) = -\sum_{jk} w_{jk} (x_j - x_k)^2 \geq -\sum_{jk} w_{jk} (2x_j - \hat{x}_j - \hat{x}_k)^2 = \sum_j Q_{2j}(x_j)$$

posterior

increase L + βM by maximizing Q1 + Q2: (we can absorb β in w_{jk})

$$\frac{\partial(Q_{1j}(x_j) + Q_{2j}(x_j))}{\partial x_j} = 0$$

80/107

MAP: surrogate functions

$$\frac{\partial\left(Q_{1j}(x_j)+Q_{2j}(x_j)\right)}{\partial x_j}=0$$

81/107

MAP: surrogate functions

$$\frac{\partial}{\partial x_j}\left(\sum_i y_i \sum_j \varphi_{ij} \ln\left(\frac{a_{ij}x_j}{\varphi_{ij}}\right) - a_{ij}x_j - \sum_{jk} w_{jk}(2x_j - \hat{x}_j - \hat{x}_k)^2\right)=0$$

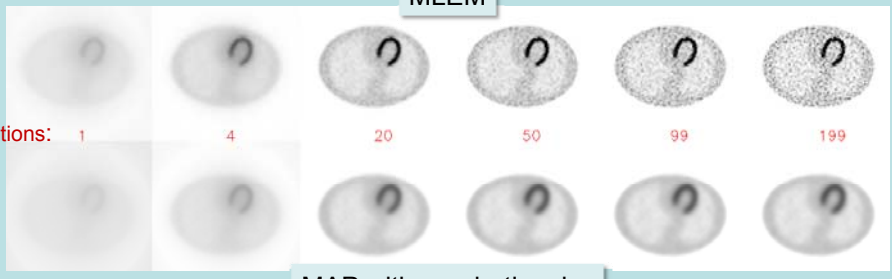
$$\left(8\sum_k w_{jk}\right)^2 x_j^2 + \left(\sum_i a_{ij} - 4\sum_k w_{jk}(\hat{x}_j+\hat{x}_k)\right)x_j - \sum_i \varphi_{ij} y_i = 0$$

$$\begin{cases} A = 16\sum_k w_{jk} \\ B = 4\sum_k w_{jk}(\hat{x}_j+\hat{x}_k) - \sum_i a_{ij} \\ C = 32\left(\sum_k w_{jk}\right)\left(\sum_i a_{ij}\frac{y_i}{\hat{y}_i}\right)\hat{x}_j \end{cases} \qquad x_j = \frac{B+\sqrt{B^2+C}}{A}$$

82/107

MAP: surrogate functions

MLEM



MAP with quadratic prior

83/107

prior behavior

- the effect of the likelihood is complex:
  - shift variant
  - orientation dependent
  - object dependent

- when combined with prior, things get even more complicated

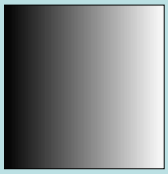
- study prior
  - without likelihood
  - with very simple likelihood

84/107

prior behavior

without likelihood

choose a initial image
examples:
- uniform random noise
- ramp



optimize prior (here with gradient ascent algorithm)

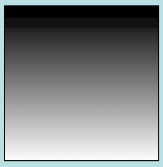
$$x_j = x_j^{old} + step(x)\left.\frac{\partial M(x)}{\partial x_j}\right|_{x^{old}}$$

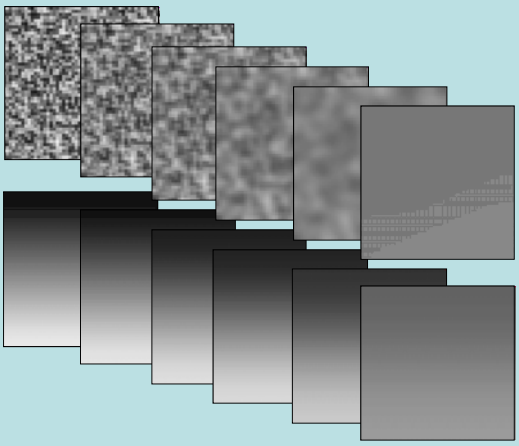
85/107

prior behavior

Apply quadratic prior, without likelihood

$$M(x) = -\beta \sum_{j\,k} w_{j-k}(x_j - x_k)^2$$





86/107

prior behavior

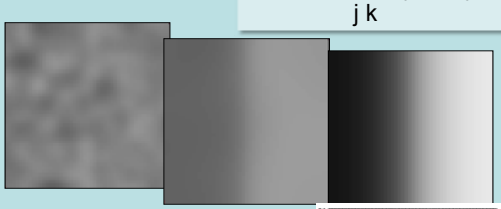
combine quadratic prior with simple likelihood, based on reference image, e.g. step function

ref image

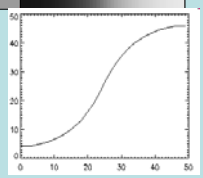
likelihood: $L(x \mid x^{ref}) = -\sum_j \left(x_j - x_j^{ref} \right)^2$

prior: $M(x) = -\beta \sum_{j\,k} w_{j-k} (x_j - x_k)^2$

initial image:



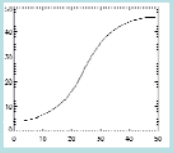
MAP optimization



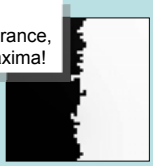
87/107

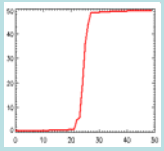
prior behavior

quadratic prior

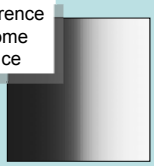


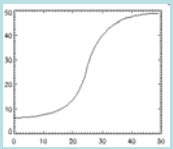
Geman prior, strong edge tolerance, but has local maxima!



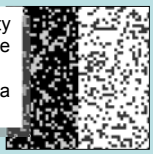


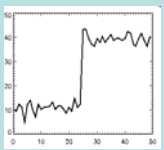
relative difference prior, with some edge tolerance



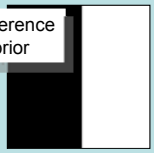


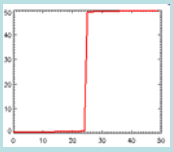
absolute intensity prior, favoring the values 0 and 1, has local maxima too!





absolute difference (= median) prior





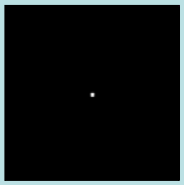
Chance of getting stuck in a bad local maximum can be reduced by gradually increasing the strength of the prior

88/107

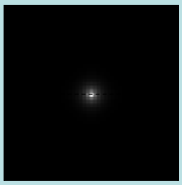
prior behavior

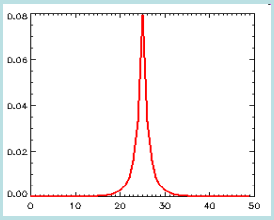
impulse response of quadratic prior:

reference image



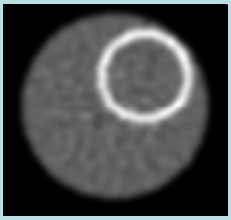
MAP image

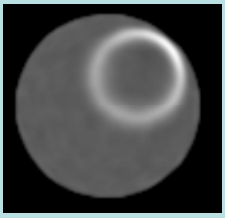




89/107

Shift-variant "prior" for shift-invariant resolution





There is strong evidence that:
if MAP and post-smoothed MLEM have same PSF,
then they have also the same (co)variance.

The "prior" (or penalty) should smooth more
when the likelihood is stronger!

90/107

Shift-variant "prior" for shift-invariant resolution

This was seen experimentally [1,2] for MAP.
It can be computed (with matrices) for WLS with a quadratic prior.

$$M(x) = -\beta \sum_{j\,k} w_{j-k}(x_j - x_k)^2 \quad = -\beta \sum_{j\,k} w_{j-k}(x_j^2 + x_k^2 - 2x_j x_k)$$

$$= \beta X'RX = \beta \sum_{j\,k} r_{jk} x_j x_k \qquad r_{jj} = -\sum_k (w_{j-k} + w_{k-j})$$

$$r_{jk} = w_{j-k} + w_{k-j} \quad \text{for } j \neq k$$

$$L = (Y - AX)'C_Y^{-1}(Y - AX) + \beta X'RX$$

solution: $\frac{\partial L}{\partial x_j} = 0 \Rightarrow X = \left(A'C_Y^{-1}A + \beta R\right)^{-1} A'C_Y^{-1}Y$

JW Stayman, JA Fessler, "Compensation for nonuniform resolution using penalized-likelihood reconstruction in space-variant imaging systems," *IEEE Trans Med Imaging.* 2004
JA Fessler, "Analytical approach to regularization design for isotropic spatial resolution". *IEEE NSS MIC* 2003.

91/107

Shift-variant "prior" for shift-invariant resolution

WLS + prior: $L = (Y - AX)'C_Y^{-1}(Y - AX) + \beta X'RX$

MAP_{wls} solution: $X = \left(A'C_Y^{-1}A + \beta R\right)^{-1} A'C_Y^{-1}Y \quad = \Psi Y$

MAP_{wls} PSF: $Y = A\breve{X} \Rightarrow X = \Psi A\breve{X}$

covar of MAP_{wls}: $C_X = E\left\{(X - \bar{X})(X - \bar{X})'\right\} \quad = E\left\{\Psi(Y - \bar{Y})(Y - \bar{Y})'\Psi'\right\}$

$$= \Psi E\left\{(Y - \bar{Y})(Y - \bar{Y})'\right\}\Psi' \quad = \Psi C_Y \Psi'$$

$$= \left(A'C_Y^{-1}A + \beta R\right)^{-1} A'C_Y^{-1} C_Y C_Y^{-1} A \left(A'C_Y^{-1}A + \beta R\right)^{-1}$$

$$= \left(A'C_Y^{-1}A + \beta R\right)^{-1} A'C_Y^{-1} A \left(A'C_Y^{-1}A + \beta R\right)^{-1}$$

92/107

Shift-variant "prior" for shift-invariant resolution

PSF of MAP_{wls} $\quad X_{map} = \left(A'C_Y^{-1}A + \beta R\right)^{-1} A'C_Y^{-1}A\breve{X}$

covar of MAP_{wls} $\quad C_{X_{map}} = \left(A'C_Y^{-1}A + \beta R\right)^{-1} A'C_Y^{-1}A\left(A'C_Y^{-1}A + \beta R\right)^{-1}$

$\beta = 0$

PSF of ML_{WLS} $\quad X_{ml} = \left(A'C_Y^{-1}A\right)^{-1} A'C_Y^{-1}A\breve{X} \quad = \breve{X}$

covar of ML_{wls} $\quad C_{X_{ml}} = \left(A'C_Y^{-1}A\right)^{-1} A'C_Y^{-1}A\left(A'C_Y^{-1}A\right)^{-1} = \left(A'C_Y^{-1}A\right)^{-1}$

post-smoothed ML_{WLS} filter P $\quad X_{pml} = P\breve{X}$

$$C_{X_{pml}} = P\left(A'C_Y^{-1}A\right)^{-1}P'$$

93/107

Shift-variant "prior" for shift-invariant resolution

MAP_{wls} $\quad X_{map} = \left(A'C_Y^{-1}A + \beta R\right)^{-1} A'C_Y^{-1}A\breve{X}$

$$C_{X_{map}} = \left(A'C_Y^{-1}A + \beta R\right)^{-1} A'C_Y^{-1}A\left(A'C_Y^{-1}A + \beta R\right)^{-1}$$

post-smoothed ML_{WLS} $\quad X_{pml} = P\breve{X}$

$$C_{X_{pml}} = P\left(A'C_Y^{-1}A\right)^{-1}P'$$

Design prior matrix R such that we have a shift-invariant PSF P

$X_{map} = \left(A'C_Y^{-1}A + \beta R\right)^{-1} A'C_Y^{-1}A\,\breve{X} \quad = P\,\breve{X}$

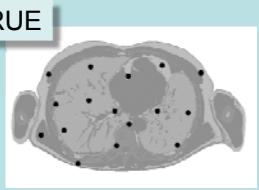
$C_{X_{map}} = \left(A'C_Y^{-1}A + \beta R\right)^{-1} A'C_Y^{-1}A\left(A'C_Y^{-1}A + \beta R\right)^{-1}$

$= P\left(A'C_Y^{-1}A\right)^{-1}P' \quad = C_{X_{pml}}$

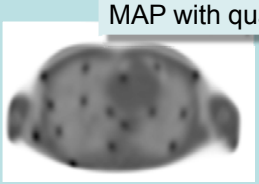
94/107

Shift invariant prior for detection

TRUE



MAP with quad.prior



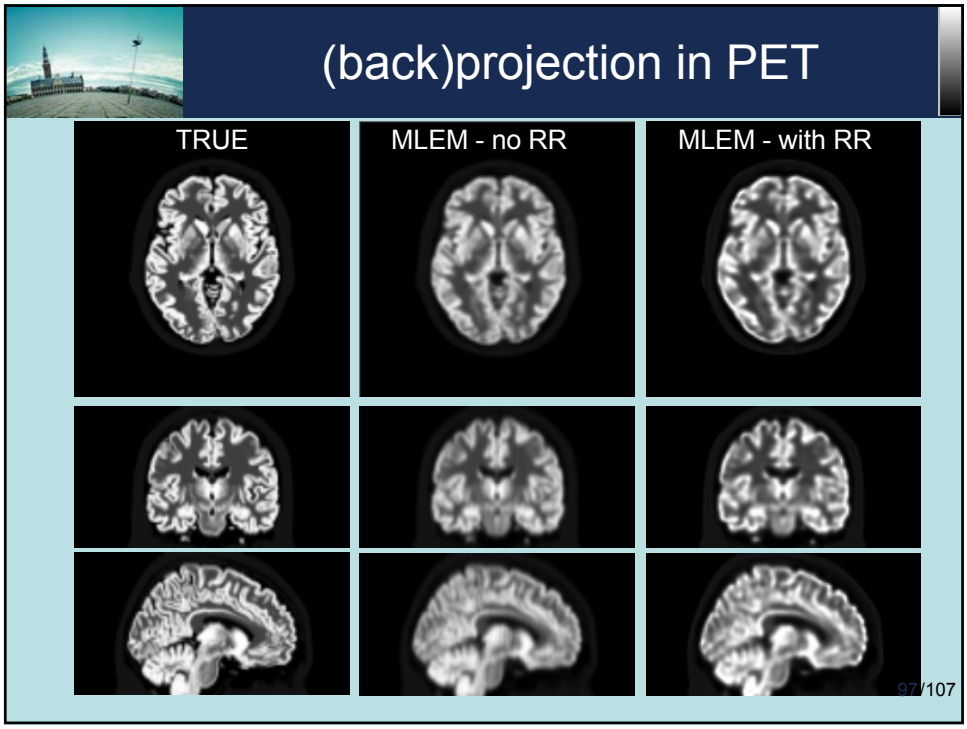
There is evidence that
MAP with quadratic prior (shift variant resolution)
is better for detection than
MAP with a prior for shift invariant resolution
≈ post-smoothed MLEM with same PSF

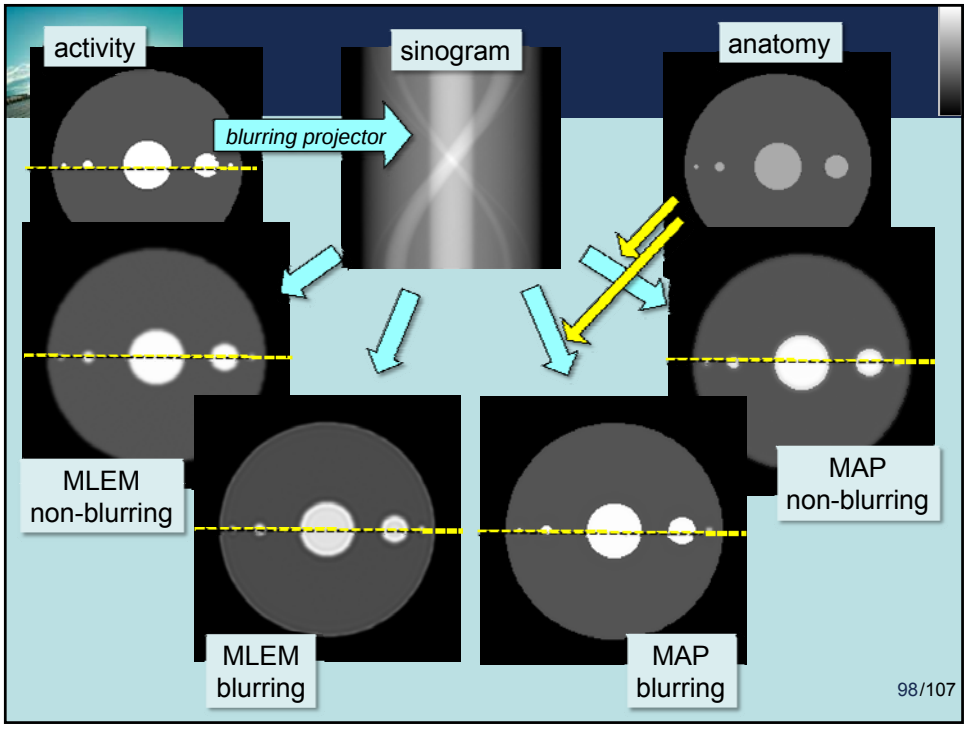
J Qi, R Huesman. "Penalized maximum-likelihood image reconstruction for lesion detection" *Phys Med Biol* 2006.
J Nuyts, C Michel et al. "Performance of MAP Reconstruction for Hot Lesion Detection in Whole-body PET/CT: an Evaluation with Human and Numerical Observers". *IEEE Trans Med Imaging*, 2009

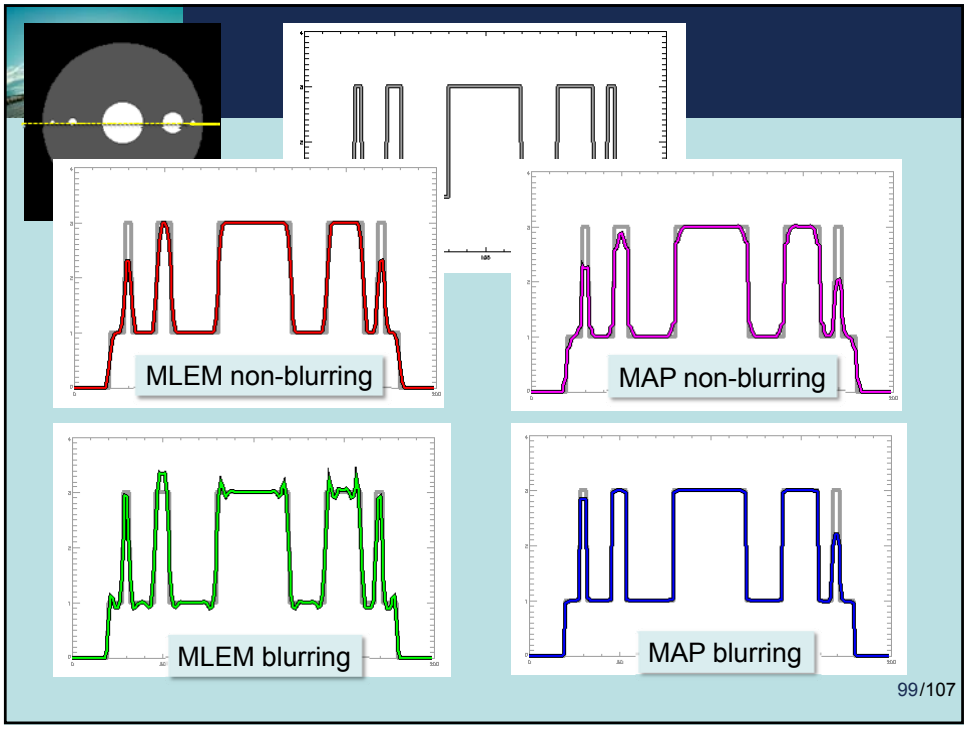
95/107

Partial volume correction
+
MAP

96/107

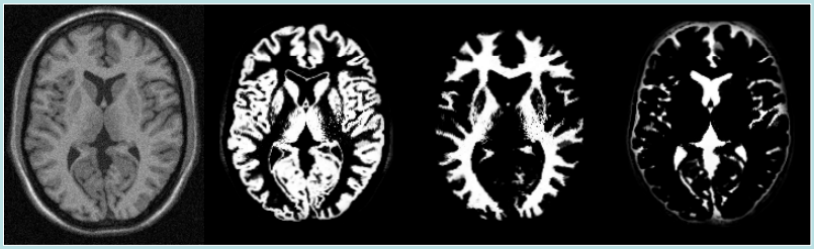

(back)projection in PET
TRUE
MLEM - no RR
MLEM - with RR
97/107


activity
sinogram
anatomy
blurring projector
MLEM
non-blurring
MLEM
blurring
MAP
blurring
MAP
non-blurring
98/107



MAP with anatomical prior

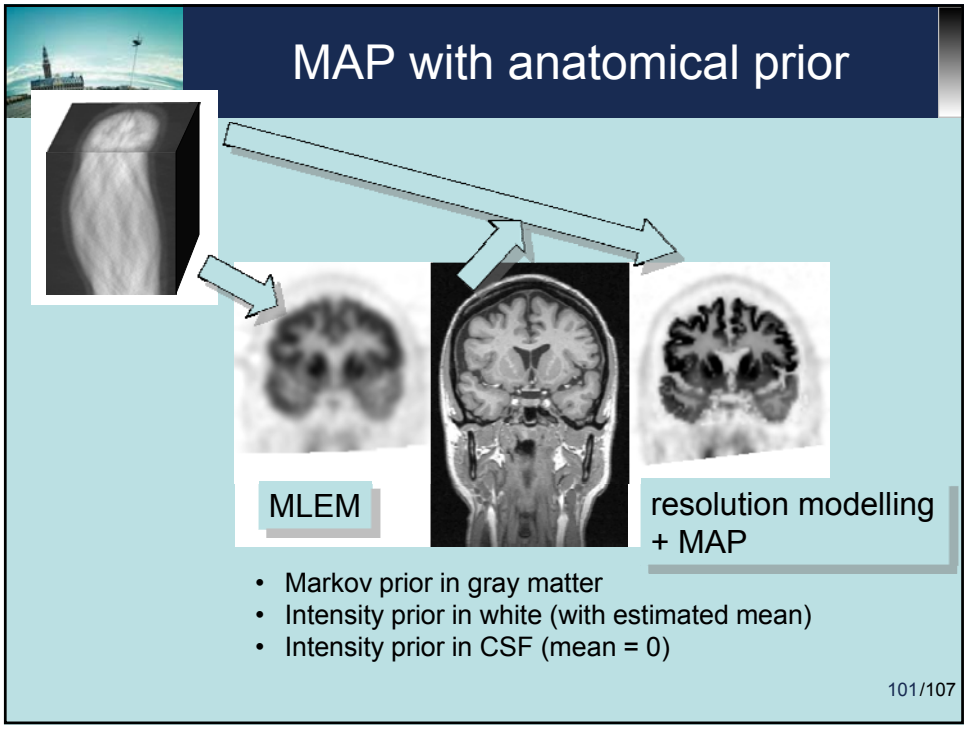
T1 **Grey** **White** **CSF**

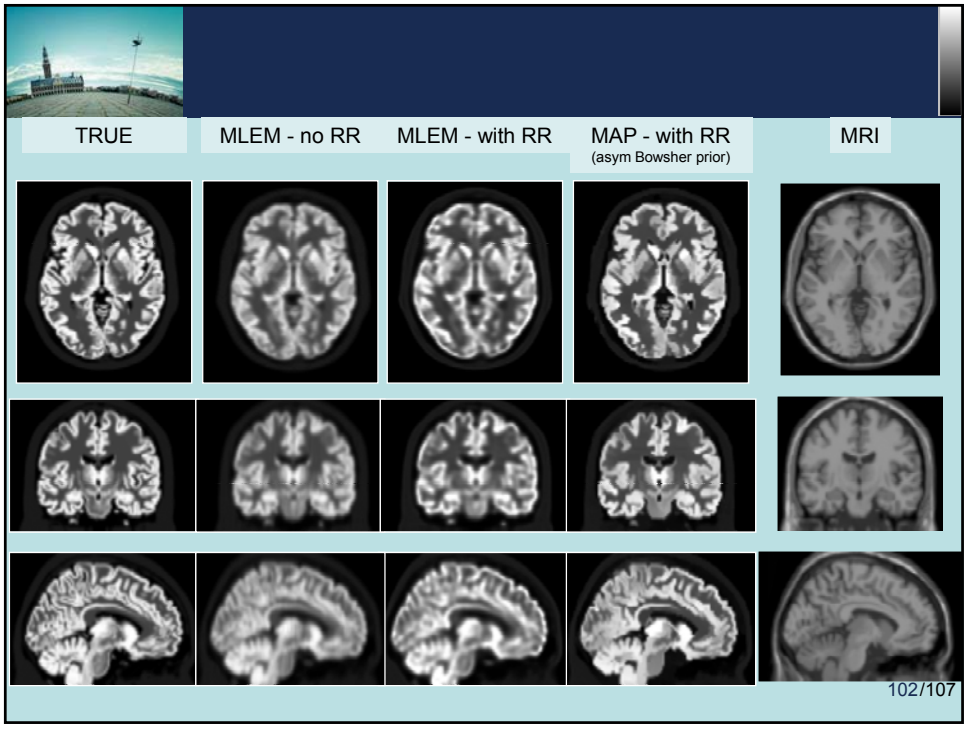


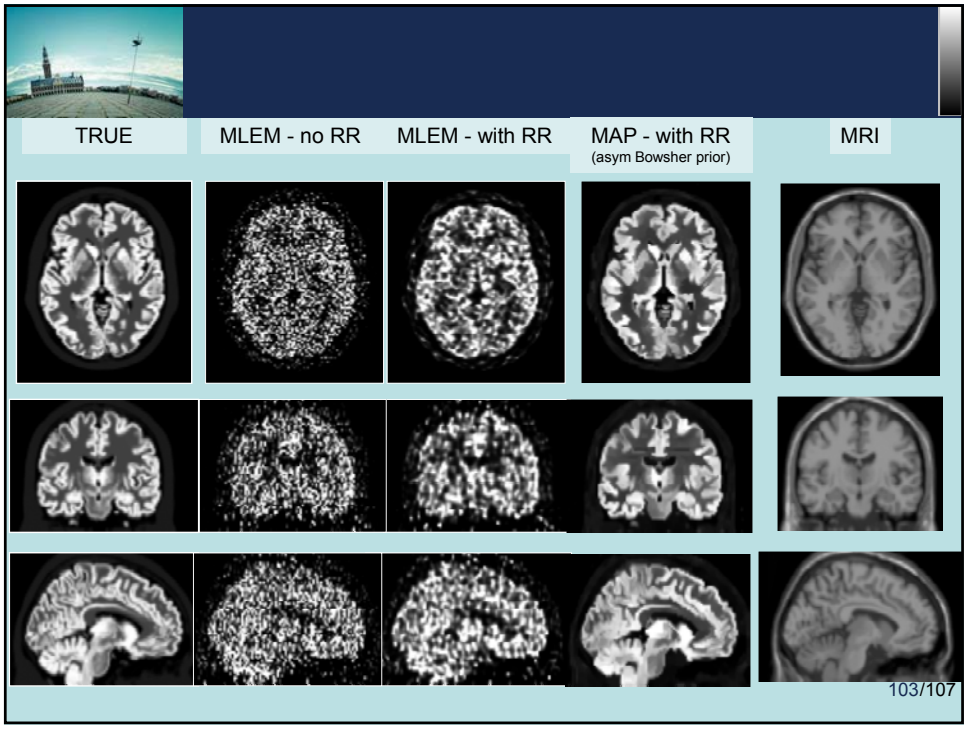
prior knowledge, valid for several tracer (FDG, ECD, ...)

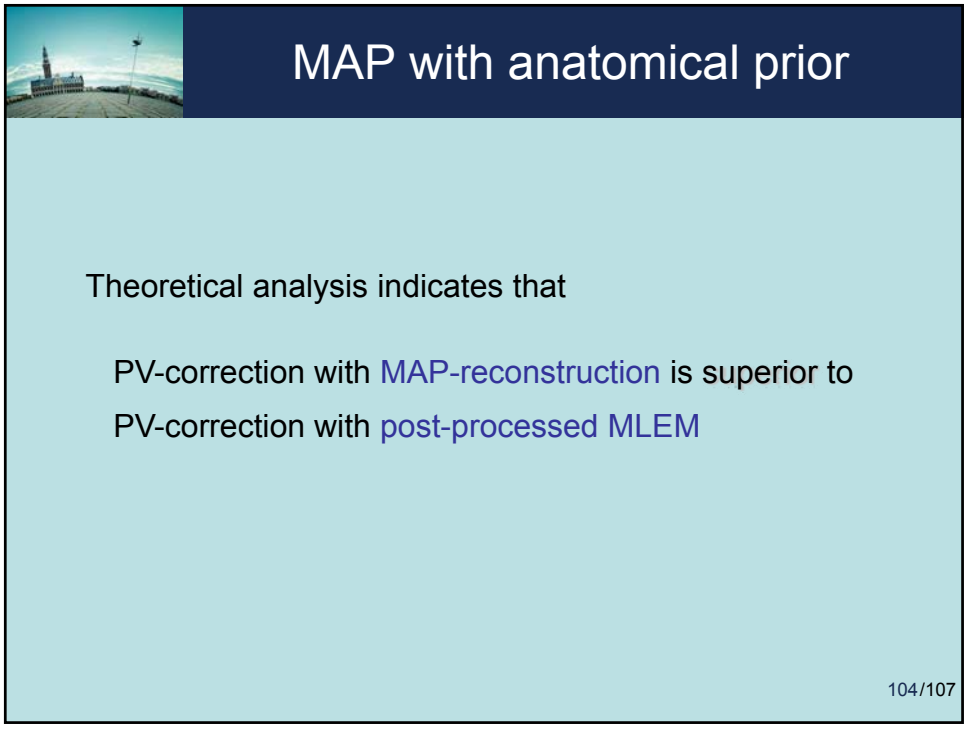
- CSF: no tracer uptake
- white: uniform, low tracer uptake
- grey: higher tracer uptake, possibly lesions

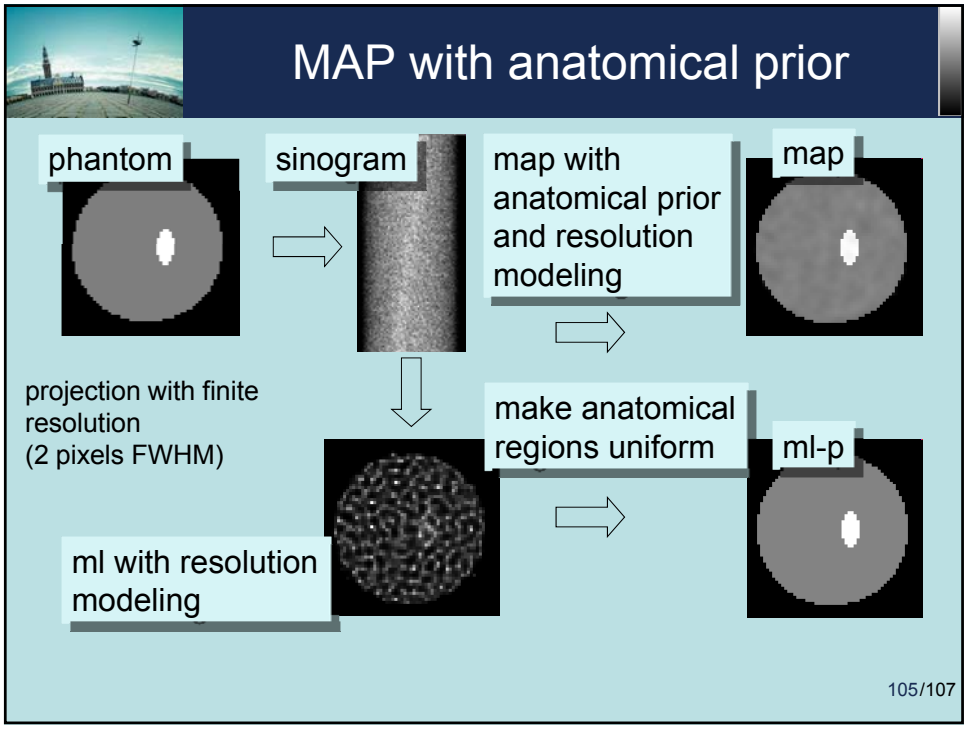
100/107

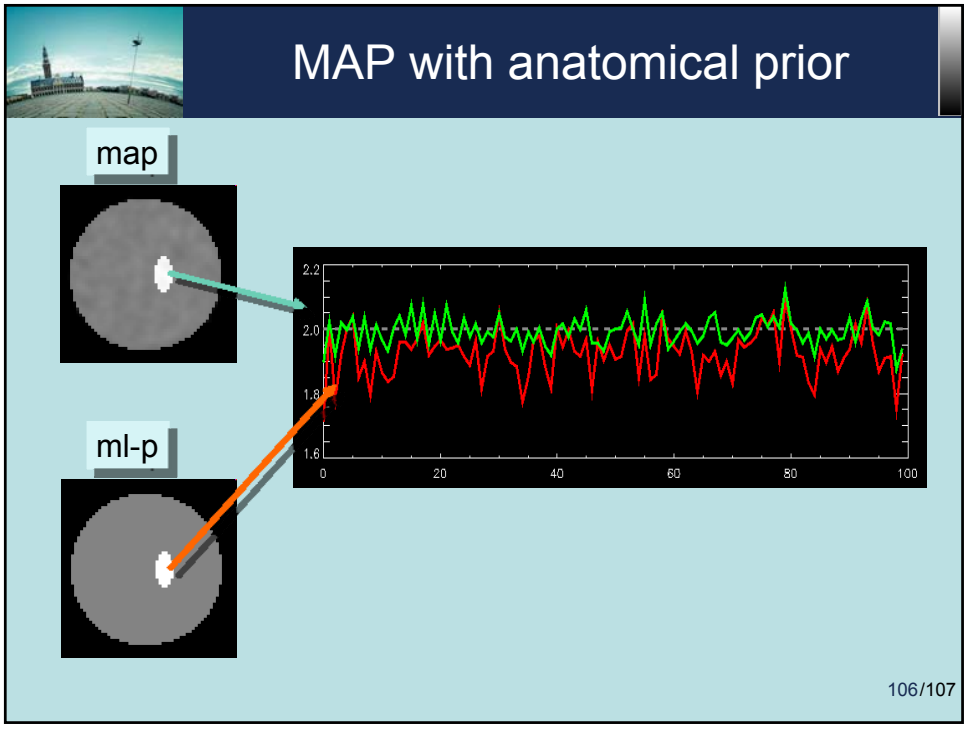

MAP with anatomical prior
MLEM
resolution modelling
+ MAP
• Markov prior in gray matter
• Intensity prior in white (with estimated mean)
• Intensity prior in CSF (mean = 0)
101/107

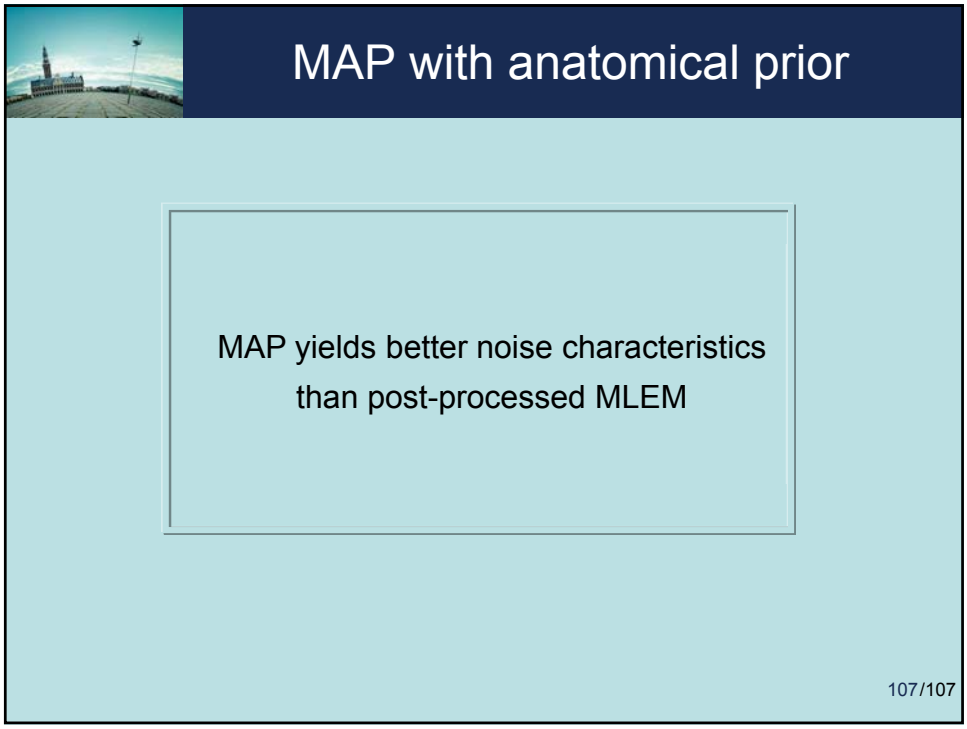

TRUE
MLEM - no RR
MLEM - with RR
MAP - with RR
(asym Bowsher prior)
MRI
102/107


TRUE
MLEM - no RR
MLEM - with RR
MAP - with RR
(asym Bowsher prior)
MRI
103/107


MAP with anatomical prior
Theoretical analysis indicates that
PV-correction with MAP-reconstruction is superior to
PV-correction with post-processed MLEM
104/107


MAP with anatomical prior
phantom
sinogram
map with anatomical prior and resolution modeling
map
projection with finite resolution (2 pixels FWHM)
make anatomical regions uniform
ml-p
ml with resolution modeling
105/107


MAP with anatomical prior
map
ml-p
2.2
2.0
1.8
1.6
0
20
40
60
80
100
106/107


MAP with anatomical prior
MAP yields better noise characteristics
than post-processed MLEM
107/107

IEEE MIC Reconstruction Short Course 2011

Iterative Methods in X-Ray Computed Tomography

Bruno De Man

deman@ge.com

IEEE Nuclear Science Symposium and Medical Imaging Conference, Valencia, 2011

Statistical Methods for Image Reconstruction

Arkadiusz Sitek, Johan Nuyts and Bruno De Man

Part 2 : X-ray Computed Tomography

Bruno De Man

Symbols

$\mu_i^n, \mu(x,y)$	linear attenuation coefficient	$\tilde{p}_i, \tilde{p}_m(s)$	filtered projection data
N, N	iteration number, total number of iterations	$h(s)$	ramp filter kernel
j, J	voxel index, total number of voxels	θ_m	view angle of view number m
I_{ij}	interpolation coefficient or element of the system matrix	$\otimes$	convolution
$p_i, p_m(s)$	attenuation line integral measurement or projection data	y_i	intensity measurement
i, I	line integral index, total number of line integrals	e_i	intensity measurement noise
x, y	voxel coordinates	$\overline{y}_i$	intensity expected value
m, M	view number, total number of views	s, S	sample index, total number of samples

Acronyms

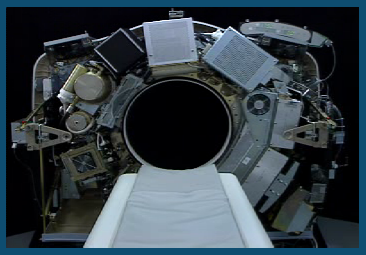
2D	two dimensional
3D	three dimensional
bpm	beats per minute
CG	conjugate gradients
COS	convergent ordered subsets
CT	computed tomography
DD	distance driven
FBP	filtered nackprojection
FDK	Feldkamp Davis Kress
FWHM	full width at half maximum
HU	Hounsfield Units
ICD	iterative coordinate descent
IQ	image quality
IR	iterative reconstruction
LS	least squares
MAP	maximum a posteriori
MAR	metal artifact reduction
ML	maximum likelihood
MLTR	ML for transmission
MRF	Markov random field
MTF	modulation transfer function
OS	ordered subsets
PD	pixel driven
PSF	point spread function
PWLS	penalized WLS
PWMLTR	phased weighted MLTR
RD	ray driven
ROI	region of interest
SPS	separable parabolic surrogate
WLS	weighted least squares

Overview

- CT basics
- Bayesian framework & noise models
- Forward model – projector/backprojector
- Update step
- Image quality
- Making it work
- Advanced forward model - Incorporate physics

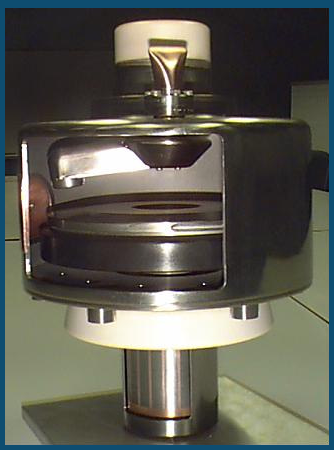
CT scanner

Under the covers



Bruno De Man
GE Global Research
IEEE MIC, Valencia, 2011

X-ray tube



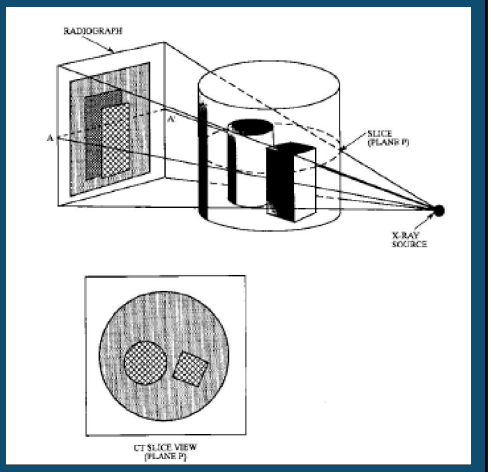
Bruno De Man
GE Global Research
IEEE MIC, Valencia, 2011

CT detector

Bruno De Man
GE Global Research
IEEE MIC, Valencia, 2011

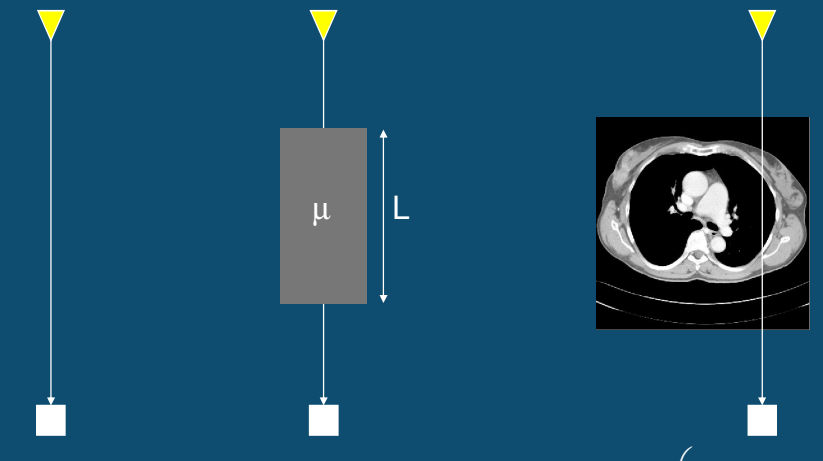
Basic principle

1. Produce X-rays w/ X-ray tube
2. Pass x-rays through patient
3. Detect on the other side
4. Repeat from all angles surrounding patient
5. Reconstruct cross sectional images
6. Pixel values represent attenuation of tissue



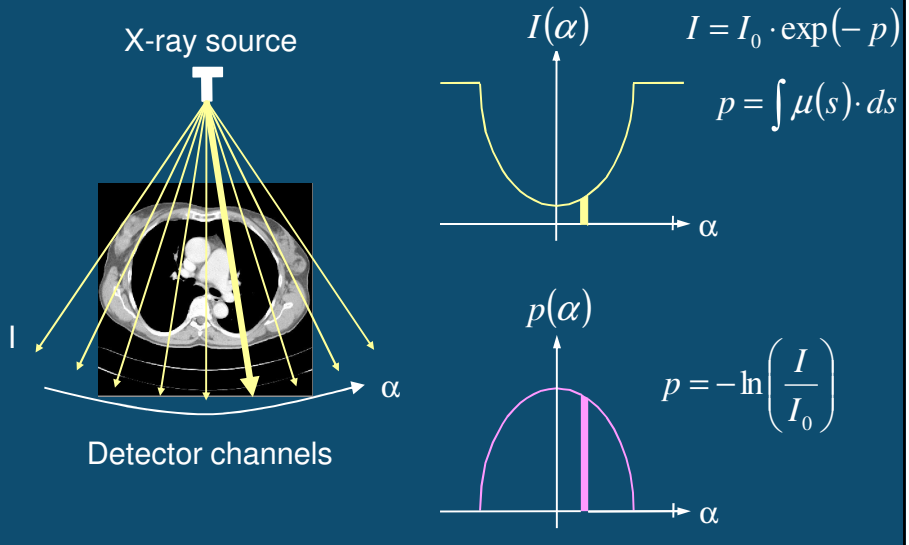
http://www.slaney.org/pct/

Beer's law



$$I = I_0 \qquad I = I_0 \cdot \exp(-\mu \cdot L) \qquad I = I_0 \cdot \exp\left(-\int \mu(s) \cdot ds\right)$$

One "view" or "projection"

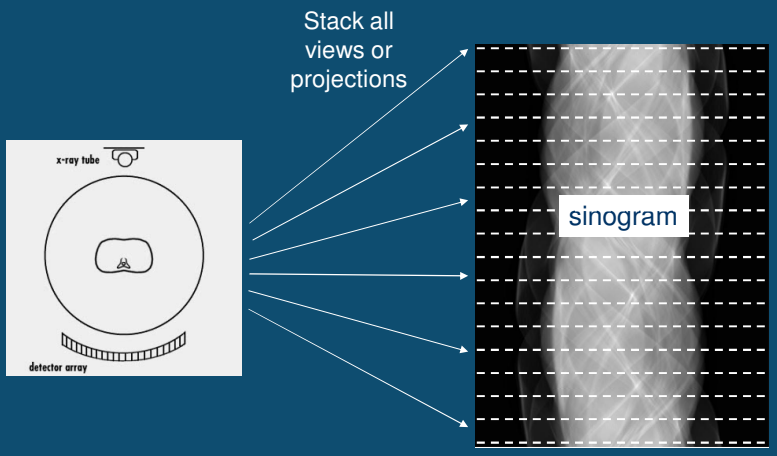


$$I = I_0 \cdot \exp(-p)$$

$$p = \int \mu(s) \cdot ds$$

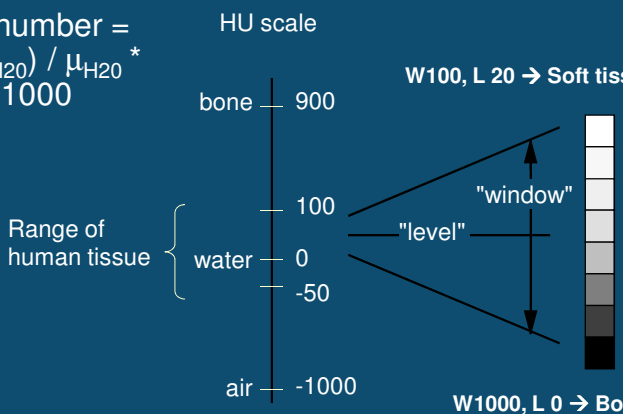
$$p = -\ln\left(\frac{I}{I_0}\right)$$

Sinogram



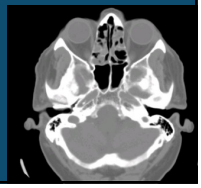
CT number - Hounsfield Units

CT number = $(\mu-\mu_{H20}) / \mu_{H20} * 1000$



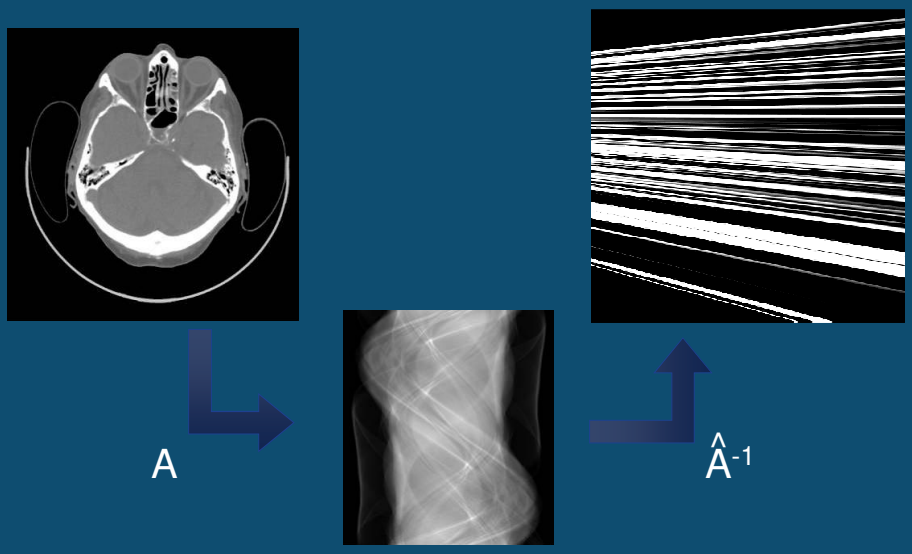
W100, L 20 → Soft tissue contrast visible

W1000, L 0 → Bone structure visible



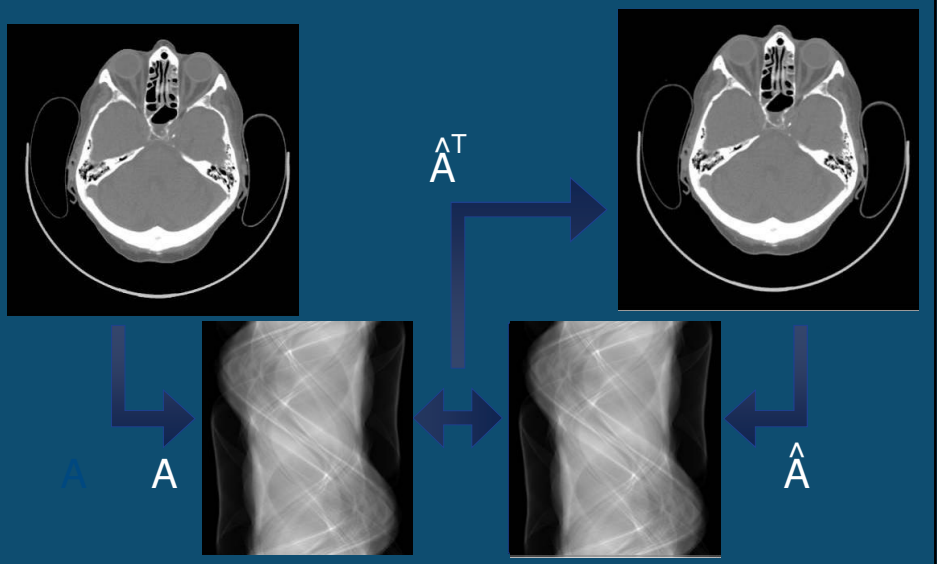
< Courtesy of Tom Toth (GE Healthcare) >

Filtered backprojection



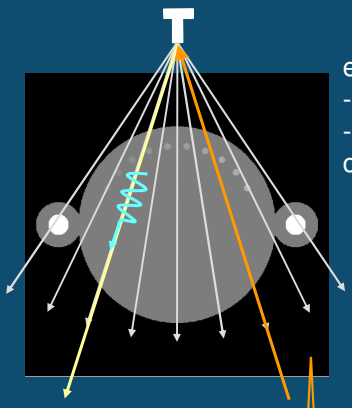
Shepp, IEEE Trans. Nucl. Sci., 1974

Iterative reconstruction



CT measurement noise

quantum noise :
-Poisson distribution for photon-counting
-variance equal to number of photons
-Compound-Poisson for energy-integrating



electronic noise :
-normal distribution
-variance is independent of signal strength

$\sigma_I^2 = \sigma_{quantum}^2 + \sigma_{electronic}^2$

σ_p

Exercise 1

Assume :
- an x-ray flux of 1.e6 photons per channel per view (air scan)
- a 20cm water phantom with $\mu_{water}=0.2cm^{-1}$
- a 1cm central low-contrast object with $\mu_{water}=0.21cm^{-1}$
- a photon-counting detector with 100% detection efficiency

Question :
- what is the contrast-to-noise ratio (CNR) in one single view ?
- in the intensity-domain ?
- and in the attenuation-domain ? (after log-conversion)

Hints :
- CNR=contrast/noise= $\Delta_{signal}/\sigma_{signal}$
- Start with Beer's law to compute detected signal
- Compute contrast as difference in I (or p) when adding low-contrast object
- Compute Poisson noise : σ_{signal} = sqrt(mean)
- $\sigma_p = \sigma_I \cdot |\partial p/\partial I|$

Overview

- CT basics
- Bayesian framework & noise models
- Forward model – projector/backprojector
- Update step
- Image quality
- Making it work
- Advanced forward model - Incorporate physics

Bruno De Man
GE Global Research
IEEE MIC, Valencia, 2011

Bayesian framework

$$\arg\max_{\text{img}} \left[P(\text{img} \mid \text{meas}) \right]$$

$$= \arg\max_{\text{img}} \left[\frac{P(\text{meas} \mid \text{img}) \cdot P(\text{img})}{P(\text{meas})} \right]$$

$$= \arg\max_{\text{img}} \left[\log \frac{P(\text{meas} \mid \text{img}) \cdot P(\text{img})}{P(\text{meas})} \right]$$

$$= \arg\max_{\text{img}} \left[\underbrace{\log P(\text{meas} \mid \text{img})}_{\text{LIKELIHOOD}} + \underbrace{\log P(\text{img})}_{\text{PRIOR}} \right]$$

ML

MAP

Bayesian framework

$$\arg\max_{img} [\overbrace{\log P(meas \mid img)}^{\text{LIKELIHOOD}} + \overbrace{\log P(img)}^{\text{PRIOR}}]$$

Forward model → $calc = f(img)$

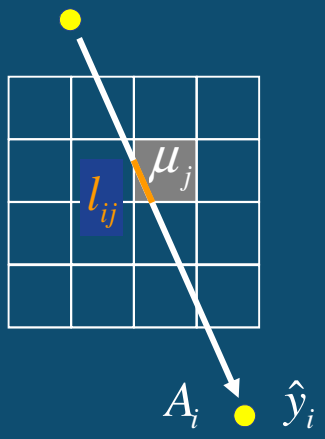
Noise model → $P(meas \mid calc)$

$\Big\} P(meas \mid img)$

Optimization method → Update step

Basic CT forward model

$$\hat{y}_i = A_i \exp\left(-\sum_{j=1}^{J} l_{ij}\mu_j\right)$$



A_i: intensity sinogram in air

$\hat{y}_i$: calculated intensity sino

i : projection line (or sino) index

μ_j : linear attenuation coeff (1/cm)

j : image index

l_{ij} : intersection length (cm)

Log-likelihood

$$P(\text{meas} \mid \text{calc}) = \prod P(y_i \mid \hat{y}_i)$$

$$\log P(\text{meas} \mid \text{calc}) = \log \prod P(y_i \mid \hat{y}_i) = \sum \log P(y_i \mid \hat{y}_i)$$

with $\hat{y}_i = A_i \exp\left(-\sum_{j=1}^{J} l_{ij}\mu_j\right)$

Bruno De Man
GE Global Research
IEEE MIC, Valencia, 2011

Noise model

POISSON :

$$P(y_i \mid \hat{y}_i) = \frac{\hat{y}_i^{\,y_i} e^{-\hat{y}_i}}{y_i!}$$

$$\log P(y_i \mid \hat{y}_i) = y_i \log \hat{y}_i - \hat{y}_i - \log(y_i!)$$

GAUSSIAN :

$$P(y_i \mid \hat{y}_i) = \frac{1}{2\pi\sigma_i} e^{-\frac{(y_i-\hat{y}_i)^2}{2\sigma_i^2}}$$

$$\log P(y_i \mid \hat{y}_i) = -\frac{(y_i-\hat{y}_i)^2}{2\sigma_i^2} - \log(2\pi\sigma_i)$$

Poisson log-likelihood

Poisson Noise model

$$\log P(y_i \mid \hat{y}_i) = y_i \log \hat{y}_i - \hat{y}_i - \log(y_i!)$$

Log-Likelihood

$$\log P(meas \mid calc) = \sum_i (y_i \log \hat{y}_i - \hat{y}_i)$$

with $$\hat{y}_i = A_i \exp\left(-\sum_{j=1}^{J} l_{ij}\mu_j\right)$$

Cost functions

ML/MAP : (LIKELIHOOD, PRIOR)

$$\arg\max_{\mu_j}\left[\sum_{i\in I}(y_i \ln \hat{y}_i - \hat{y}_i) - \beta\sum_{j\in J}\sum_{k\in J} N_{jk}\phi(\mu_j - \mu_k)\right]$$

WLS/PWLS : (DATAFIT, REGULARIZER)

$$\arg\min_{\mu_j}\left[\sum_{i\in I}\frac{1}{\sigma_i^2}(p_i - \hat{p}_i)^2 + \beta\sum_{j\in J}\sum_{k\in J} N_{jk}\phi(\mu_j - \mu_k)\right]$$

y_i : measured intensity sino

$\hat{y}_i$: calculated intensity sino

p_i : measured int./atten. sino

$\hat{p}_i$: calculated int./atten. sino

β : prior weight

N_{jk} : neighbourhood mask

ϕ : potential function

σ_i : standard deviation

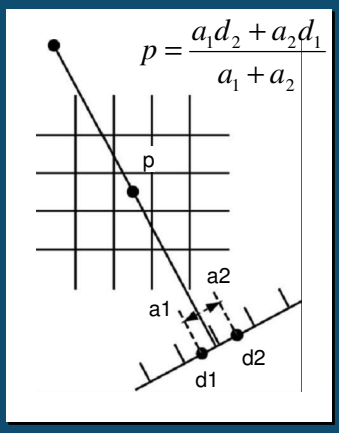
Overview

- CT basics
- Bayesian framework & noise models
- Forward model – projector/back-projector
- Update step
- Image quality
- Making it work
- Advanced forward model - Incorporate physics

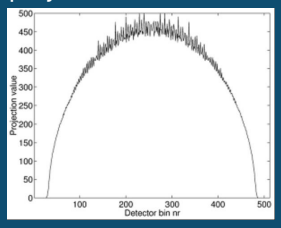
Bruno De Man
GE Global Research
IEEE MIC, Valencia, 2011

Pixel-driven with linear interpolation

Backprojection



Reprojection of uniform disk

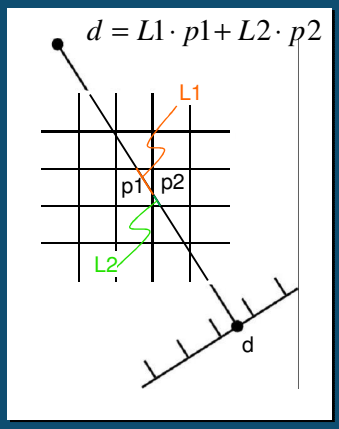


Backprojection of uniform view

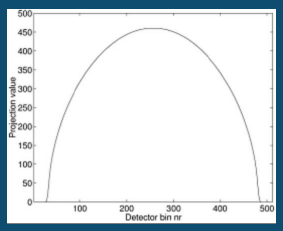
T. Peters, IEEE Trans. Nucl. Sci., 1981

Ray-driven with intersection length

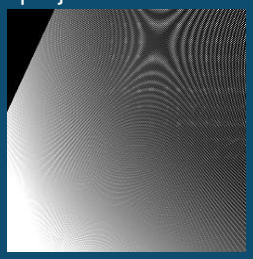
- Re-projection



- Reprojection of uniform disk



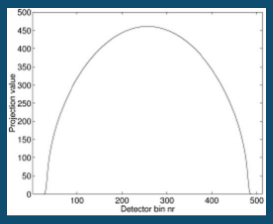
- Backprojection of uniform view



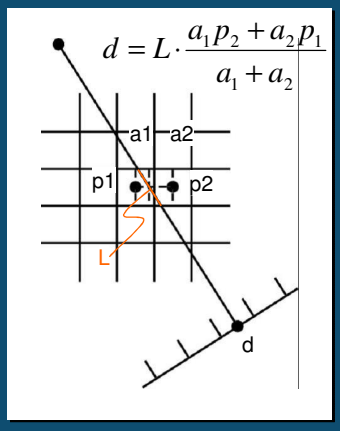
R.Siddon, Med. Phys., 1985

Ray-driven with linear interpolation

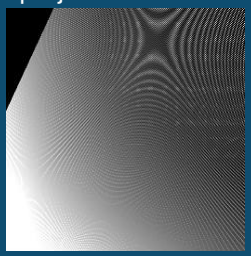
- Reprojection of uniform disk



- Re-projection



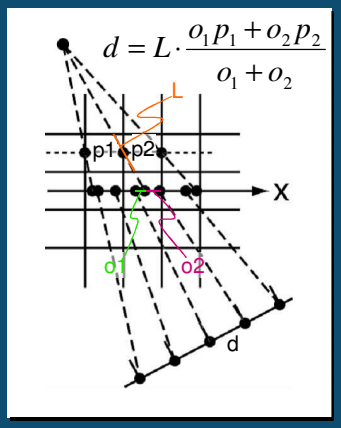
- Backprojection of uniform view



P.M.Joseph, IEEE Trans. Med. Im., 1983

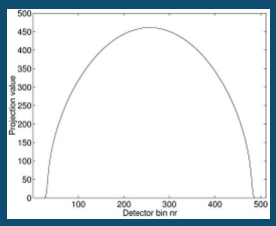
Distance-driven

- Re-projection



Based on pixel and detector *boundaries* !

- Reprojection of uniform disk

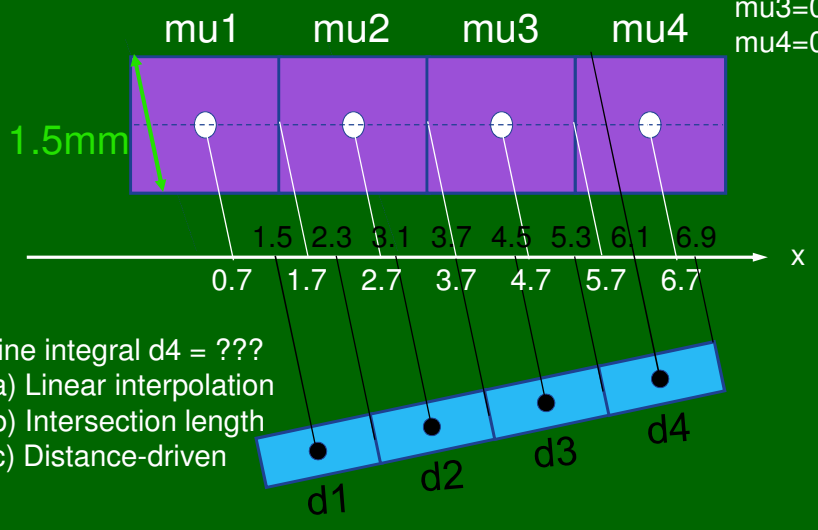


- Backprojection of uniform view

B. De Man, IEEE MIC, 2002

Exercise 2

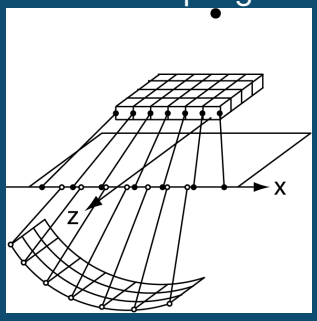
mu1=0.2cm^{-1}
mu2=0.4cm^{-1}
mu3=0.6cm^{-1}
mu4=0.4cm^{-1}



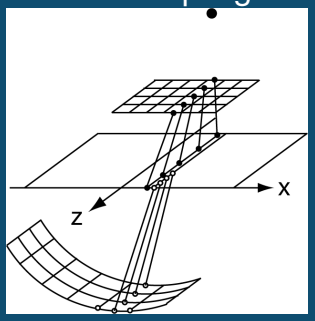
Line integral d4 = ???
(a) Linear interpolation
(b) Intersection length
(c) Distance-driven

Distance-driven in 3D

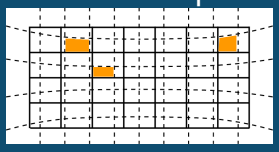
x-resampling



z-resampling

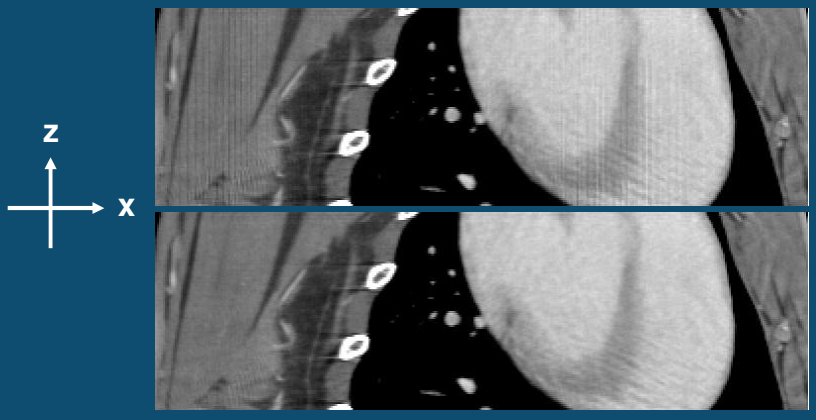


xz-overlap



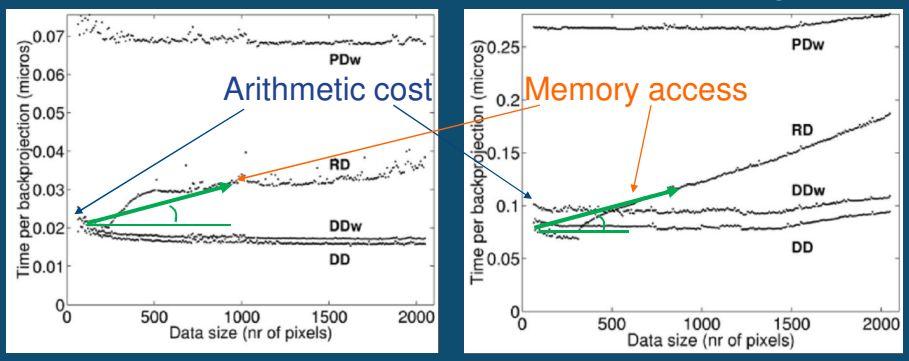
B. De Man, Phys. Med. Biol., 2004

3D iterative recon example
(longitudinal reformat)



Computational performance

Pentium4 – 1.8Ghz | **SUN E4500 UltraSparc-II**



Overview

- CT basics
- Bayesian framework & noise models
- Forward model – projector/backprojector
- → Update step
- Image quality
- Making it work
- Advanced forward model - Incorporate physics

Iterative Coordinate Descent

$$L=\frac{1}{2}\sum_i w_i(p_i-\hat{p}_i)^2+\frac{1}{2}\sum_j\frac{1}{q\sigma^q}\sum_{k\in N_j}n(k-j)\frac{|\mu_j-\mu_k|^p}{1+\left|\frac{(\mu_j-\mu_k)}{\delta}\right|^{p-q}}$$

For n = 1 : N (iteration number)

For j = 1 : J (random scan pattern)

$$\mu_j^n=\arg\min_{\mu_j}\left(L(\mu_1^n,...,\mu_{j-1}^n,\mu_{j+1}^{n-1},...,\mu_J^{n-1})\right)$$

half-interval line search applied to find root of $\frac{\partial L}{\partial \mu}$

K Sauer, IEEE Trans. Signal Proc., 1993

Exercise 4

ICD

$$L=\frac{1}{2}\sum_i w_i\left(p_i-\hat{p}_i^n\right)^2$$

$$\hat{p}_i^n=\sum_j l_{ij}\mu_j^n$$

$$j\neq k\rightarrow\mu_j^{n+1}=\mu_j^n$$

$$j=k\rightarrow\mu_j^{n+1}=\mu_j^n+\delta^{n+1}$$

Update step → $\left.\frac{dL}{d\mu}\right|_{\mu_j^{n+1}}=0\Leftrightarrow\delta^{n+1}=???$

MLTR / SPS derivation

$$\log P(meas \mid calc) = \sum_i \left(y_i \log \hat{y}_i - \hat{y}_i \right)$$

$$\hat{y}_i = A_i \exp\left(-\sum_{j=1}^{J} l_{ij}\mu_j \right)$$

1. The likelihood for transmission tomography

$$L(\mu) = \sum_i \ln p\left(y_i \mid b_i e^{-\sum_j l_{ij}\mu_j} + r_i \right)$$

$$= \sum_i \left(y_i \ln(b_i e^{-\sum_j l_{ij}\mu_j} + r_i) - (b_i e^{-\sum_j l_{ij}\mu_j} + r_i) \right)$$

$$= \sum_i h_i(\sum_j l_{ij}\mu_j)$$

with $h_i(x) = y_i \ln t_i(x) - t_i(x)$

$$t_i(x) = b_i e^{-x} + r_i$$

J A Fessler, IEEE Trans. Med. Imag., 1997

< Courtesy of J. Nuyts (U. Leuven) >

MLTR / SPS derivation

2. Rewrite as a function of difference between new and old

$$\sum_j l_{ij}\mu_j = \sum_j \alpha_{ij} \left(\frac{l_{ij}}{\alpha_{ij}} \left(\mu_j - \mu_j^{\text{old}} \right) + \sum_k l_{ik}\mu_k^{\text{old}} \right)$$

with

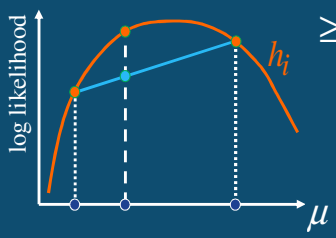
$$\sum_j \alpha_{ij} = 1$$

< Courtesy of J. Nuyts (U. Leuven) >

MLTR / SPS derivation

3. Use concavity

$$L(\mu) = \sum_i h_i\left(\sum_j l_{ij}\mu_j\right) = \sum_i h_i\left(\sum_j \alpha_{ij}\left(\frac{l_{ij}}{\alpha_{ij}}\left(\mu_j - \mu_j^{\text{old}}\right) + \sum_k l_{ik}\mu_k^{\text{old}}\right)\right)$$



$$\geq \sum_i \sum_j \alpha_{ij} h_i\left(\frac{l_{ij}}{\alpha_{ij}}\left(\mu_j - \mu_j^{\text{old}}\right) + \sum_k l_{ik}\mu_k^{\text{old}}\right)$$

$$= \sum_j Q_j\left(\mu_j, \mu^{\text{old}}\right)$$

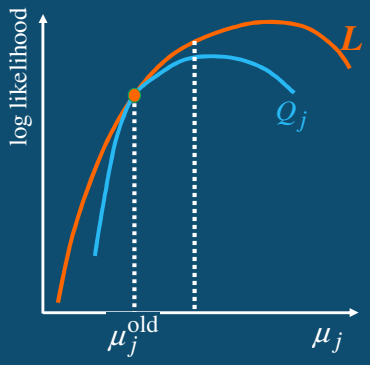
with $Q_j\left(\mu_j, \mu^{\text{old}}\right) = \sum_i \alpha_{ij} h_i\left(\frac{l_{ij}}{\alpha_{ij}}\left(\mu_j - \mu_x^{\text{old}}\right) + \sum_k l_{ik}\mu_k^{\text{old}}\right)$

$$h_i(x) = y_i \ln t_i(x) - t_i(x)$$

< Courtesy of J. Nuyts (U. Leuven) >

MLTR / SPS derivation

4. Maximizing (or increasing) *Q* increases *L*



$$L\left(\mu^{\text{old}}\right) = \sum_j Q_j\left(\mu_j^{\text{old}}, \mu^{\text{old}}\right)$$

$$L(\mu) \geq \sum_j Q_j\left(\mu_j, \mu^{\text{old}}\right)$$

$$\left.\frac{\partial L}{\partial \mu_j}\right|_{\mu_j^{\text{old}}} = \left.\frac{\partial Q_j}{\partial \mu_j}\right|_{\mu_j^{\text{old}}}$$

< Courtesy of J. Nuyts (U. Leuven) >

MLTR / SPS derivation

5. Newton method

$$\mu_j = \mu_j^{\text{old}} + \Delta\mu_j \qquad \Delta\mu_j = \frac{\left.\frac{\partial Q_j}{\partial \mu_j}\right|_{\mu_j^{\text{old}}}}{-\left.\frac{\partial^2 Q_j}{\partial \mu_j^{\,2}}\right|_{\mu_j^{\text{old}}}}$$

$$\Delta\mu_j = \frac{\sum_i l_{ij}(t_i - y_i)}{\sum_i \frac{l_{ij}^{\,2}}{\alpha_{ij}}(t_i)} \quad \textbf{with} \quad t_i = b_i e^{-\sum_k l_{ik}\mu_k^{\text{old}}} + r_i$$

< Courtesy of J. Nuyts (U. Leuven) >

MLTR / SPS derivation

6. Choosing α_{ij} Recall that $\sum_j \alpha_{ij} = 1$

Assume for simplicity that $r_i = 0$

$$\alpha_{ij} = \frac{l_{ij}\mu_j}{\sum_k l_{ik}\mu_k} \Rightarrow \Delta\mu_j = \frac{\mu_j \sum_i l_{ij}(t_i - y_i)}{\sum_i l_{ij}\left(\sum_k l_{ik}\mu_k\right)t_i}$$

$$\alpha_{ij} = \frac{l_{ij}}{\sum_k l_{ik}} \Rightarrow \Delta\mu_j = \frac{\sum_i l_{ij}t_i - \sum_i l_{ij}y_i}{\sum_i l_{ij}\left(\sum_k l_{ik}\right)t_i}$$

$$t_i = b_i e^{-\sum_k l_{ik}\mu_k} + r_i$$

< Courtesy of J. Nuyts (U. Leuven) >

Exercise 5

$$L = \sum_i y_i \ln \hat{y}_i - \hat{y}_i$$

$$\hat{y}_i = A_i \exp(-\sum_j l_{ij}\mu_j)$$

$$\Delta\mu_j^{n+1} = -\frac{\left.\frac{\partial L}{\partial \mu_j}\right|_{\{\mu_j^n\}}}{\sum_{\xi\in J}\left.\frac{\partial^2 L}{\partial \mu_j \partial \mu_\xi}\right|_{\{\mu_j^n\}}} \qquad \Delta\mu_j^{n+1} = ?$$

J Nuyts, 2nd IEEE Workshop CMP, 1996
J Nuyts, Phys. Med. Biol., 1998

Ordered-subsets algorithms

C=Calculated
M=Measured

OS-MLTR (Nuyts '96)

$$\mu_j^{n+1} = \mu_j^n + \frac{\sum_{i\in S} l_{ij}(C_i - M_i)}{\sum_{i\in S} l_{ij} C_i \sum_{\xi\in J} l_{i\xi}}$$

OS-MLMOD

$$\mu_j^{n+1} = \mu_j^n + \frac{\sum_{i\in S} l_{ij}(1 - M_i/C_i)M_i}{\frac{|S|}{|I|}\sum_{i\in I} l_{ij} M_i \sum_{\xi\in J} l_{i\xi}}$$

OS-SPS (Fessler '97)

$$\mu_j^{n+1} = \mu_j^n + \frac{\sum_{i\in S} l_{ij}(C_i - M_i)}{\sum_{i\in S} l_{ij} M_i \sum_{\xi\in J} l_{i\xi}}$$

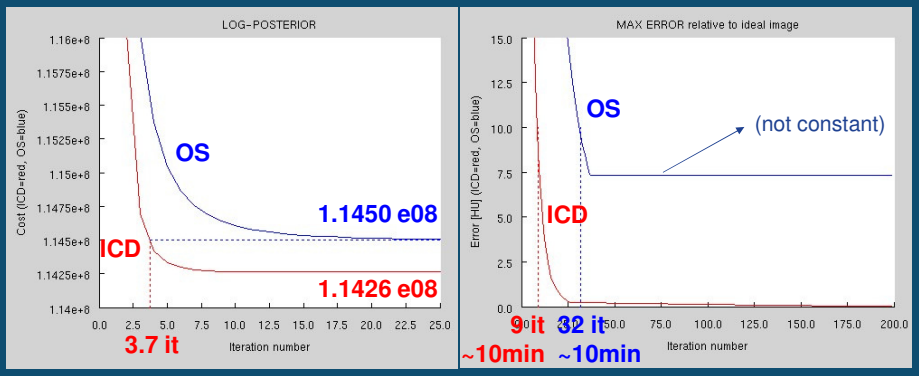
OS-WLS

$$\mu_j^{n+1} = \mu_j^n + \frac{\sum_{i\in S} l_{ij}\left(P_i - \sum_{k\in S} l_{ik}\mu_k^n\right)M_i}{\frac{|S|}{|I|}\sum_{i\in I} l_{ij} M_i \sum_{\xi\in J} l_{i\xi}}$$

J Nuyts, 2nd IEEE Workshop CMP, 1996
J A Fessler, IEEE Trans. Med. Imag., 1997
B. De Man, ICMP, 2005

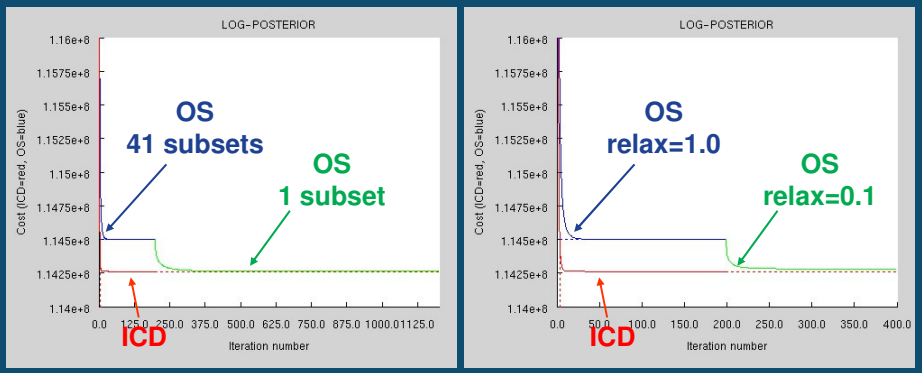
Results : ICD versus OS

With positivity-constraint Initialize with FBP 200 iter
41 subsets (OS)



Results : ICD versus OS

With positivity-constraint Initialize with FBP

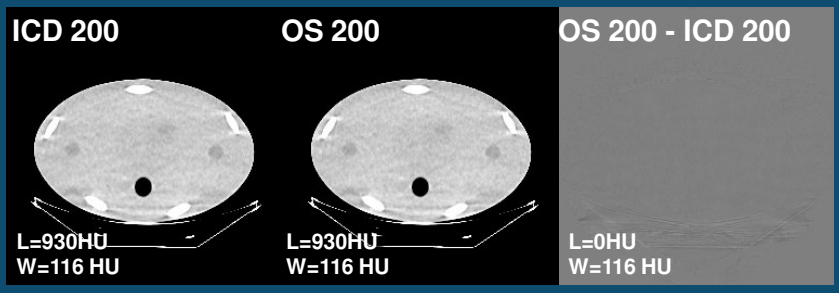


Results : ICD versus OS

With positivity-constraint

Initialize with FBP

200 iter
41 subsets (OS)



Conjugate Gradients

$$L = \frac{1}{2}\sum_i w_i (p_i - \hat{p}_i)^2 + \sum_j \alpha \sum_{\substack{k \in N_j \\ k>j}} n(k-j)\frac{(\mu_j - \mu_k)^2}{2}$$

$$d^{n+1} = \nabla L + \gamma^n d^n$$

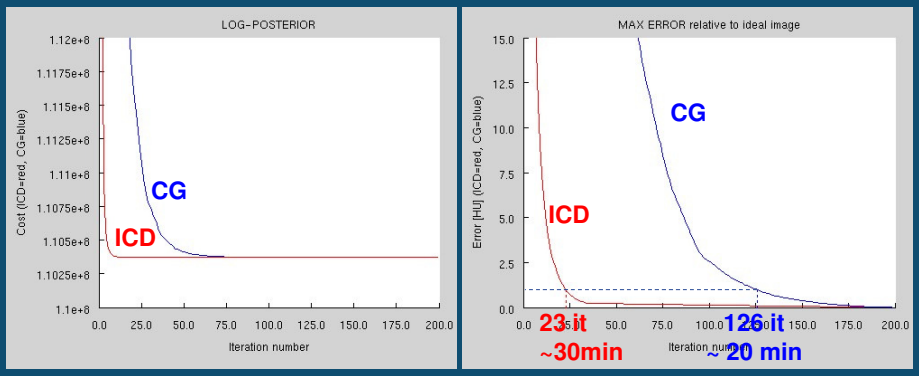
$$\hat{\alpha} = \arg\min_{\alpha} L\left(x^n + \alpha d^{n+1}\right)$$

$$x^{n+1} = x^n + \hat{\alpha} d^{n+1}$$

Results : ICD versus CG

No positivity-constraint
Initialize with FBP
200 iterations

	ICD (200 iter)	CG (200 iter)
Lik	4.2992e+07	4.2993e+07
Prior	6.7379e+07	6.7376e+07
Post	1.1037e+08	1.1037e+08



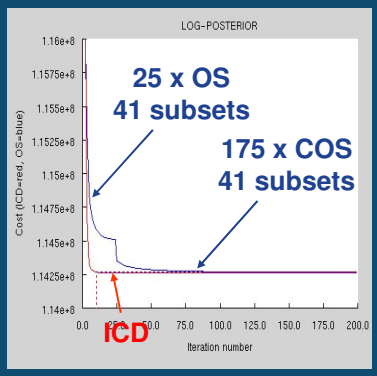
Convergent Ordered Subsets

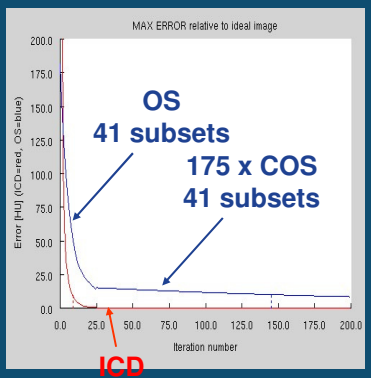
1. Express log-likelihood as sum of N subset terms
2. For each subset, update gradient for one term
3. Perform image update using most recent gradients

Results : ICD versus COS

With positivity-constraint

Initialize with FBP



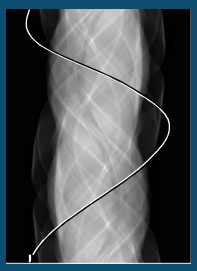


Performance tradeoff

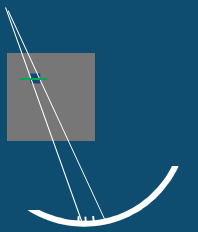
Simultaneous update :
- many iterations
- low arithmetic cost
- sequential memory access

Iterative coordinate descent :
- few iterations
- high arithmetic cost
- “random” memory access

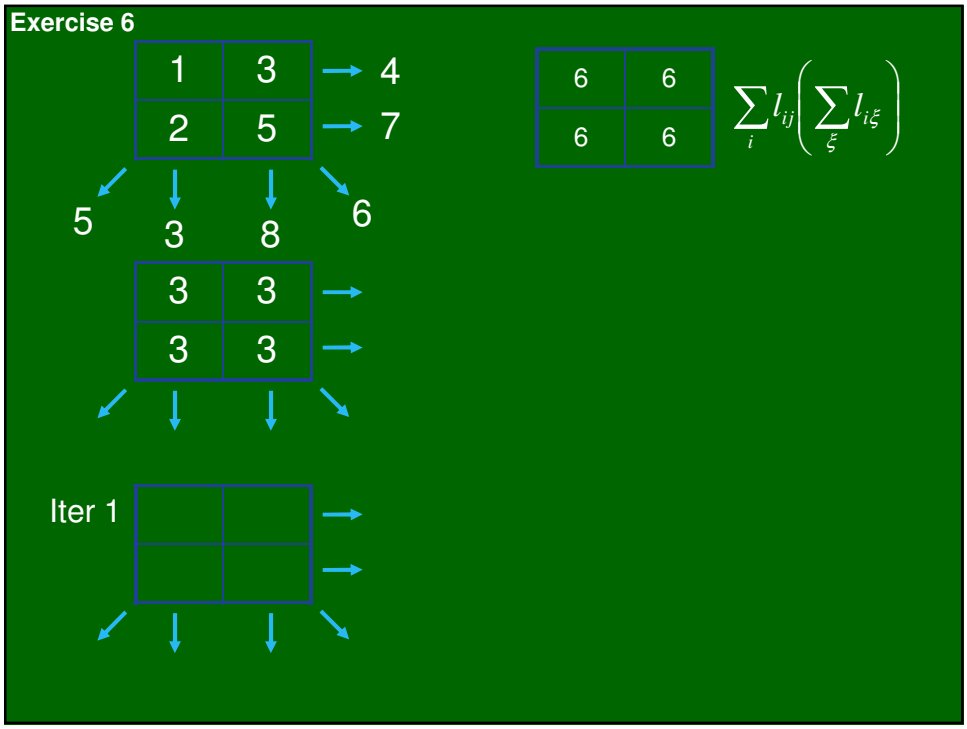
Grouped-coordinate methods :
- combine the best of both worlds ?

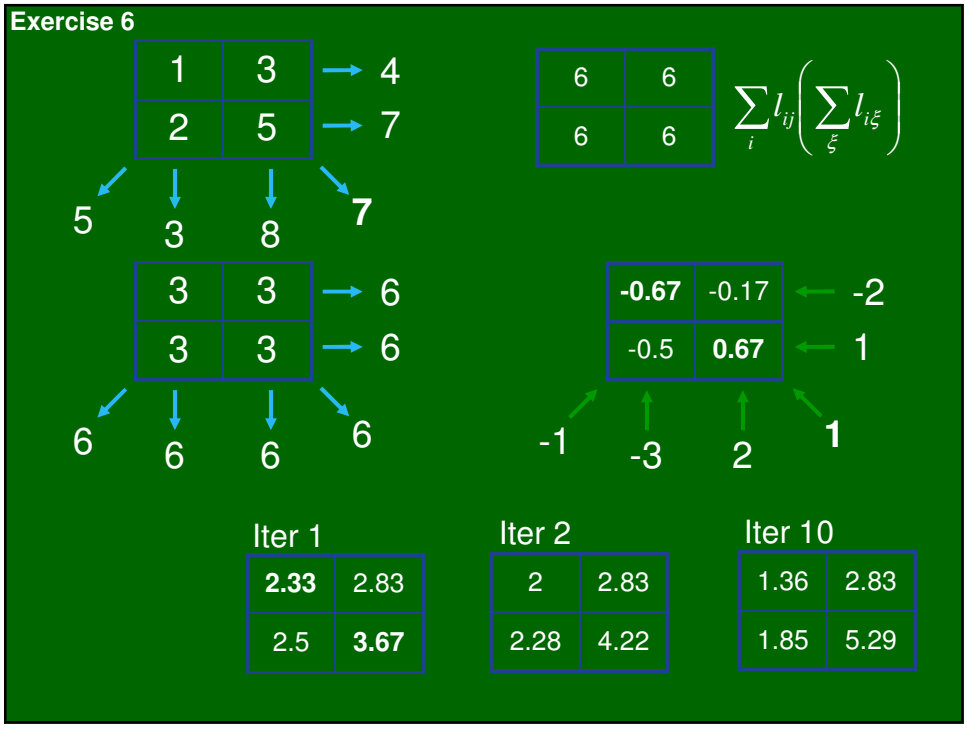


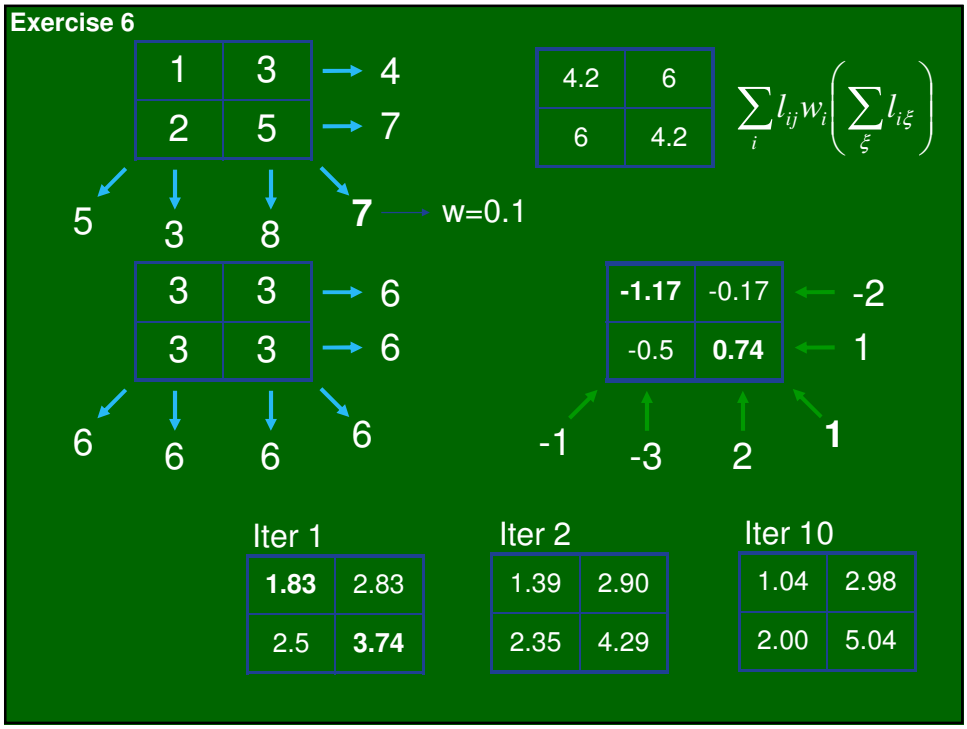
ICD memory access pattern



DD projector for ICD


Exercise 6
1 3
2 5
4
7
5
3
8
6
3 3
3 3
Iter 1
6 6
6 6
$\sum_i l_{ij}\left(\sum_\xi l_{i\xi}\right)$


Exercise 6
1 3
2 5
4
7
5
3
8
7
3 3
3 3
6
6
6
6
6
6
6 6
6 6
$\sum_i l_{ij}\left(\sum_\xi l_{i\xi}\right)$
-0.67 -0.17
-0.5 0.67
-2
1
-1
-3
2
1
Iter 1
2.33 2.83
2.5 3.67
Iter 2
2 2.83
2.28 4.22
Iter 10
1.36 2.83
1.85 5.29



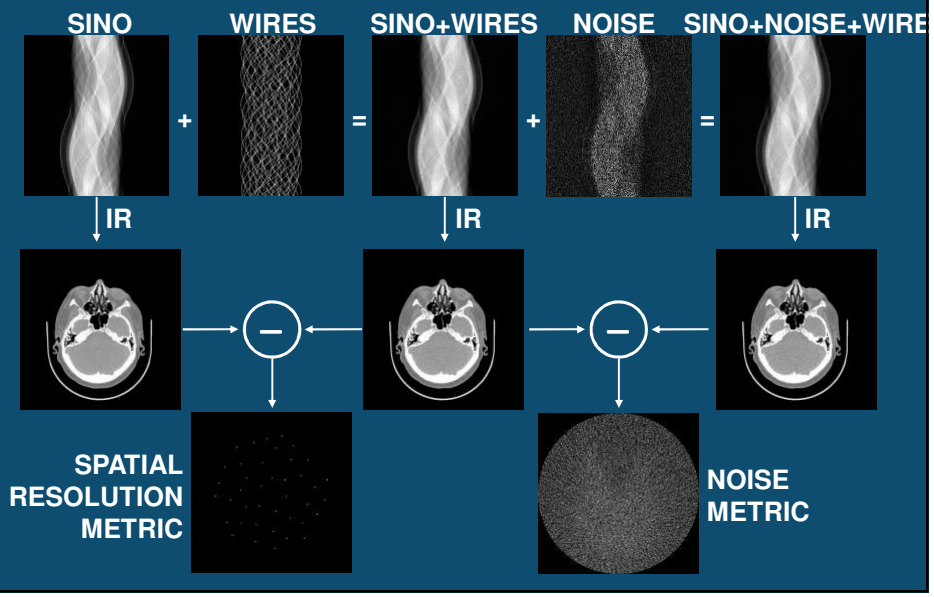
Overview

- CT basics
- Bayesian framework & noise models
- Forward model – projector/backprojector
- Update step
- → Image quality
- Making it work
- Advanced forward model - Incorporate physics

Iterative recon IQ properties

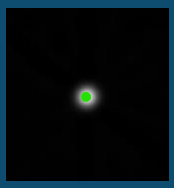
- Dependent on voxel size :
  - voxel size determines number of unknowns
  - effect of prior depends on voxel size
- Non-linearity :
  - effect of prior depends on object contrast
  - strongly attenuating objects impact statistics, and therefore noise and resolution
- Spatial dependence :
  - system geometry results in non-uniform/non-isotropic IQ
  - local statistics determines noise, but also impacts resolution

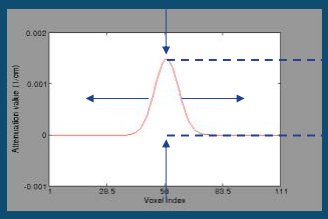
Noise-resolution metric



Use wire amplitude as a relative resolution metric

wire image → wire profile → wire amplitude



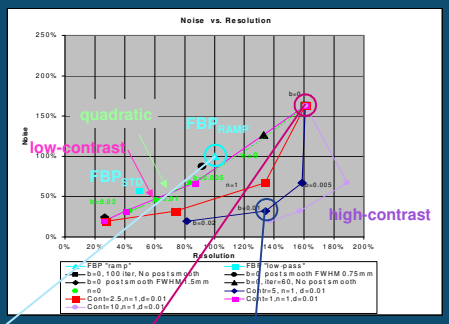


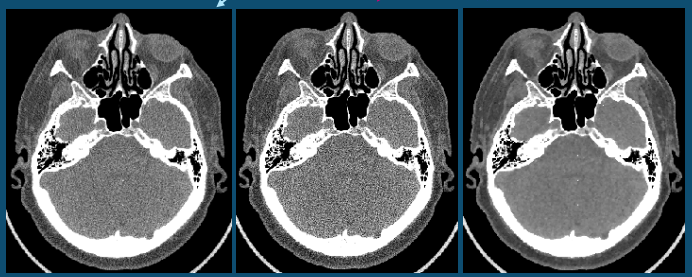
$$\sigma \sim \sqrt{\frac{\iint f}{\max(f)}}$$

PSF_Integral/PSF_Max = $2\pi(\sigma/\text{pixelsize})^2$

B. De Man, Imaging, 2006

IQ analysis (2D example)





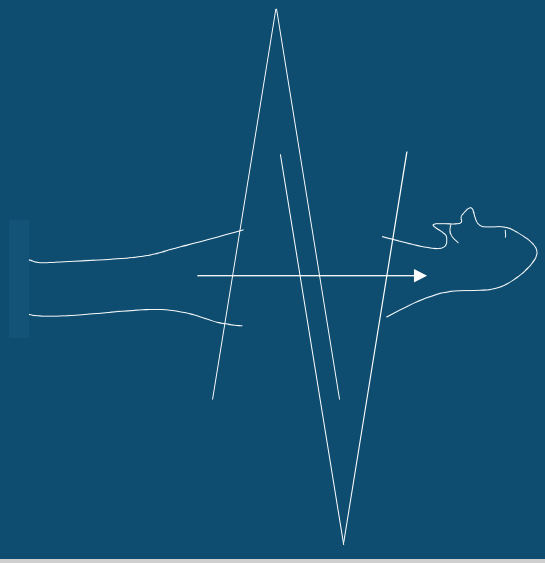
FBP | IR without PRIOR | IR with PRIOR

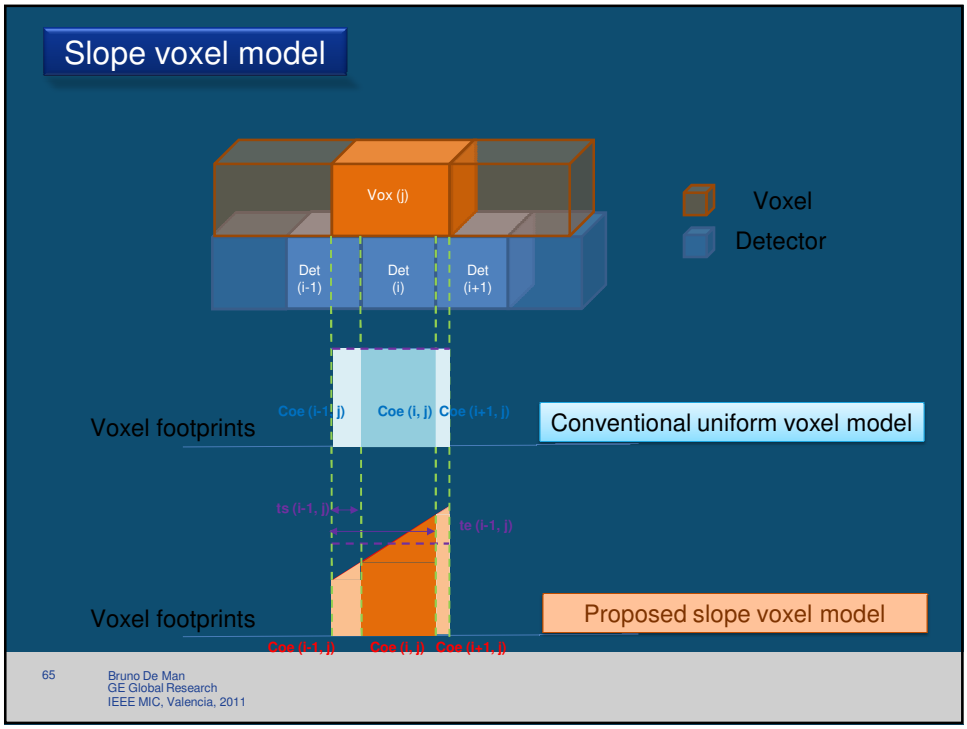
M. Iatrou, IEEE Med. Im. Conf., 2006 & 2007

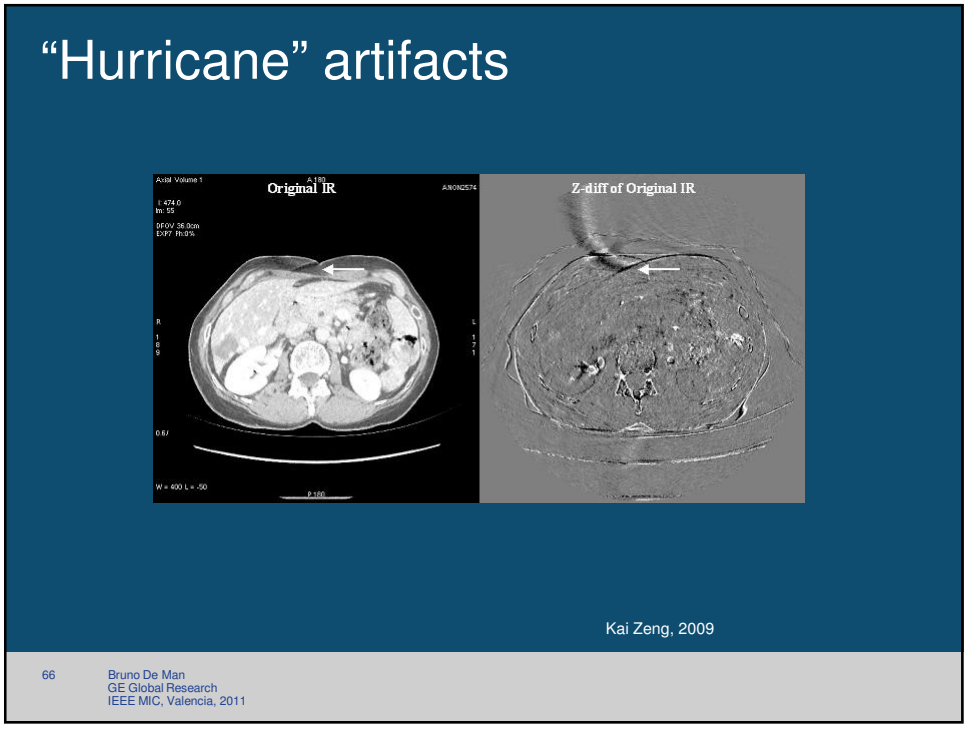
Overview

- CT basics
- Bayesian framework & noise models
- Forward model – projector/backprojector
- Update step
- Image quality
- Making it work
- Advanced forward model - Incorporate physics

Long-object problem




Slope voxel model
Vox (j)
Det (i-1)
Det (i)
Det (i+1)
Voxel
Detector
Voxel footprints
Coe (i-1, j)
Coe (i, j)
Coe (i+1, j)
Conventional uniform voxel model
ts (i-1, j)
te (i-1, j)
Voxel footprints
Coe (i-1, j)
Coe (i, j)
Coe (i+1, j)
Proposed slope voxel model
65
Bruno De Man
GE Global Research
IEEE MIC, Valencia, 2011


"Hurricane" artifacts
Original IR
Z-diff of Original IR
Kai Zeng, 2009
66
Bruno De Man
GE Global Research
IEEE MIC, Valencia, 2011

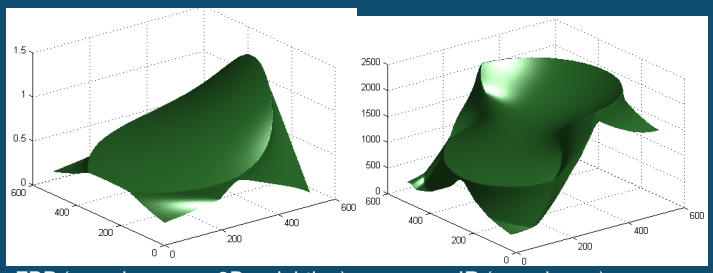
Analysis

- Observations:
  - FBP is free of artifacts
  - IR requires a more advanced system model for forward / back projection

$$\hat{x} = \arg\min_{x \in \Omega} \left\{ \frac{1}{2} (y - Ax)^T W (y - Ax) + \beta U(x) \right\}$$

- Forward model characteristics:
  - Backprojection of unity sinogram

$$x_1 = A^T y_1$$



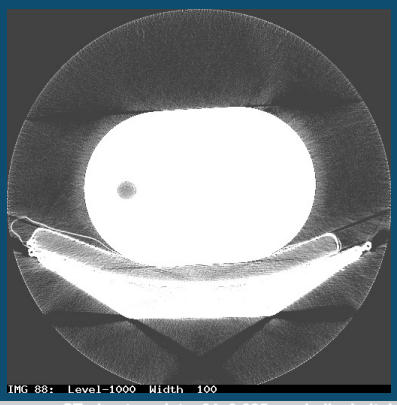
FBP (cone-beam, no 3D weighting) IR (cone-beam)

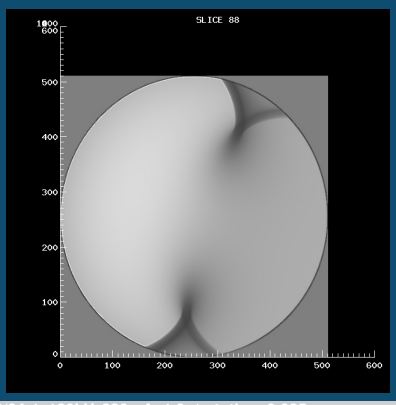
  - FBP: each voxel receives contributions from fixed angular range → smooth
  - IR: voxel updates use all available angular ranges → includes redundant data
  - IR benefits: better statistics but non-uniform sampling

Bruno De Man
GE Global Research
IEEE MIC, Valencia, 2011

Analysis

- Theory indicates that IR may use any and all available data for best results
- But redundant data is correlated with helical artifacts:
  - Originating from reconstruction in native 3D cone-beam geometry
  - Enhancing any inconsistencies between model and acquired data



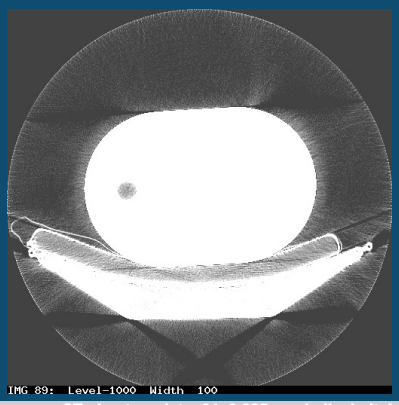


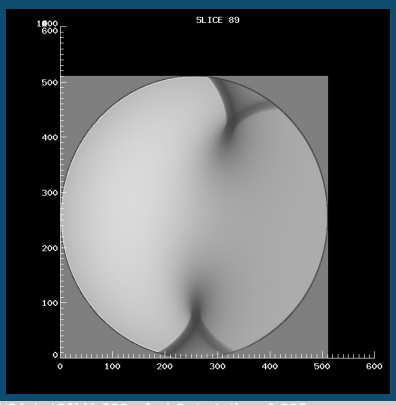
CT phantom data, 64x0.625mm, helical pitch 63/64:1, 120kV, 320mA, 1.0s/rotation, 0.625mm slices

Bruno De Man
GE Global Research
IEEE MIC, Valencia, 2011

Analysis

- Theory indicates that IR may use any and all available data for best results
- But redundant data is correlated with helical artifacts:
  - Originating from reconstruction in native 3D cone-beam geometry
  - Enhancing any inconsistencies between model and acquired data



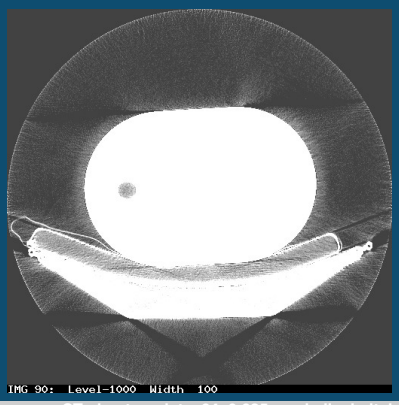


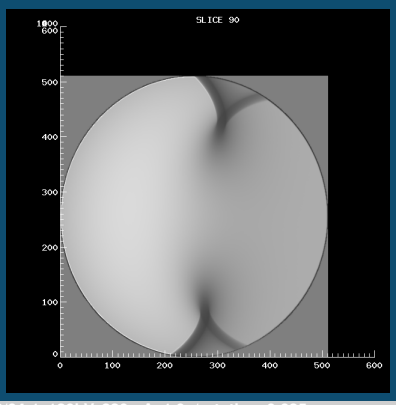
CT phantom data, 64x0.625mm, helical pitch 63/64:1, 120kV, 320mA, 1.0s/rotation, 0.625mm slices

69 Bruno De Man
GE Global Research
IEEE MIC, Valencia, 2011

Analysis

- Theory indicates that IR may use any and all available data for best results
- But redundant data is correlated with helical artifacts:
  - Originating from reconstruction in native 3D cone-beam geometry
  - Enhancing any inconsistencies between model and acquired data





CT phantom data, 64x0.625mm, helical pitch 63/64:1, 120kV, 320mA, 1.0s/rotation, 0.625mm slices

70 Bruno De Man
GE Global Research
IEEE MIC, Valencia, 2011

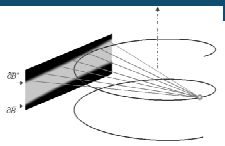
Projection Weighting Methods

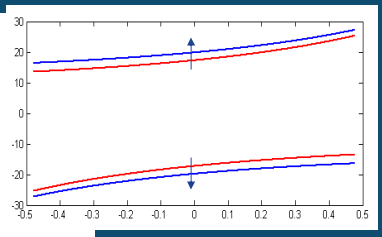
- **Tam-window weighted IR:**
  - Successfully eliminated helical artifact
  - Results in poor data utilization → noise
- **Expand concept to enlarged projection weighting:**
  - Loose constraints on data utilization vs. analytical techniques
  - Example: expanded Tam-based weighting with shift & feathering

$$w_i = win(col(i), row(i))$$

$$\partial BS(c)^+ = \partial B(c)^+ + st$$

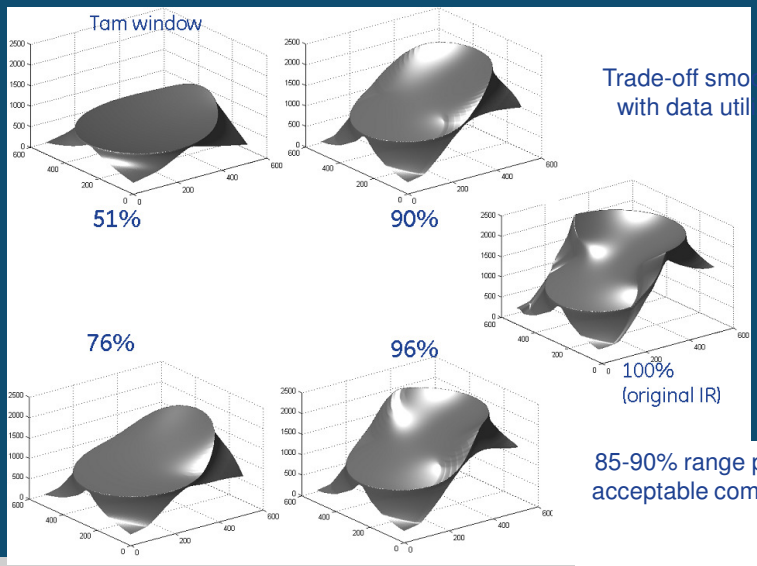
$$\partial BS(c)^- = \partial B(c)^- - st$$





$$win(c,r) = \begin{cases} 1, \ if \ \partial BS(c)^+ > r > \partial BS(c)^- \\ 0, \ if \ \partial BS(c)^+ < r \ or \ r > \partial BS(c)^- \\ \sin^2\left(\dfrac{\min\left(r - \partial SB(c)^+, r - \partial BS(c)^-\right)}{fe} \right) otherwise \end{cases}$$

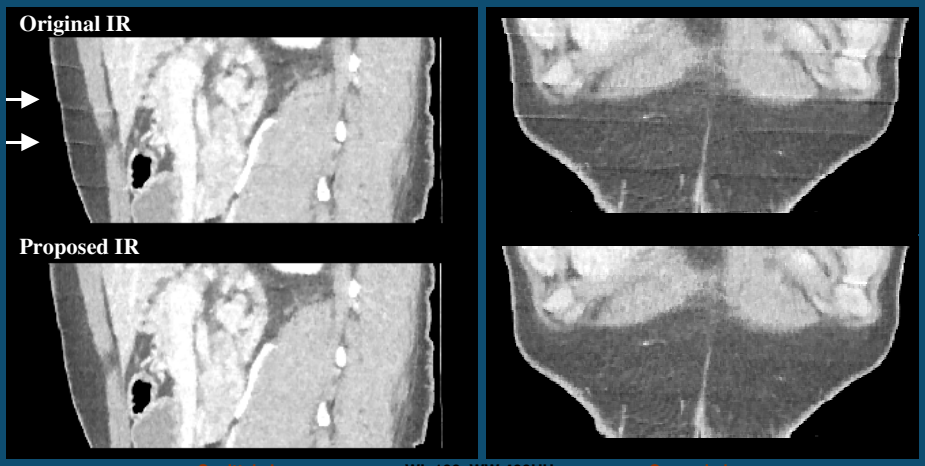
Projection Weighting Methods



Trade-off smoothness with data utilization

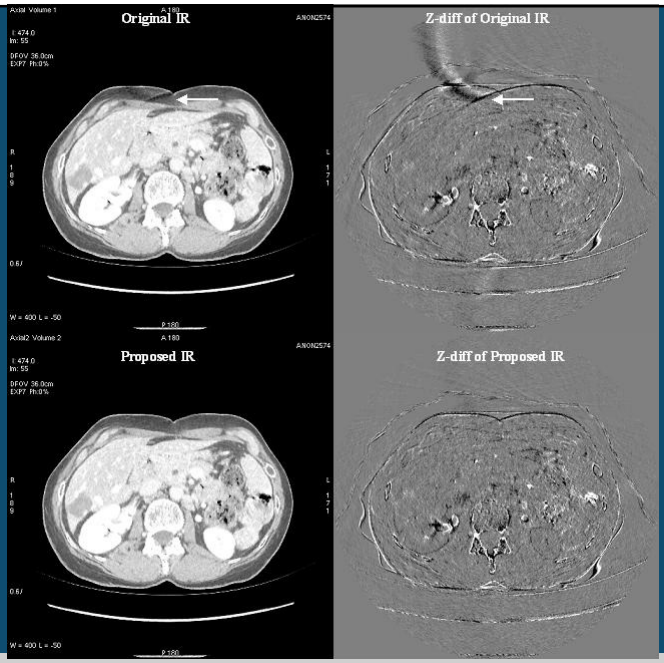
85-90% range provides acceptable compromise

Clinical Results: Reformats



Successful artifact suppression with weighted IR approach

73 Bruno De Man
GE Global Research
IEEE MIC, Valencia, 2011

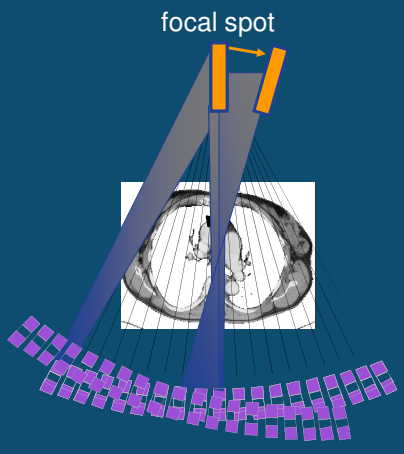


74 Bruno De Man
GE Global Research
IEEE MIC, Valencia, 2011

Overview

- CT basics
- Bayesian framework & noise models
- Forward model – projector/backprojector
- Update step
- Image quality
- Making it work
- → Advanced forward model - Incorporate physics

Finite beam width

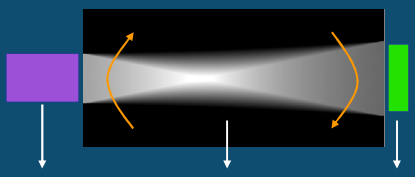


Focal spot :
-Width (x)
-Thermal length
-Target angle
-Off-focal radiation

Detector cell :
-Cell size (in x and z)
-Cross-talk

Azimuthal blur :
-View duration
-Detector response time

Incorporate physics : finite beam width



finite focal spot

finite rotation

finite detector cell

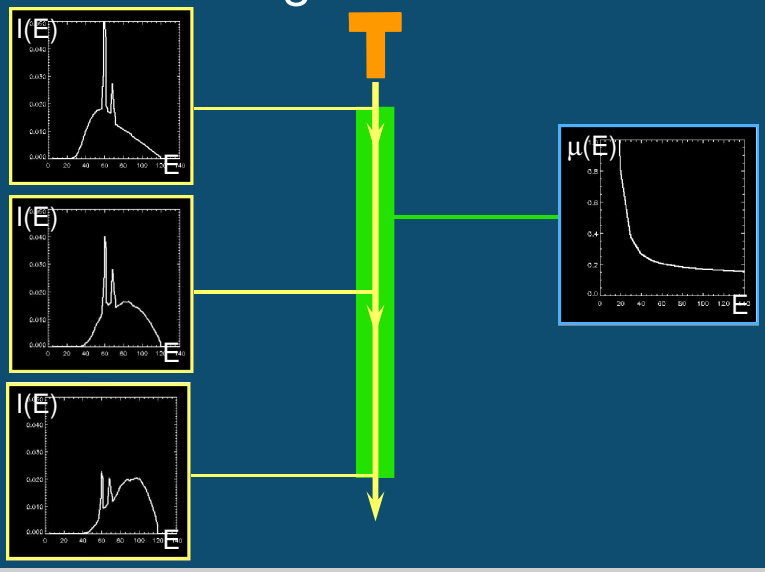
$$\hat{y}_i = \sum_{s=1}^{S} \frac{b_i}{S} \cdot \exp(-\sum_{j=1}^{J} l_{ijs}\mu_j)$$

→ Re-derive update step

77 Bruno De Man
GE Global Research
IEEE MIC, Valencia, 2011

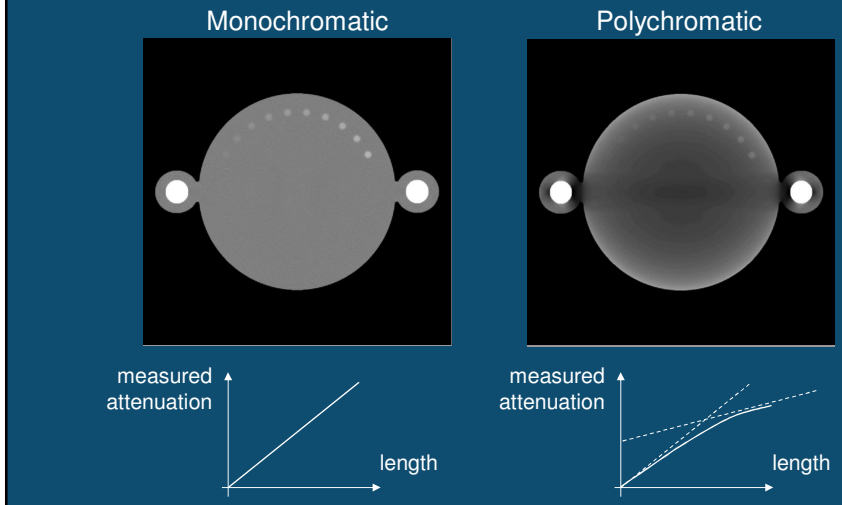
B De Man, Ph.D. Thesis, 2001

Beam hardening



78 Bruno De Man
GE Global Research
IEEE MIC, Valencia, 2011

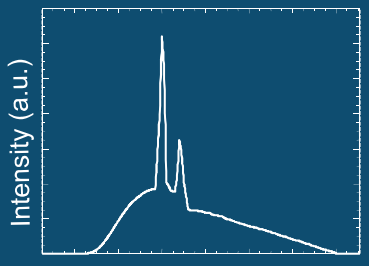
Beam hardening



Bruno De Man
GE Global Research
IEEE MIC, Valencia, 2011

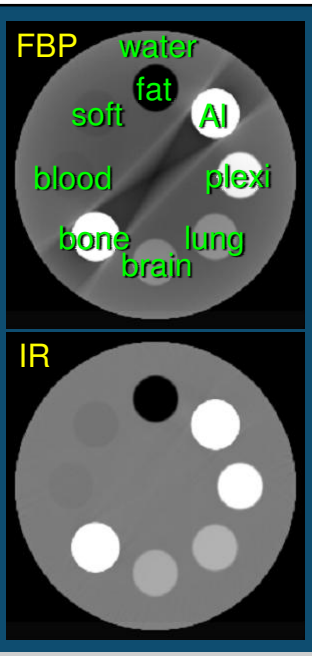
B De Man, IEEE Trans. Nucl. Sci., 1999

Incorporate physics : poly-chromaticity



$$\mu_j(E) = \phi_j \cdot \Phi(E) + \theta_j \cdot \Theta(E)$$

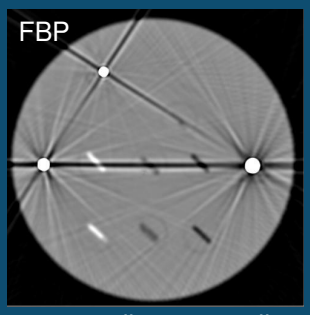
→ Need multi-energy measurements
OR constraint on ϕ_j, θ_j, μ_j
→ Re-derive update step



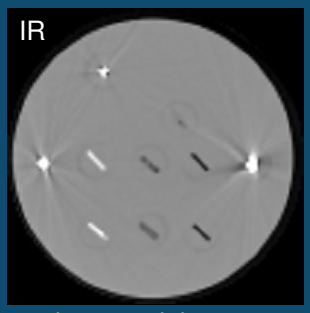
Bruno De Man
GE Global Research
IEEE MIC, Valencia, 2011

B De Man, IEEE Trans. Med. Im., 2001
I Elbakri, IEEE Trans. Med. Im., 2002

Missing data : MAR



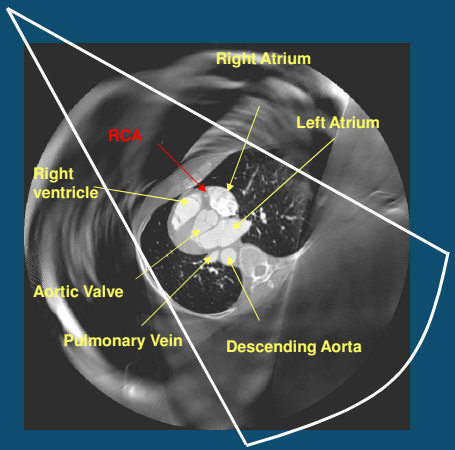
1. treat all rays equally
2. pre-correct for physics



1. lower weight to corrupted rays
2. add prior
(3. incorporate physics)

B De Man, IEEE Trans. Nucl. Sci., 2000

Missing data : truncation or local ROI



Exact reconstruction not possible for the pure local ROI case,
Exact reconstruction IS possible with some extra information

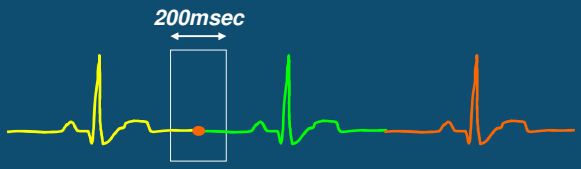
D Langan, SPIE, 2006

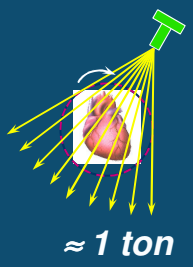
Dynamic imaging (cardiac CT)

rotation period : 0.35s

full-scan = 360° → half-scan ≈ 230°

temporal resolution : 200ms

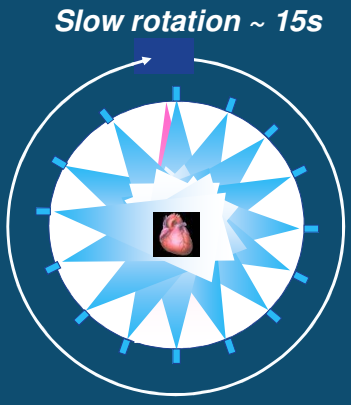


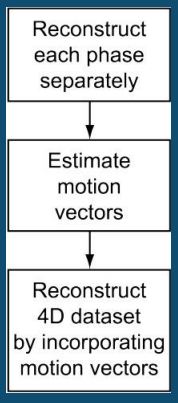


Solutions : (1) spin faster
(3) multiple sources
(2) combine multiple heart beats
(4) motion compensation

Experiment : slow-gantry cardiac CT

- 180bpm (rabbit heart)
- 1 full rotation
- 18s acquisition
- 1500 views
- 1 heart cycle = 1/3sec
- 54 heart cycles
- 28 phases / cycle
- temp.res. ~ 12ms



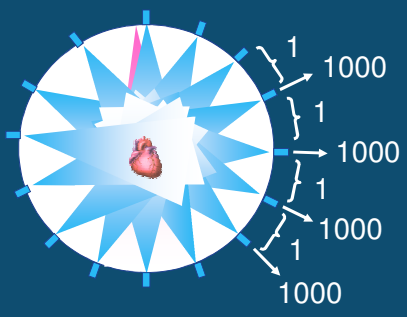


B De Man, Fully 3D Meeting, 2005

Phase-weighted IR

Rationale :
- Using data from 1 phase results in missing data
- Using all phases eliminates temporal info
- Therefore we should use phasic data if available and use other phases where phasic data is missing
- PWMLTR :

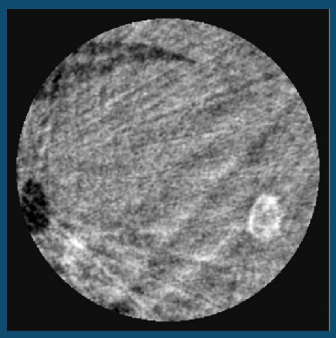
data $\in$ phase $\Rightarrow$ weight = 1000
data $\notin$ phase $\Rightarrow$ weight = 1



85 Bruno De Man
GE Global Research
IEEE MIC, Valencia, 2011

Experiment : slow-gantry cardiac CT

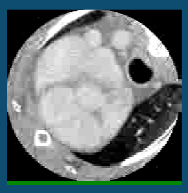
FBP



PW-MLTR

86 Bruno De Man
GE Global Research
IEEE MIC, Valencia, 2011

Experiment : slow-gantry cardiac CT



Overview

- CT basics
- Bayesian framework & noise models
- Forward model – projector/backprojector
- Update step
- Image quality
- Making it work
- Advanced forward model - Incorporate physics

IEEE MIC Reconstruction Short Course 2011

Conditional Statistics in Emission Tomography

Arkadiusz Sitek

asitek@bwh.harvard.edu

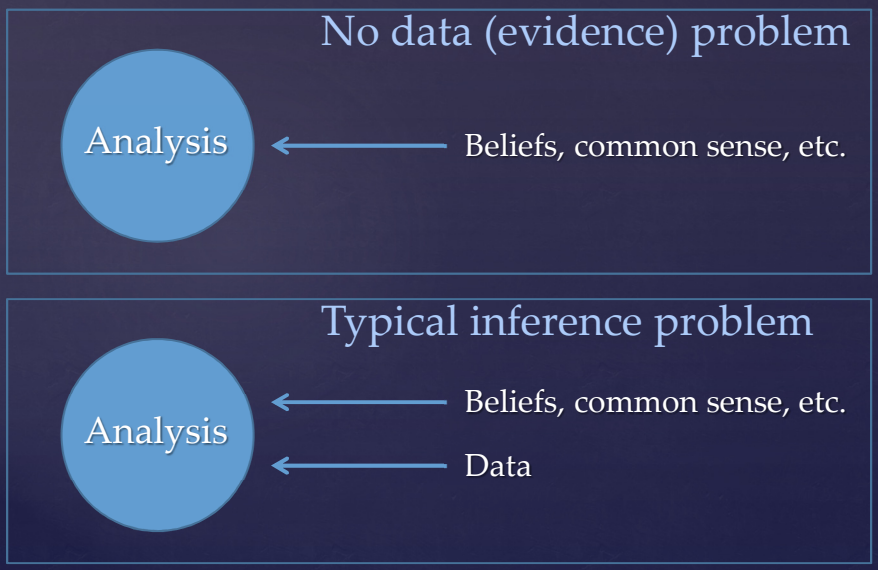
Conditional Analysis in Emission Tomography

Arkadiusz Sitek

Introduction

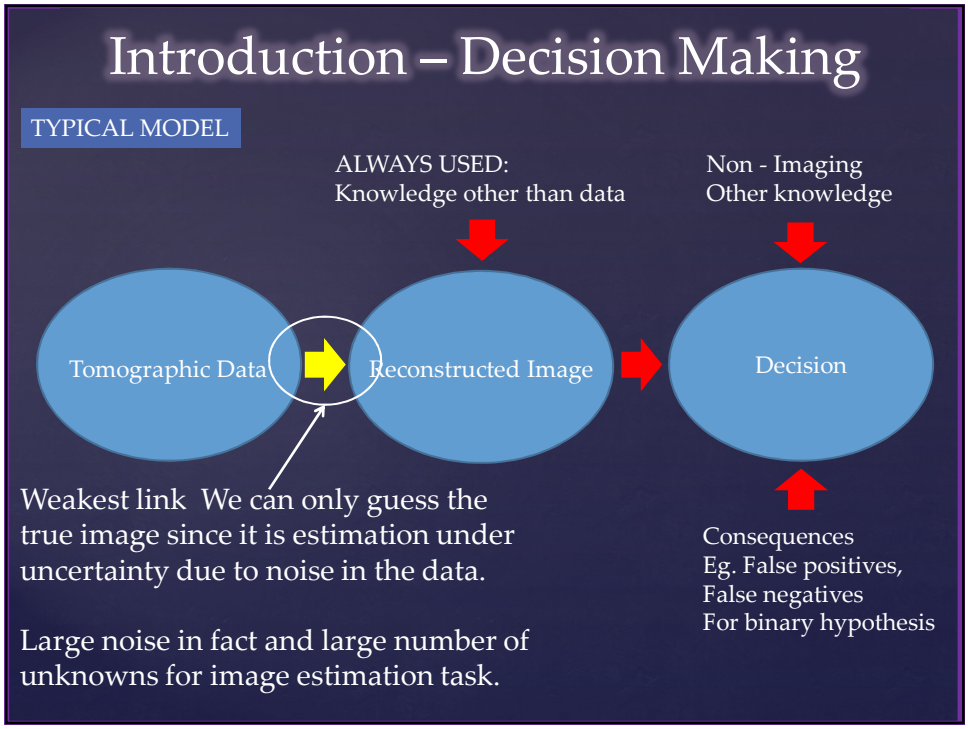
- The ultimate goal of any medical imaging is to reach a decision.
- Examples of decisions:
  - Does patient has a tumor (Yes/No)
  - How many tumors does patient has in his brain (may be classification task 0,1,2,3,…)
  - What is the activity of FDG in some volume of interest (VOI)?
  - Is the blood flow depressed in septal wall of the heart (classification) and by how much (estimation).

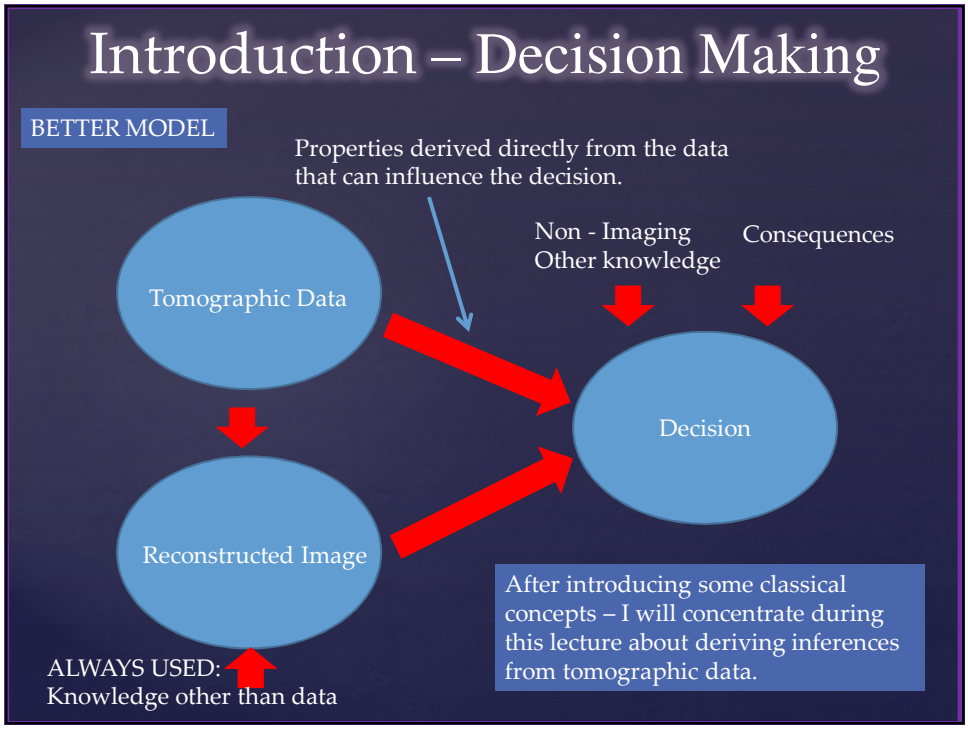
Introduction – Decision Making



Introduction

Tomographic reconstruction is a mid step needed to create an image used for decision making (inference).


Introduction – Decision Making
TYPICAL MODEL
ALWAYS USED:
Knowledge other than data
Non - Imaging
Other knowledge
Tomographic Data
Reconstructed Image
Decision
Weakest link We can only guess the true image since it is estimation under uncertainty due to noise in the data.
Large noise in fact and large number of unknowns for image estimation task.
Consequences
Eg. False positives,
False negatives
For binary hypothesis


Introduction – Decision Making
BETTER MODEL
Properties derived directly from the data that can influence the decision.
Non - Imaging
Other knowledge
Consequences
Tomographic Data
Decision
Reconstructed Image
After introducing some classical concepts – I will concentrate during this lecture about deriving inferences from tomographic data.
ALWAYS USED:
Knowledge other than data

Introduction – Basic Concepts

- For clarity, in this part of the course I always use voxels for image representation.
- I assume that activity (number of **events** that occurred per second) per voxel is the unknown state of nature (parameters).
- I assume that all voxels have volumes 1 and acquisition times are also 1 in which case voxel activity is measured by counts.

Introduction – Basic Concepts

Definition of Event :

Radioactive decay that either directly or indirectly (positrons) produce photons that are detected by ET camera

This may be confused with "event" as defined in statistics.

Introduction – Basic Concepts

Before experiment

Projection data **g** (number of counts in projection bins) and voxel activities **f** are unknown.

After experiment

g – known
f – unknown.

Introduction – Heads up

Conditioning in statistics:

Classical (frequentist) conditioning:

$$p(\mathbf{g}|\mathbf{f})$$

Probability of **g** assuming **f** is true. This makes sense before experiment since both g and **f** are unknown.

Before experiment

g – unknown
f – unknown.

Bayesian conditioning:

$$p(\mathbf{f}|\mathbf{g})$$

The correct way to condition after the experiment since **g** is known, so the statement:
$p(\mathbf{f}|\mathbf{g})$ is a probability of **f** assuming **g** is true really makes sense, doesn't it ?

After experiment

g – known
f – unknown.

Introduction – Basic Concepts

Parametric Statistical model:

In ET the Poisson model is most common (in 1D):

$$pdf(g;f) = \frac{e^{-f} f^{g}}{g!}$$

This describes a family of pdf's – one pdf for a single value of f

This equation is a key to take any inference from data g about the unknown state of nature f.

HOWEVER in medical imaging estimation of f (point estimate, image) is not the end of analysis (at least it should not be) - a clinical decision has to be made based f.

To make a best decision a point estimate of f is not enough.

Introduction – Basic Concepts

pdf = probability distribution function curve – assigns probability (or probability density) of occurrence of outcomes. Make sense ONLY before the experiment !!!

I will not make a distinction in this course between discrete and continues outcomes (probabilities and probability densities) !!! It is always obvious from the context.

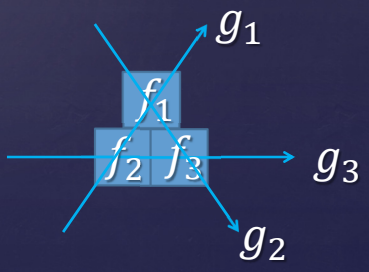
If the outcome is discrete – always probability, else if the outcome is continues – probability density. It is hard to get confused, but nonetheless it is a good idea to watch if anything that we come up with make sense in this respect.

Eg. Statement "Probability that right now I am talking for 10 minutes is 0.1" does not make sense. It is of course 0.

Introduction to SINS Using Classical (Frequentist) Statistics

The Simplest Imaging Nontrivial System (SINS)

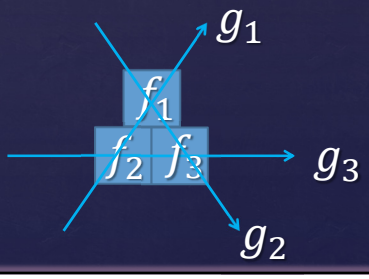
- 3 voxels
- 3 one–bin projections



The Simplest Imaging Nontrivial System (SINS)

- Projection Operation

$$\tilde{g}_1 = 0.5(f_1 + f_2)$$
$$\tilde{g}_2 = 0.5(f_1 + f_3)$$
$$\tilde{g}_3 = 0.5(f_2 + f_3)$$

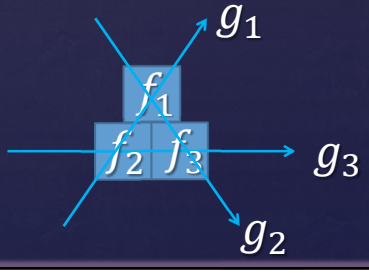


The Simplest Imaging Nontrivial System (SINS)

- Projection Operation

$$\tilde{g}_1 = 0.5(f_1 + f_2)$$
$$\tilde{g}_2 = 0.5(f_1 + f_3)$$
$$\tilde{g}_3 = 0.5(f_2 + f_3)$$

$$\mathbf{f} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix}, \mathbf{g} = \begin{bmatrix} g_1 \\ g_2 \\ g_3 \end{bmatrix}$$

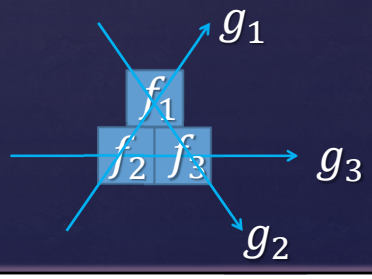


The Simplest Imaging Nontrivial System (SINS)

$$\mathbf{f} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix}, \mathbf{g} = \begin{bmatrix} g_1 \\ g_2 \\ g_3 \end{bmatrix}$$

$$\mathbf{f} = \begin{bmatrix} .5 & .5 & 0 \\ .5 & 0 & .5 \\ 0 & .5 & .5 \end{bmatrix} \mathbf{g}$$

System matrix



The Simplest Imaging Nontrivial System (SINS)

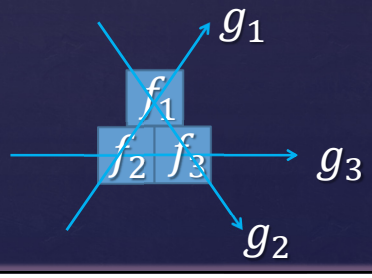
Type equation here.

Sensitivity of each voxel = 1

$$\varepsilon_1 = \varepsilon_2 = \varepsilon_3 = 1$$

$$\mathbf{g} = \begin{bmatrix} .5 & .5 & 0 \\ .5 & 0 & .5 \\ 0 & .5 & .5 \end{bmatrix} \mathbf{f}$$

$= \varepsilon_1$

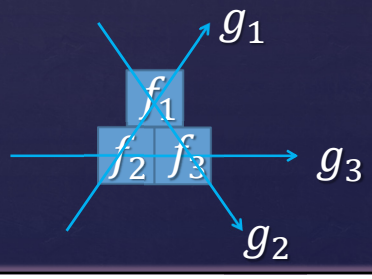


The Simplest Imaging Nontrivial System (SINS)

Reconstruction of SINS (matrix inversion)

$$\mathbf{g} = \begin{bmatrix} .5 & .5 & 0 \\ .5 & 0 & .5 \\ 0 & .5 & .5 \end{bmatrix} \mathbf{f}$$

$$\mathbf{f} = \begin{bmatrix} .5 & .5 & 0 \\ .5 & 0 & .5 \\ 0 & .5 & .5 \end{bmatrix}^{-1} \mathbf{g}$$

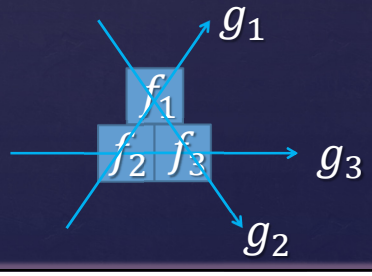


The Simplest Imaging Nontrivial System (SINS)

Reconstruction of SINS (matrix inversion)

$$\mathbf{g} = \begin{bmatrix} .5 & .5 & 0 \\ .5 & 0 & .5 \\ 0 & .5 & .5 \end{bmatrix} \mathbf{f}$$

$$\mathbf{f} = \begin{bmatrix} 1 & 1 & -1 \\ 1 & -1 & 1 \\ -1 & 1 & 1 \end{bmatrix} \mathbf{g}$$



The Simplest Imaging Nontrivial System (SINS)

Reconstruction of SINS Maximum Likelihood

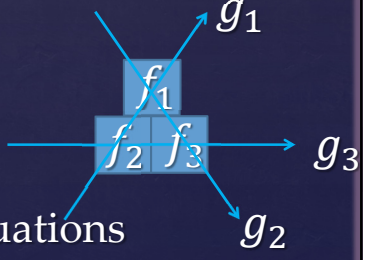
$$\ell(\mathbf{f}|\mathbf{g}) = \frac{e^{-0.5(f_1+f_2)}[0.5(f_1+f_2)]^{g_1}}{g_1!}\frac{e^{-0.5(f_1+f_3)}[0.5(f_1+f_3)]^{g_2}}{g_2!}\frac{e^{-0.5(f_2+f_3)}[0.5(f_2+f_3)]^{g_3}}{g_3!}$$

Function of **f** for observed **g**.

$$\log \ell(\mathbf{f}|\mathbf{g}) = g_1 \log(f_1+f_2) + g_2 \log(f_1+f_3) + g_3 \log(f_2+f_3) - (f_1+f_2+f_3) + Z$$

Solving:

$$\frac{\mathrm{d} \log \ell(\mathbf{f}|\mathbf{g})}{\mathrm{d}\,\mathbf{f}} = 0$$



The above is a set of three equations

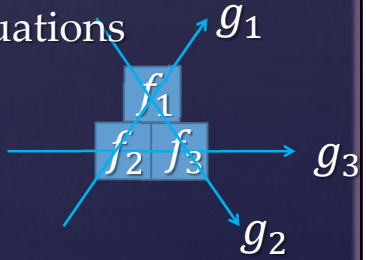
The Simplest Imaging Nontrivial System (SINS)

Solving for ML solution:

$$\frac{\mathrm{d} \log \ell(\mathbf{f}|\mathbf{g})}{\mathrm{d}\,\mathbf{f}} = 0$$

The above is a set of three equations

$$\hat{f}_1 = g_1 + g_2 - g_3$$
$$\hat{f}_2 = g_1 + g_3 - g_2$$
$$\hat{f}_3 = g_2 + g_3 - g_1$$

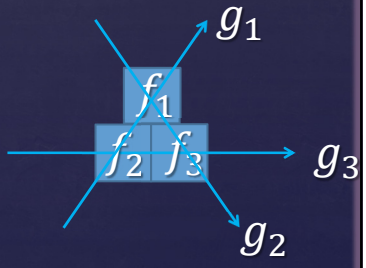


ML estimator same as direct inversion

The Simplest Imaging Nontrivial System (SINS)

$g_1 = 10, g_2 = 30, g_3 = 50$

$\widehat{f_1} = g_1 + g_2 - g_3$
$\widehat{f_2} = g_1 + g_3 - g_2$
$\widehat{f_3} = g_2 + g_3 - g_1$
$\widehat{f_1} = -10$
$\widehat{f_2} = 30$
$\widehat{f_3} = 70$



ML occurs for a negative value of activity $\widehat{f_1}$

The Simplest Imaging Nontrivial System (SINS)

ML occurs for a negative value of activity $\widehat{f_1}$

- Activity has to be positive
- The result is then obviously wrong
- To improve:

$$\frac{\mathrm{d} \log \ell(\mathbf{f}|\mathbf{g})}{\mathrm{d}\, \mathbf{f}} = 0,\ f_i > 0$$

- Constraining to $f_i > 0$ is typically done in ET reconstruction

This applies to all constrained by non-negativity statistical reconstructions

- f_i >0 is typically done in ET reconstruction

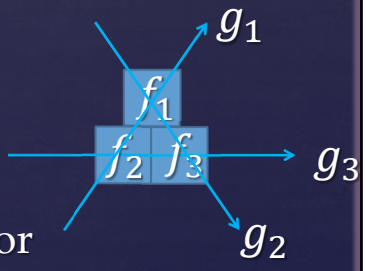
It follows that for ML recons:

- Only constrained ML is reached
- Constrained ML != ML
- ML reconstruction is biased !!!

The Simplest Imaging Nontrivial System (SINS)

- Solving SINS for ML with non-negativity constraints
- Minimize $-\log \ell(\mathbf{f}|\mathbf{g})$ subject to $f_i > 0$

Use Karush-Kuhn-Tucker (KKT) conditions which describe conditions needed for constrained minimum.



SINS Using KKT

- Minimize $-\log \ell(\mathbf{f}|\mathbf{g})$ subject to $f_i > 0$
- v_i = KKT multipliers $i = 1,2, or\ 3$

(1) Gradient is equal to zero
$\nabla \log \ell(\mathbf{f}|\mathbf{g}) + v_1 + v_2 + v_3 = 0$
(2) $v_i > 0$
(3) $f_i > 0$
(4) $v_i f_i = 0$

Solving for $g_1 = 10, g_2 = 30, g_3 = 50$

SINS Using KKT

Solving for $g_1 = 10, g_2 = 30, g_3 = 50$

$\widehat{f_1} = 0, \widehat{f_2} = 22.5, \widehat{f_3} = 67.5$

Take home exercise: Derive of the above
Hints:
- Consider all 8 cases from Eq. (4) $v_i f_i = 0$

$v_0 = 0, f_1 = 0, v_1 = 0$ etc.

SINS Using KKT - Helper

Solving for $g_1 = 10, g_2 = 30, g_3 = 50$ subject to $\widehat{f_i} \geq 0$ for $i = 1,2,3$

$$\widehat{f_1} = 0$$

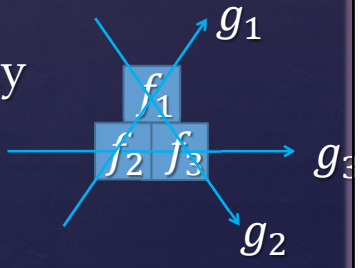
$$\widehat{f_2} = (g_1 + g_2 + g_3)\frac{g_1}{(g_1 + g_2)}$$

$$\widehat{f_3} = (g_1 + g_2 + g_3)\frac{g_2}{(g_1 + g_2)}$$

The Simplest Imaging Nontrivial System (SINS)

It is obvious that for real imaging situation KKT cannot be used directly to obtain ML.

The EM algorithm is most Frequently used as of today



The Simplest Imaging Nontrivial System (SINS)

Expectation Maximization (EM) gives ML for **f** > **0**

$$f_1^{(k+1)} = f_1^{(k)} 0.5 \left(\frac{g_1}{f_1 + f_2} + \frac{g_2}{f_1 + f_3} \right)$$

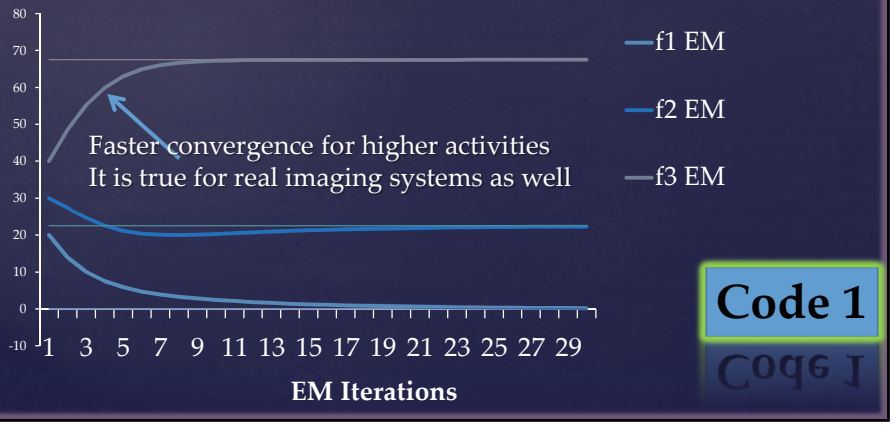
$$f_2^{(k+1)} = f_2^{(k)} 0.5 \left(\frac{g_1}{f_1 + f_2} + \frac{g_3}{f_2 + f_3} \right)$$

$$f_3^{(k+1)} = f_3^{(k)} 0.5 \left(\frac{g_2}{f_1 + f_3} + \frac{g_3}{f_2 + f_3} \right)$$

k – iteration number

The Simplest Imaging Nontrivial System (SINS)

Expectation Maximization (EM) (see the C++ code 1) gives ML for **f** ≥ **0**



Code 1

Classical Estimators in ET

- If taken to convergence the algorithms with non-negativity constraints typically do not converge to ML solution in medical imaging applications.
- The ML is biased especially for low-intensity regions
- In practice EM is never carried to convergence anyway
- Name "EM" should be used instead of "ML-EM" since solutions have not much to do with ML

Classical Estimators in ET

VARIANCE

Variance corresponds to the "spread" of a random variable around the mean.

Higher variance – more uncertain we are about the true value of the mean
Lower variance – *vice versa*.

Classical Estimators in ET

Definition of VARIANCE

RV

$$var(x) = \int p(x)\,(x - \bar{x})^2$$

Mean

x- Random variable (RV) (scalar)
$p(x)$- probability distribution for x
$\int -$ generalized integral (sum/integral) over all outcomes (discrete/continuous)
$\bar{x}$-mean $= \int p(x)\,x$

Classical Estimators in ET

VARIANCE (weighted quadratic deviation from the mean)

For Poisson data:
Variance = Mean

$$\sum_{j=0}^{\infty} (j - \bar{x})^2 \frac{e^{-x}\bar{x}^j}{j!} = \bar{x}$$

Classical Estimators in ET

VARIANCE

Problem in ET: In projection data the mean is unknown. Only the sample from Poisson distribution is known.

Solution: Just use the sample as approximation of the mean. It works.

Higher the number of counts – better the approximation

Classical Estimators in ET

VARIANCE properties:

$$var\left(\sum_{k=1}^{K} a_k x_k\right) = \sum_{k=1}^{K}\sum_{m=1}^{K} a_k a_m cov(x_k, x_m)$$

x_k – random variable

Interesting exercise is to demonstrate the above.

Classical Estimators in ET

$$var\left(\sum_{k=1}^{K} a_k x_k\right) = \sum_{k=1}^{K}\sum_{m=1}^{K} a_k a_m cov(x_k, x_m)$$

True for any *pdf* of x_m.

Hints:

$$cov(x_i, x_j) = \int p(x_i, x_j)\,(x_i - \bar{x}_i)(x_j - \bar{x}_j)$$

$$p(x_i, x_j) = p(x_i|x_j)p(x_j) = p(x_j|x_i)p(x_i)$$

$$\int p(x_i, x_j) = \int p(x_j) = \int p(x_i) = 1$$

Classical Estimators in ET

VARIANCE

Filtered Backprojection reconstruction:

1. Filter the sinogram data
2. Backproject (a linear operation)

Classical Estimators in ET

VARIANCE in FBP reconstruction
General backprojection:

$$\hat{f}_i = \sum_{k=1}^{K} \widetilde{\alpha_{ki}} g_k$$

$\widetilde{\alpha_{ki}}$- "filtered" system matrix (elements may be negative)
g_k −count at projection element k

Classical Estimators in ET

Using previously derived formula:
$var\left(\sum_{k=1}^{K} a_k x_k\right) = \sum_{k=1}^{K} \sum_{m=1}^{K} a_k a_m cov(x_k, x_m)$

$$var(\hat{f}_i) = \sum_{k=1}^{K} \widetilde{\alpha_{ki}}^2 var(g_k)$$

since:
$\sum_{k=1}^{K} \sum_{m=1}^{K} a_k a_m cov(x_k, x_m) =$
$\int_{k=1}^{K} a^2{}_k \, var(x_k)$ for uncorrelated data

Classical Estimators in ET

Also:

$$\sum_{k=1}^{K}\sum_{m=1}^{K} a_k a_m cov(x_k, x_m) = \int_{k=1}^{K} a^2{}_k \, var(x_k)$$

To derive:

1. Use definitions
2. For uncorrelated data:

$$p(x_i, x_j) = p(x_i)p(x_j)$$

Classical Estimators in ET

Confidence Interval (CI) (alternative to variance):

- Estimator $\widehat{f_i}$ and variance $var\left(\widehat{f_i}\right)$
- Assuming Gaussian distribution of $\widehat{f_i}$
- 95% confidence interval for $\widehat{f_i}$

$$\left(\widehat{f_i} - 1.96\sqrt{var(\widehat{f_i})}, \widehat{f_i} - 1.96\sqrt{var(\widehat{f_i})}\right)$$

If we repeat the experiment the true value of f_i will be 95% of times within the 95% CI.

The CIs are different for different data sets

Classical Estimators in ET

Confidence Interval (CI):

- 95% confidence interval for $\hat{f}_i$

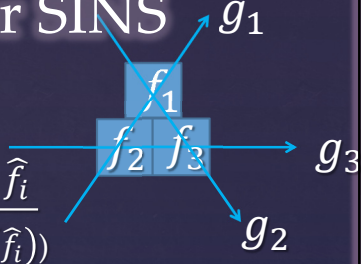
$$\left(\hat{f}_i - 1.96\sqrt{var(\hat{f}_i)}, \hat{f}_i - 1.96\sqrt{var(\hat{f}_i)}\right)$$

If we repeat the experiment the true value of f_i will be 95% of times within the 95% CI.

We assumed Gaussian Distribution for $\hat{f}_i$. Really ?

If this works for SINS, it will work even better for ET due to central limit theorem.

Confidence Interval for SINS



Confidence Interval (CI):

- 95% confidence interval for $\hat{f}_i$

$$\left(\hat{f}_i - 1.96\sqrt{var(\hat{f}_i)}, \hat{f}_i + 1.96\sqrt{var(\hat{f}_i)}\right)$$

Case 1

Variance in projections is known.
Only possible for computer simulations.

Case 2

Variance in projections assumed equal to data (for Poisson only)
Typical assumption used for real data.

Confidence Interval for SINS

Code 2

Confidence Interval (CI):

- 10,000,000 noise realizations
- Unbiased ML estimator (allows negative estimates)

Case 1

Variance in projections is known.
Only possible for computer simulations.

Case 2

Variance in projections assumed equal to data (for Poisson only)
Typical assumption used for real data.

Confidence Int. for ML SINS

Code 2

	True	n[%] True variance	n[%] Sample variance	Δn	Counting Statistics
f_1	100	95.032	94.997	0.007	Poisson
f_2	200	95.028	94.997		
f_3	500	95.021	94.981		
f_1	1	94.830	94.412		Poisson
f_2	2	94.856	95.178		
f_3	5	94.966	93.124		
f_1	100	94.997	94.981	0.007	Gaussian
f_2	200	95.000	94.980		
f_3	500	95.000	94.976		
f_1	1	94.997	92.629		Gaussian
f_2	2	95.000	92.610		
f_3	5	95.000	92.358		

These lower values are due "wider" distribution obtained due to unknown variance and poor estimation due to low count.

n - fraction of times the true mean is contained in 95% confidence interval. It should be 95%.
In general very good predictions (similar results for other CIs – try it at home using Code 2)

Variance estimations for ML SINS True ML, and Non-negative ML

Code 2

Code 3

	True	Mean	Std err	sqrt(var)	Counting Statistics
f_1	100	100.01	0.01	28.28	ML (Code 2)
f_2	200	199.99		28.28	
f_3	500	499.98		28.28	
f_1	100	99.96	0.01	28.18	Non Negative ML (Code 3)
f_2	200	199.95		28.18	
f_3	500	499.96		28.18	
f_1	1	1.000	0.003	2.828	ML (Code 2)
f_2	2	2.000		2.828	
f_3	5	5.000		2.828	
f_1	1	1.392	0.02	1.841	Non Negative ML (Code 3)
f_2	2	1.927	0.02	2.132	
f_3	5	4.692	0.03	2.847	

Exercise (advanced):
It can be shown that ML (unrestricted) for SINS achieves Cramer-Rao bound (CRB) tt is than unbiased with smallest variance.
Show that: ML for SINS achieves CRB (this is not trivial, but hard either)

Classical Estimators in ET

Summary:

1. Linear methods (*FBP Huesman PMB 1984, SVD Kadrmas PMB 1999*) are unbiased with high variances that tend to be uniform in entire image.

2. For FBP variance can be estimated with ease.

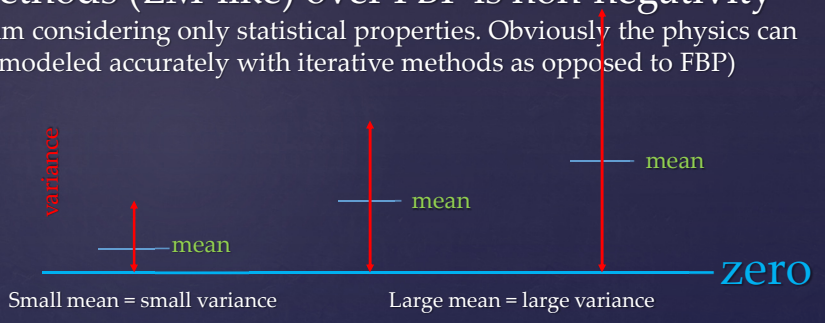
3. A factor making iterative methods (EM-like) superior to FBP is non-negativity
(I am considering only statistical properties. Obviously the physics can be modeled accurately with iterative methods as opposed to FBP)

Classical Estimators in ET

Summary:

3. The factor for superiority of iterative methods (EM-like) over FBP is non-negativity

(I am considering only statistical properties. Obviously the physics can be modeled accurately with iterative methods as opposed to FBP)



- for iterative methods (NN) variance ∝ mean

Classical Estimators in ET

Summary:

4. Iterative methods in ET **DO NOT:**
   - "model the statistics"
   - The "ML" solutions are NOT
     - Unbiased
     - In any way optimal from statistical point of view.
- There are thee main reasons:
  - Global ML cannot be reached due to non-negativity
  - Non-Negative ML are not reached anyway since iterations are terminated.
  - Usually data is not exactly Poisson

Classical Estimators in ET

Summary:
5. Variance estimation with iterative methods is difficult, especially if single data set is available.

There are approximate methods to calculate variances. Up to now they have not found a wide range of applications. Why ?

- They are approximations

(data=mean + other, how about list–mode ?)

- Object dependent (frequentist)

(It was shown in simulations but am I really sure it will work for specific patient)

- Difficult to implement

(Seldom people who publish these methods use them)

- Compute intensive

Barrett et al PMB '84, Fessler IEEE TIP '96 , Qi PMB '06, Li PMB 2011

Classical Estimators in ET

Summary:
5. Variance estimation with iterative methods is difficult, especially if single data set is available.

The way around of the limitation of the single data set is a bootstrapping. New datasets are generated from a single acquired dataset. Works for list–mode.

- Compute intensive, but works pretty well.

Buvat, et al. JNM 2000
Dahlbom TNS 2002

Classical Estimators in ET

Unbiasness – impossible to achieve with NN

Variance: OK measure for system design, but problematic for a single data set.

If we repeat the experiment the true value will be 95% of times within the 95% CI.

Frequentist statistics:

Probability is interpreted as frequency

Classical Estimators in ET

Is there a better way of getting reference from the data ?

Conditional statistics is an alternative to classical approach.

Bayesian Statistics in ET

Bayesian Statistics in ET

In imaging very popular is MAP estimator

- MAP => Bayesian, but Bayesian ~~=>~~ MAP

- Sometimes I will use word "Conditional" because analysis is conditioned on the acquired data. In other words, acquired data is considered constant and randomness is modeled in the unknown parameters (state of nature in our case activities).

Bayesian Statistics in ET

What does it mean to use a Bayesian method

- Prior (subjective) knowledge is used
- A Bayesian decision is a decision made based on the prior beliefs that are modified by the data.

Statistical Inference

Frequentist	Conditional
In estimation: parameters are constant and unknown.	In estimation: parameters are RV (this should surprise you)
Probability is considered as A frequency of a given outcome.	Probability is a measure of a belief (more to come…)
Previous part of the course	

Estimation

Frequentist	Conditional

Methods are often (not always) identical – just the interpretation of the results is (should be) different.

Statistical Inference

Lets assume that we performed patient heart perfusion scan.

Since voxel activities are RV does that mean that activities of voxels in are drawn from some distribution?

NO, wrong interpretation. The above would be frequentist. In conditional view we just assume that underlying activities have some values and quantify this belief with a probability.

Conditional

In estimation: parameters are RV (this should surprise you)

Probability is a measure of a belief

Conditional Statistical Inference

$\mathbf{g}$ – data discrete vector size K
$\mathbf{f}$ – voxel activities (parameters)
cont. vector size I

Both $\mathbf{g}$ and $\mathbf{f}$ are RVs (before experiment)

Often in the literature parameters are denoted as θ. We depart from this convention as we are not try to be general and to signify this we use $\mathbf{f}$.

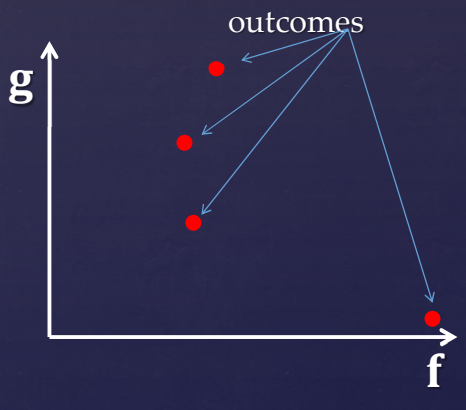
Conditional Statistics

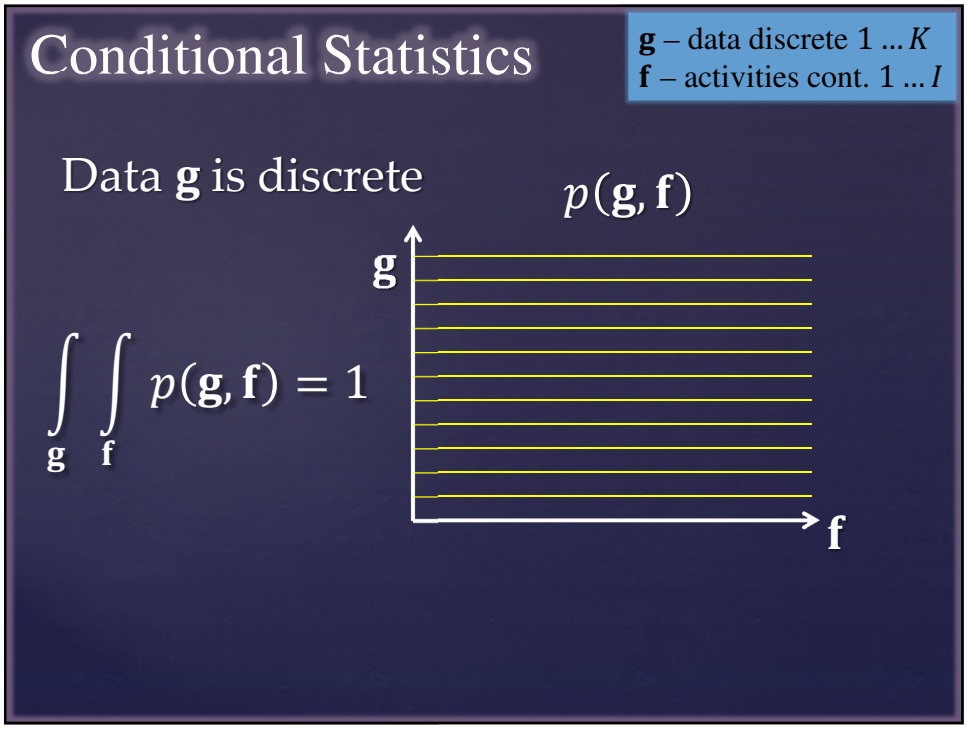
$\mathbf{g}$ – data discrete 1 ... K
$\mathbf{f}$ – activities cont. 1 ... I

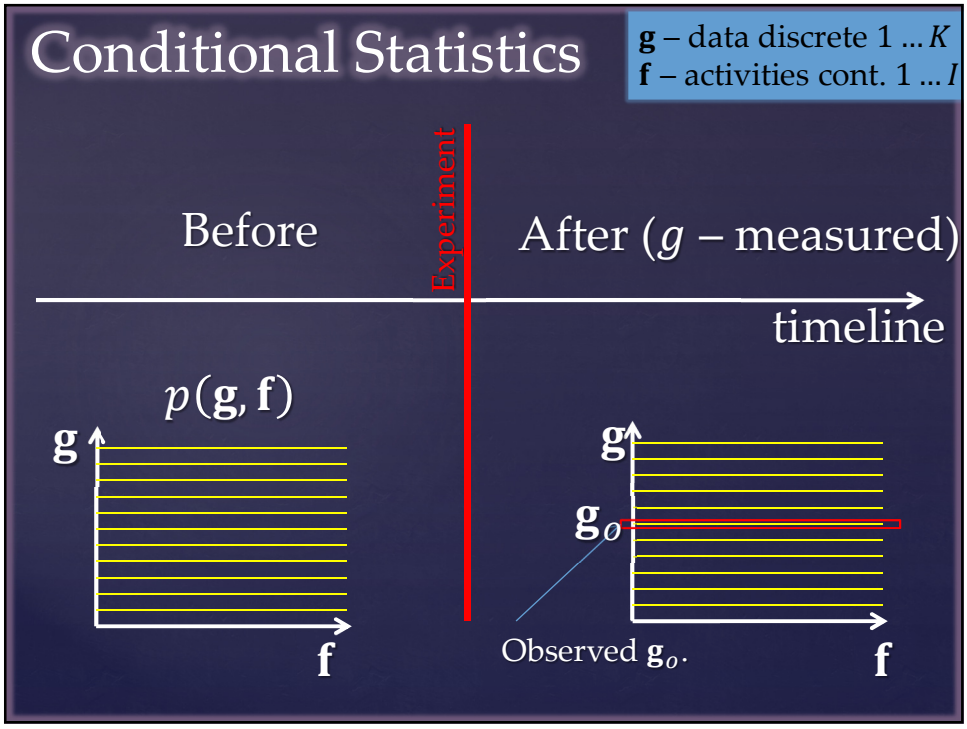
Statistical system is described by a joint probability distribution $p(\mathbf{g}, \mathbf{f})$

Generalized integral (sum or integral depending on context) over all possible outcomes

$$\int_{\mathbf{g}} \int_{\mathbf{f}} p(\mathbf{g}, \mathbf{f}) = 1$$

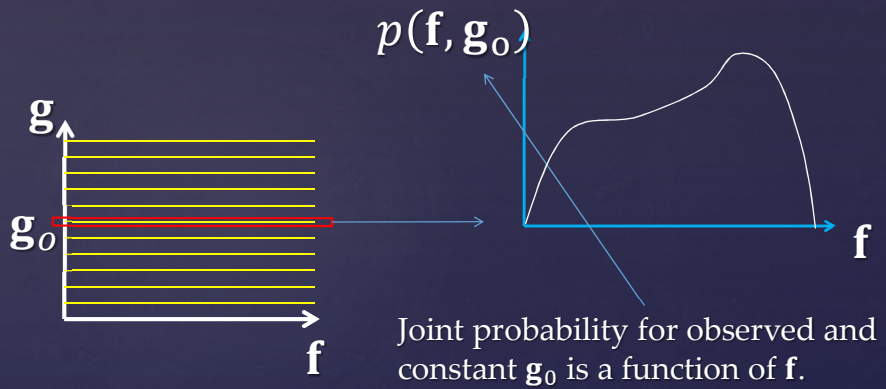



Conditional Statistics
g – data discrete 1 ... K
f – activities cont. 1 ... I
Data g is discrete
p(g, f)
g
f
∫g ∫f p(g, f) = 1


Conditional Statistics
g – data discrete 1 ... K
f – activities cont. 1 ... I
Before
Experiment
After (g – measured)
timeline
p(g, f)
g
f
g
go
f
Observed go.

Conditional Statistics

g – data discrete 1 ... K
f – activities cont. 1 ... I



Joint probability for observed and constant $\mathbf{g}_0$ is a function of **f**.

$$\frac{p(\mathbf{f},\mathbf{g}_o)}{\int_{\mathbf{f}} p(\mathbf{f},\mathbf{g}_o)} = \text{Posterior probability of } \mathbf{f} \text{ for } \mathbf{g}_0$$

Conditional Statistics

g – data discrete 1 ... K
f – activities cont. 1 ... I

$$\frac{p(\mathbf{f},\mathbf{g}_o)}{\int_{\mathbf{f}} p(\mathbf{f},\mathbf{g}_o)} = \text{Posterior probability of } \mathbf{f} \text{ for } \mathbf{g}_0$$

We will denote posteriors as $\mathcal{P}(\mathbf{f}|\mathbf{g}_o)$
The normalization constant = $Z(\mathbf{g}_o)$

$$\mathcal{P}(\mathbf{f}|\mathbf{g}_o) = \frac{p(\mathbf{f},\mathbf{g}_0)}{Z(\mathbf{g}_o)} = \frac{p(\mathbf{g}_0|\mathbf{f})\pi(\mathbf{f})}{Z(\mathbf{g}_o)}$$

$p(x,y) = p(x|y)p(y)$ was used in the above

Conditional Statistics

$\mathbf{g}$ – data discrete $1 \ldots K$
$\mathbf{f}$ – activities cont. $1 \ldots I$

$$\mathcal{P}(\mathbf{f}|\mathbf{g}_o) \propto p(\mathbf{g}_0|\mathbf{f})\pi(\mathbf{f})$$

$p(\mathbf{g}_0|\mathbf{f})$ - probability of data **IF** $\mathbf{f}$ is true

Therefore

$$p(\mathbf{g}_0|\mathbf{f}) \propto \ell(\mathbf{f}|\mathbf{g}_0)$$

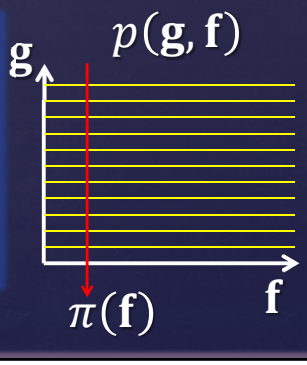
Conditional Statistics

$\mathbf{g}$ – data discrete $1 \ldots K$
$\mathbf{f}$ – activities cont. $1 \ldots I$

$$\mathcal{P}(\mathbf{f}|\mathbf{g}_o) \propto p(\mathbf{g}_0|\mathbf{f})\pi(\mathbf{f})$$

This term is called prior and is unknown.

The key to successful conditional analysis is to find good approximation of the true $\pi(\mathbf{f})$.
(known only to God).



Conditional Statistics

Thinking in "conditional statistics terms":

Our beliefs expressed mathematically by the prior $\pi(\mathbf{f})$

Experiment (scan):

Data $\mathbf{g_0}$ is acquired.
Model of experiment $\ell(\mathbf{f}|\mathbf{g_0})$

Our prior beliefs are confronted with data $\mathbf{g}$ and updated to posterior $\mathcal{P}(\mathbf{f}|\mathbf{g_0})$

Conditional Statistics

$\mathbf{g}$ – data discrete 1 ... K
$\mathbf{f}$ – activities cont. 1 ... I

$\mathcal{P}(\mathbf{f}|\mathbf{g}_0)$

What to do with the posterior once we determined it

In ET the most popular is to determine the maximum (MAP). This is example of point estimation.

OK, but there is nothing special about MAP. No optimality can be claimed, but it is popular because it is relatively easy to determine.

In other words, MAP is just one of many ways the posterior can be used in estimation.

Conditional Statistics

"Fallacy of maximum" or "maximum fallacy"

Reporting maximum of a distribution (ML, MAP) is OK for some concise summary of the problem but is very limited …

1) Ball with what number will be drawn with maximum probability ?
2) Does this mean that if experiment is performed it will be drawn ?

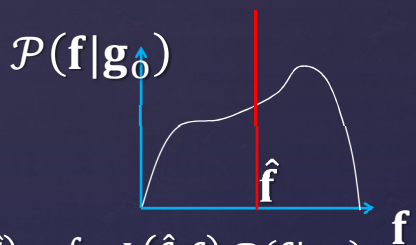
Experiment:
One ball is randomly drawn from the urn

Generalized Bayesian Point Estimation

$\mathbf{g}$ – data discrete $1 \dots K$
$\mathbf{f}$ – activities cont. $1 \dots I$

Loss function

$L(\hat{\mathbf{f}}, \mathbf{f})$



Bayesian Expected Loss $\rho(\hat{\mathbf{f}}) = \int_{\mathbf{f}} L(\hat{\mathbf{f}}, \mathbf{f})\, \mathcal{P}(\mathbf{f}|\mathbf{g}_0)$

By minimizing $\rho(\hat{\mathbf{f}})$ (finding $\hat{\mathbf{f}}$ for which expected loss is smallest) an estimator is obtained.

Depending on the loss function different estimators are obtained.

In conditional analysis posterior is most complete result. Estimator is only a single way to interpret the posterior.

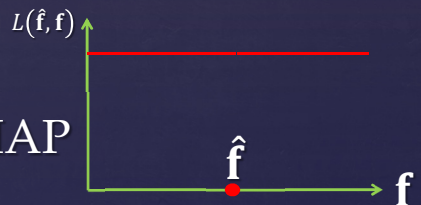
MAP and MMSE

g – data discrete 1 ... K
f – activities cont. 1 ... I

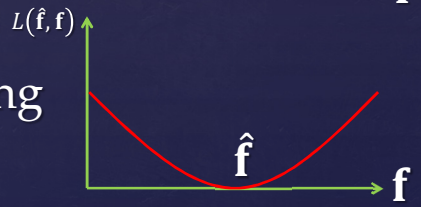
There are many loss functions. In general a special loss functions can be designed for special needs. It is an interesting area of future research in ET ...

Two standard losses:

0-1 loss leading to MAP



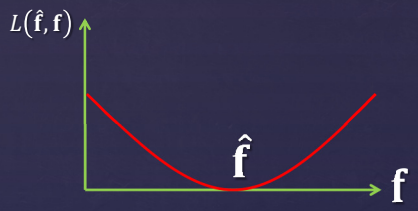
Quadratic loss leading to MMSE



Point Estimation

g – data discrete 1 ... K
f – activities cont. 1 ... I

Quadratic loss leads to Minimum Mean Square Error (MMSE) estimator



$$\rho(\hat{\mathbf{f}}) = \int_{\mathbf{f}} \left(\boldsymbol{f} - \hat{f}\right)^2 \mathcal{P}(\mathbf{f}|\mathbf{g}_o) \qquad (1)$$

Expected loss with quadratic loss function

$$\frac{d\rho(\hat{\mathbf{f}})}{d\hat{\mathbf{f}}} = 0 \qquad (2)$$

We look for the maximum so derivative should be zero

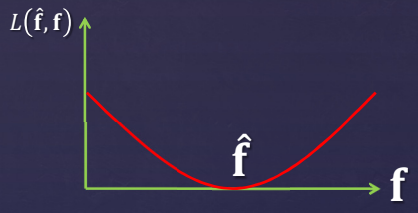
$$\hat{\mathbf{f}} = \int_{\mathbf{f}} \mathbf{f}\, \mathcal{P}(\mathbf{f}|\mathbf{g}_o) \qquad (3)$$

Solving Eq. (2) gives that the estimator with quadratic loss is simple expectation of **f** over the posterior.
Take home exercise: Derive (3) from (2)

Point Estimation

$\mathbf{g}$ – data discrete $1 \ldots K$
$\mathbf{f}$ – activities cont. $1 \ldots I$

Quadratic loss leads to Minimum Mean Square Error (MMSE) estimator



Therefore MMSE estimator = expectation

$$\hat{\mathbf{f}} = \int_{\mathbf{f}} \mathbf{f}\, \mathcal{P}(\mathbf{f}|\mathbf{g}_o)$$

Point Estimation

$\mathbf{g}$ – data discrete $1 \ldots K$
$\mathbf{f}$ – activities cont. $1 \ldots I$

1. Assume a prior $\pi(\mathbf{f})$
2. Have model of the data summarized by likelihood $\ell(\mathbf{f}|\mathbf{g}_o)$

Steps 1 and 2 define the posterior which is a full statistical description of the problem.

3. To make a decision based on the posterior the loss function has to be specified.

Point Estimation Example I

$\mathbf{g}$ – data discrete 1 ... K
$\mathbf{f}$ – activities cont. 1 ... I

1. Assume a prior $\pi(\mathbf{f})$
We only know that activity per voxel should be non-negative
- $\pi(f_i) \propto 1$ for $f_i \geq 0$
- $\pi(f_i) = 0$ for $f_i < 0$

When prior is vague is called *uninformative*
When a prior does not correspond to a distribution it is called *improper*
Note: $\int \pi(f_i) df_i = \infty$

2. Have model of the data summarized by likelihood $\ell(\mathbf{f}|\mathbf{g}_o)$
For Poisson data $\ell(\mathbf{f}|\mathbf{g}_o) \propto \prod_{k=1}^{K} e^{-\tilde{g}_k} (\tilde{g}_k)^{g_k}$, $\tilde{g}_k = \sum_{i=1}^{I} \alpha_{ki} f_i$

3. To make a decision based on the posterior the loss function has to be specified.
Decision: Determine the estimator with 0-1 loss
Numerical problem:

$$\max_{\mathbf{f} \geq 0} \mathcal{P}(\mathbf{f}|\mathbf{g}_o)$$

Point Estimation Example I

Maximum Likelihood estimation with non-negativity constraints is in fact the Bayesian estimation with uninformative improper flat prior.
Belief: All activities >0 are equally likely.

Do we really believe that in every voxel activities equal to zero and activities equal to ∞ are equally likely?

If someone is using ML that is essentially what he/she believes since flat prior indicates such beliefs?

Point Estimation Example I

The fact that ML reconstruction in medical imaging is very noisy is in fact consistent with the prior.

The large variations in activities of the reconstructed images are consistent with the "prior" that assumes that all voxel activities are equally likely.

Reconstructed activities are "spread" over $[0, \infty]$.

Other Examples of Bayesian Point Estimation

Many priors can be investigated

1. Assume a prior $\pi(\mathbf{f})$
   - Roughness prior (Belief: Neighboring voxels should have similar Activities)
   - Total variation prior (Belief: Sum of differences between voxels should be limited)
   - Entropy prior. (Belief: Disorder should be maximized, which indicates that activities should not be concentrated in few voxels)
   - Gaussian and Gamma prior. (Belief: Probabilities of activity values in voxels can be specified independently for each voxel)

   …

Always the same

2. Have model of the data summarized by likelihood $\ell(\mathbf{f}|\mathbf{g}_o)$
   For Poisson data $\ell(\mathbf{f}|\mathbf{g}_o) \propto \prod_{k=1}^{K} e^{-\tilde{g}_k} (\tilde{g}_k)^{g_k}$, $\tilde{g}_k = \sum_{i=1}^{I} \alpha_{ki} f_i$

3. Typically MAP estimator (0-1 loss function) is used, but many others can be investigated.

Bayesian Inference

Point estimations (MAP, MMSE) are examples of the statistical inference.

Bayesian Inference

$\mathbf{g}$ – data discrete 1 ... K
$\mathbf{f}$ – activities cont. 1 ... I

Inference is made based on the posterior

$$\mathcal{P}(\mathbf{f}|\mathbf{g}_o)$$

Posterior has all information but hard to report (Doctors would not like the idea of reporting the posterior ☺)

It has to be summarized in some fashion

Bayesian Inference

$\mathbf{g}$ – data discrete 1 ... K
$\mathbf{f}$ – activities cont. 1 ... I

Examples of the analysis of the posterior

1. Point estimators (MAP, MMSE) which corresponds to image reconstruction
2. Marginalized distributions (extract only clinically relevant information eg. Ratio of activities between the defect and normal myocardium)
3. Credible sets (range of activities of the cardiac defect in which the activity is contained with 95% chance)
4. Hypothesis testing using the posterior

Marginalized distributions

$\mathbf{g}$ – data discrete 1 ... K
$\mathbf{f}$ – activities cont. 1 ... I

Posterior:

$$\mathcal{P}(\mathbf{f}|\mathbf{g}_o) = \mathcal{P}(f_1, f_2, \ldots, f_I|\mathbf{g}_o)$$

Conditional joint distribution of any combination of f_i's can be obtained by marginalizing of other f_i's (integrating out).

$$\mathcal{P}(f_2, f_3, \ldots, f_I|\mathbf{g}_o) = \int df_1 \, \mathcal{P}(f_1, f_2, \ldots, f_I|\mathbf{g}_o)$$

Marginalized distributions

g – data discrete 1 ... K
f – activities cont. 1 ... I

$$\mathcal{P}(f_2, f_4, \dots, f_I|\mathbf{g}_o) = \int df_3 \int df_1 \, \mathcal{P}(f_1, f_2, \dots, f_I|\mathbf{g}_o)$$

1D conditional distributions can be obtained

$$\mathcal{P}(f_I|\mathbf{g}_o) = \int df_1 \int df_2 \dots \int df_{I-1} \, \mathcal{P}(f_1, f_2, \dots, f_I|\mathbf{g}_o)$$

Posterior of activity in a single voxel I

Marginalized distributions

g – data discrete 1 ... K
f – activities cont. 1 ... I

Interesting quantities derived from posterior:
Probability that f_1 activity is twice of f_2 activity
(note medical imaging applications)

$$\mathcal{P}(f_1 > 2f_2|\mathbf{g}_o) = \int df_1 \int_{f_2<f_1/2} df_2 \dots \int df_I \, \mathcal{P}(f_1, f_2, \dots, f_I|\mathbf{g}_o)$$

Conceptually it is a simple concept but demanding computationally in ET.

More to come on this topic

Credible sets

g – data discrete 1 ... K
f – activities cont. 1 ... I

Credible sets are similar to confidence intervals in frequentist statistics.

Reminder: 95% CI indicate a region that will contain the true value 95% of times when different noise realizations are used.

95% credible set contains true value of parameters with 95% "Bayesian certainty"

Credible sets

g – data discrete 1 ... K
f – activities cont. 1 ... I

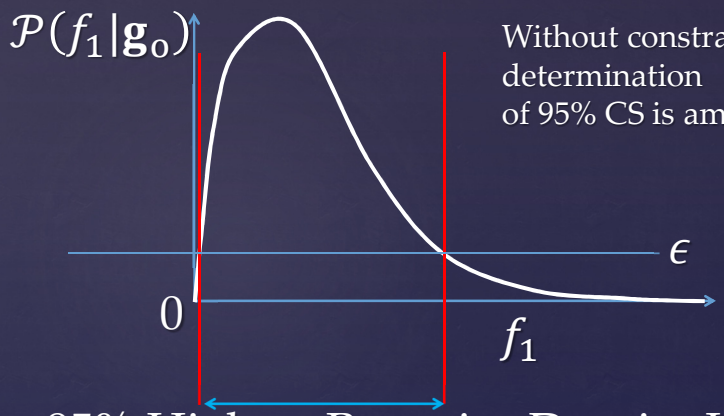
95% credible set contains true value of parameters with 95% "Bayesian certainty".

That is when taken into account prior beliefs and data. When beliefs are incorrect compared to state of nature the CS are objectively WRONG.

Bayesian is subjective.

Credible Sets

g – data discrete 1 ... K
f – activities cont. 1 ... I

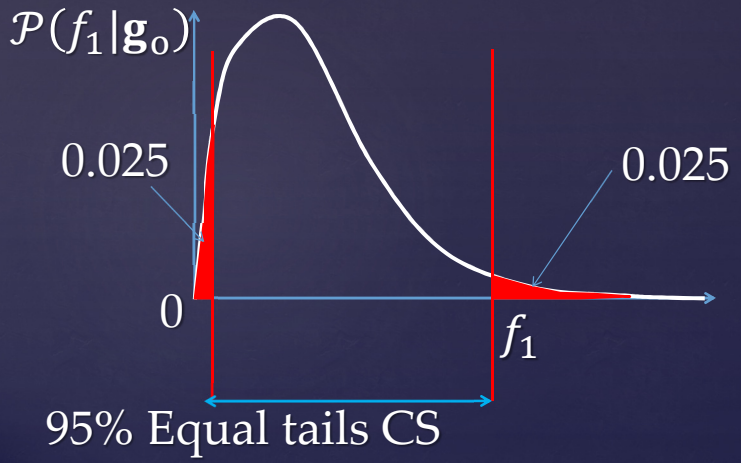


95% Highest Posterior Density HPD CS

In the 95% CS there is no value of posterior less than ϵ and integral of the posterior over the 95% CS is equal to .95. This is not ambiguous. Straightforward extension to multiple dimensions.

Credible Sets

g – data discrete 1 ... K
f – activities cont. 1 ... I



95% Equal tails CS

This can be defined only if there are exactly two tails. Cannot be easily extended to multiple dimensions. In 1D it is not ambiguous.

Bayesian hypothesis testing (detection)

Hypothesis testing in Bayesian setting is conceptually it is very straightforward compared to classical frequentist approach, Neyman-Pearson theorem, P-values)

For conditional approach for binary testing (hypotheses $H_1: f \in \mathbf{F}_1$ and $H_2: f \in \mathbf{F}_2$) the probabilities $p(H_1|\mathbf{g})$ and $p(H_2|\mathbf{g})$ are calculated first.

Bayesian hypothesis testing (detection)

Eg.

$$p(H_1|\mathbf{g}) = \int_{f \in \mathbf{F}_1} p(\mathbf{f}|\mathbf{g})$$

Once $p(H_1|\mathbf{g})$ and $p(H_2|\mathbf{g})$ are calculated decide H_1 for $p(H_1|\mathbf{g}) > p(H_2|\mathbf{g})$ and H_2 otherwise.

$\mathbf{g}$ – data discrete 1 ... K
$\mathbf{f}$ – activities cont. 1 ... I

Bayesian hypothesis testing (detection)

Extensions:

1. Multiple hypothesis testing is easy extention of binary testing. Just calculated integral over posteriors for multiple hypotheses.
2. Extension to decision-theoretic approach to hypothesis testing involving use of loss functions is also straightforward. Loss functions will quantify consequences of decisions.

(see suggested reading for more on this topic)

How all these Bayesian tools can be applied to tomographic data ?

Note that sometimes we do not have to form images from projections in order to derive quantities of interest.

Conditional statistical inference (point estimates (MMSE), marginalized distributions, credible sets, hypothesis testing) require calculation of integrals of the general form:

$$\int_{\mathbf{f}\in\mathbf{F}} \not{f}(\mathbf{f})p(\mathbf{f}|\mathbf{g})$$

where $\not{f}(\mathbf{f})$ is some function of $\mathbf{f}$.
(except MAP)

$\mathbf{g}$ – data discrete $1 \ldots K$
$\mathbf{f}$ – activities cont. $1 \ldots I$

How all these Bayesian tools can be applied to tomographic data ?

Monte Carlo Methods

Monte Carlo Methods in Conditional Statistics

Monte Carlo Methods

- In ET Bayesian methods require evaluation of multi–dimensional posterior distributions.
- In particular calculations of expectations or marginalizations is needed.
- Monte Carlo methods are the best for this task.

Monte Carlo Methods

1. MC integration
2. Markov Chains Monte Carlo (MCMC)
3. MCMC Algorithms (Metropolis, Metropolis–Hastings, Gibbs sampling)
4. Examples of Bayesian analysis using SINS

Monte Carlo Methods

- Monte Carlo Integration

$$\int \mathcal{P}(\mathbf{x})d\mathbf{x}$$

$\mathcal{P}(\mathbf{x})$ – multidimensional function.

First we split $\mathcal{P}(\mathbf{x})$ into two terms:

$$\mathcal{P}(\mathbf{x}) = p(\mathbf{x})\mathcal{R}(\mathbf{x})$$

This can **always** be done ($p(\mathbf{x}) = 1$)

Monte Carlo Methods

- Monte Carlo Integration

$$\int \mathcal{P}(\mathbf{x})d\mathbf{x} = \int p(\mathbf{x})\mathcal{R}(\mathbf{x}) = E^{p(\mathbf{x})}\big(\mathcal{R}(\mathbf{x})\big)$$

$E^{p(\mathbf{x})}(.)$ - expectation over density $p(\mathbf{x})$

Draw samples $x_1, x_2, \ldots, x_S$ from density $p(\mathbf{x})$ than:

$$\int \mathcal{P}(\mathbf{x})d\mathbf{x} \approx \frac{1}{S}\sum_{i=1}^{S} \mathcal{R}\,(\mathbf{x}_i)$$

Note that $\int p(\mathbf{x}) = 1$

Monte Carlo Methods

- Monte Carlo Integration

Draw samples $x_1, x_2, \ldots, x_S$ from density $p(\mathbf{x})$ than:

$$\int \mathcal{P}(\mathbf{x})d\mathbf{x} = \int p(\mathbf{x})\mathcal{R}(\mathbf{x}) \approx \frac{1}{S}\sum_{i=1}^{S} \mathcal{R}(\mathbf{x}_i)$$

Frequent problem with the approach:
- Large computing cost.
- Difficulties to sample from $p(\mathbf{x})$

Monte Carlo Methods

- Drawing from multi–dimensional distributions is difficult and time consuming.
- What if we came up with easy algorithm to generate the samples?
- Can these samples be related to integral at interest somehow?
- Yes they can using Markov Chains !

Markov Chains

- We have a $\mathbf{x}_0$ which is a sample from distribution $\mathcal{P}(\mathbf{x})$
- We come up with algorithm to generate next sample $\mathbf{x}_{i+1}$ based on $\mathbf{x}_i$, but not on any previous samples $\mathbf{x}_0$ ... $\mathbf{x}_{i-1}$
- If we do, we have a **Markov Chain** !

Markov Chains

- Markov Chain MC
  - Ergodicity
  - Detailed balance
- If these are satisfied MCMC reaches steady state and Bayesian integrals (sums) can be estimated.

Markov Chains

- Ergodicity = From any state all other states are accessible.
- The algorithm used to produce new states in the chain has to have a non-zero probability of reaching of any other state (not necessarily in a single move).

Markov Chains

Detailed balance

- $\mathbf{f}^s$- vector of activities in the image. State s in the Markov chain of states
- $\mathbf{f}^{s+1}$- next state in the chain
- $\pi^*(\mathbf{f}^s)$ and $\pi^*(\mathbf{f}^{s+1})$ are probabilities of these states (eg. posterior probabilities)

$$\pi^*(\mathbf{f}^s)P(\mathbf{f}^s \rightarrow \mathbf{f}^{s+1}) = \pi^*\left(\mathbf{f}^{s+1}\right)P(\mathbf{f}^{s+1} \rightarrow \mathbf{f}^s)$$

$P(\mathbf{f}^s \rightarrow \mathbf{f}^{s+1})$ transition probability from s to $s+1$

Markov Chains

$$\pi^*(\mathbf{f}^s)P(\mathbf{f}^s \to \mathbf{f}^{s+1}) = \pi^*\left(\mathbf{f}^{s+1}\right)P(\mathbf{f}^{s+1} \to \mathbf{f}^s)$$

Based on the detailed balance we can come up with the following algorithm:

1. Select a new "convenient" state $s+1$
2. Accept a new state with

$$P(\mathbf{f}^s \to \mathbf{f}^{s+1}) = \max\left(1, \pi^*\left(\mathbf{f}^{s+1}\right)/\pi^*(\mathbf{f}^s)\right)$$

otherwise $\mathbf{f}^{s+1}=\mathbf{f}^s$

METROPOLIS ALGORITHM

Markov Chains – Metropolis

1. Select a new "convenient" state $s+1$
2. Accept a new state with

$$P(\mathbf{f}^s \to \mathbf{f}^{s+1}) = \max\left(1, \pi^*\left(\mathbf{f}^{s+1}\right)/\pi^*(\mathbf{f}^s)\right)$$

otherwise $\mathbf{f}^{s+1}=\mathbf{f}^s$

Markov Chains – Metropolis

Example: Calculate Z

Code 4

$$Z = \int_0^1 dx \int_0^1 dy \; f(x,y)$$

We note that:

$$1 = \int_0^1 dx \int_0^1 dy \; 1$$

$$= \int_0^1 dx \int_0^1 dy \; f(x,y)\frac{1}{f(x,y)} = Z\int_0^1 dx \int_0^1 dy \; f_n(x,y)\frac{1}{f(x,y)}$$

Where $f_n(x,y)$ is normalized $f(x,y)$ such that $\int_0^1 dx \int_0^1 dy \; f_n(x,y)$=1.

Markov Chains

Other sampling algorithms:

Metropolis-Hastings – generalization of the Metropolis algorithm. Selection and Acceptance probabilities are used.

Gibbs algorithm – a special case of Metropolis Hastings algorithm.

1. Next Markov step is crated by drawing from 1D distributions assuming others are constant.
2. Selection probability is such, that acceptance is always 1. Step is always accepted.

Markov Chains – Metropolis

Example:

Code 4

$$1 = Z \int_0^1 dx \int_0^1 dy \; f_n(x,y) \frac{1}{f(x,y)}$$

Sampling density realized by Metropolis algorithm

Function to be integrated by MCMC

Using the code:

$\int_0^1 dx \int_0^1 dy \; f_n(x,y) \frac{1}{f(x,y)} = 4$ for $f(x,y) = xy$ => Z=0.25

$\int_0^1 dx \int_0^1 dy \; f_n(x,y) \frac{1}{f(x,y)} = 9$ for $f(x,y) = x^2y^2$ => Z=0.111(1)

$\int_0^1 dx \int_0^1 dy \; f_n(x,y) \frac{1}{f(x,y)} = 1.24631$ for $f(x,y) = e^{-\sin(xy)}$ => Z=0.80237±0.00005

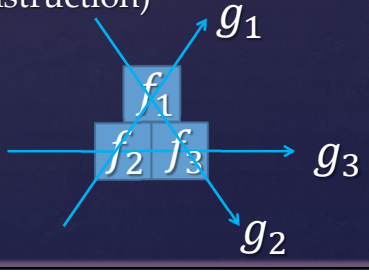
To see how standard error was calculated see the code

MCMC in Tomography

How to apply all this to tomography:

We will now perform Bayesian analysis

1. Prior distribution (what we belief in before experiment)
2. Posterior (our beliefs adjusted by the data)
3. What can be derived from the posterior
   1. Point estimates (image reconstruction)
   2. Credible sets
   3. Hypothesis testing



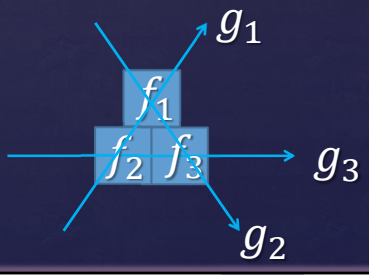
MCMC in ET – MMSE

Code 5

PRIOR

We assume that no information is available about the activities f_1, f_2, f_3 therefore we assume a non-informative prior for

$\pi(f_1, f_2, f_3) \propto 1$ for $f_1 > 0$, $f_2 > 0$, f_3 and 0 otherwise



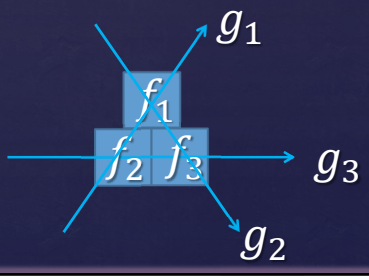
MCMC in ET – MMSE

Code 5

POSTERIOR

Experiment is performed and data g_1, g_2, g_3 observed. Posterior is formed using Bayes theorem.

$$\mathcal{P}(\mathbf{f}|\mathbf{g}) \propto \ell(\mathbf{f}|\mathbf{g})$$



MCMC in ET – MMSE

Code 5

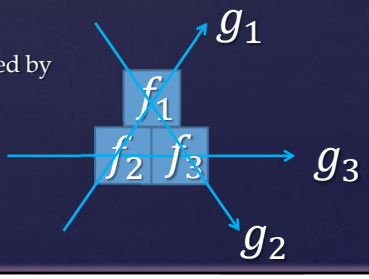
MMSE estimator of **f**

Experiment is performed and data g_1, g_2, g_3 observed. Posterior is formed using Bayes theorem.

$$\hat{\mathbf{f}} = \int_{-\infty}^{\infty} df_1 \int_{-\infty}^{\infty} df_2 \int_{-\infty}^{\infty} df_3 \; \mathbf{f} \; \mathcal{P}(\mathbf{f}|\mathbf{g})$$

Function to be integrated by MCMC

Sampling density realized by Metropolis algorithm



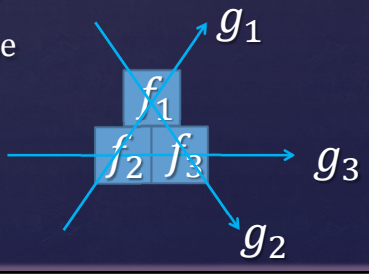
MCMC in ET – MMSE

Code 5

cap-PRIOR

We believe that we know the maximum value of activities in voxels which is $f_{\max}$. Activities are otherwise equally likely. Bounding makes the prior proper and full form can be specified:

$$\pi(f_1, f_2, f_3) = \begin{cases} \dfrac{1}{(f_{\max})^3} & \text{for } f_{\max} > f_i > 0 \\ 0 & \text{otherwise} \end{cases}$$



MCMC in ET

Code 5

Comparison of estimators for $g_1 = 10, g_2 = 30, g_3 = 50$

	$\hat{f}_1$	$\hat{f}_2$	$\hat{f}_3$
ML	-10	30	70
ML+NN	0	22.5	67.5
MAP (NN and B90)	0	22.5	67.5
MMSE NN	5.8	20.2	67.2
MMSE (NN and B90)	5.8	20.3	66.8
MMSE (NN and B50)	6.8	26.9	46.6

NN – non-negativity
BXX – Upper bound prior with maximum value of XX.

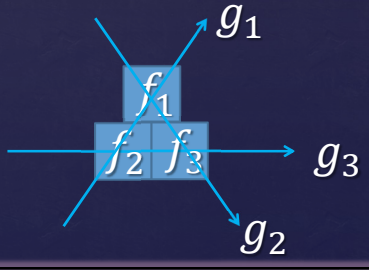
MCMC in ET – Marginalization

Code 6

Marginalization

Suppose we are interested in a posterior distribution of activity in voxel 1 $p(f_1|\mathbf{g})$. This can be done by marginalization (integration out of other variables)

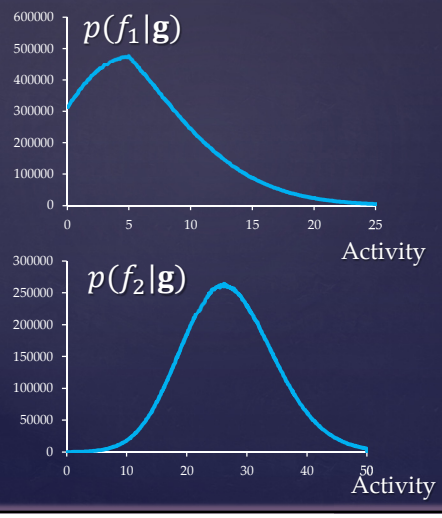
$$p(f_1|\mathbf{g}) = \int_{-\infty}^{\infty} df_2 \int_{-\infty}^{\infty} df_3 \; p(\mathbf{f}|\mathbf{g})$$

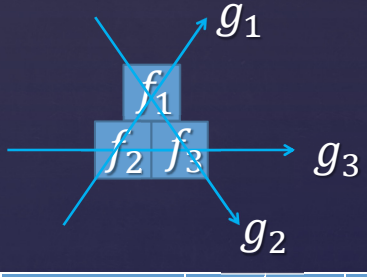


MCMC in ET – Marginalization

Code 6

Marginalization (example with prior B50)



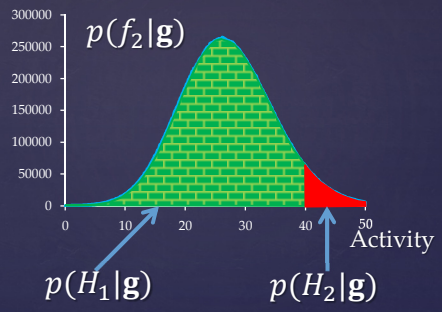


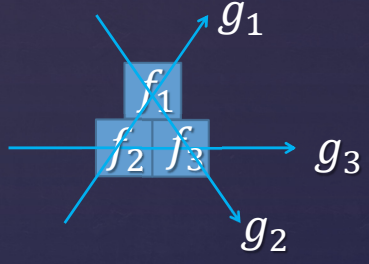
	$\hat{f}_1$	$\hat{f}_2$	$\hat{f}_3$
ML	-10	30	70
ML+NN	0	22.5	67.5
MAP (NN and B90)	0	22.5	67.5
MMSE NN	5.8	20.2	67.2
MMSE (NN and B90)	5.8	20.3	66.8
MMSE (NN and B50)	6.8	26.9	46.6

Hypothesis testing

H_1 - activity $f_2 < 40$

H_2 - activity $f_2 \geq 40$





For applications in ET:
think about SUV (standardized uptake value)

Decide H_1 if $p(H_1|\mathbf{g}) \geq p(H_2|\mathbf{g})$

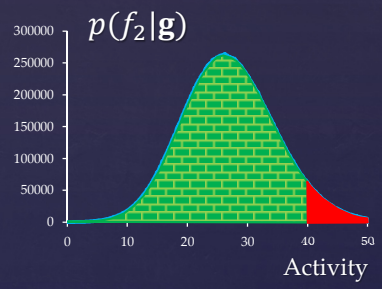
Hypothesis testing

H_1 - activity $f_2 < 40$
H_2 - activity $f_2 \geq 40$

$p(H_1|\mathbf{g}) = 0.95$
$p(H_2|\mathbf{g}) = 0.05$

Posterior odds in favor of H_1
$\frac{p(H_1|\mathbf{g})}{p(H_2|\mathbf{g})} \approx 19$

Prior odds in favor of H_1
$\frac{p(H_1)}{p(H_2)} = 4$ (since we assumed 0-50)



Bayes Factor = posterior odds / prior odds
$B = \frac{p(H_1|\mathbf{g})}{p(H_2|\mathbf{g})} / \frac{p(H_1)}{p(H_2)} \approx 5$

Odds representing support for H_1 by the data only (note that it is also dependent on the prior, but hopefully B is not very sensitive. Sensitivity analysis can be done.

Bayes factor aka marginalized likelihood ratio or simply likelihood ratio.

Decision Theory

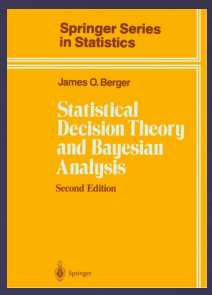
To apply the decision theory losses (costs, risks) need to be specified. These losses are specified for wrong decisions.
Eg. Deciding H_1 when H_2 is true infer a loss.

For binary hypothesis only two values of the loss (assuming independence of the loss on **f**.
K_1 - loss if H_2 is true and H_1 was decided
K_2 - loss if H_1 is true and H_2 was decided

Total loss for H_1 = $K_1 p(H_2|\mathbf{g})$
Total loss for H_2 = $K_2 p(H_1|\mathbf{g})$

Decide hypothesis for which the total loss is smallest.
Easy extension to multiple hypothesis testing.

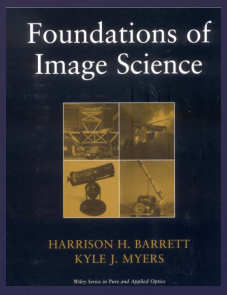
Decision Theory



My highest recommendations:

James O. Berger "Statistical Decision Theory and Bayesian Analysis"

General statistical theory



For application in imaging:

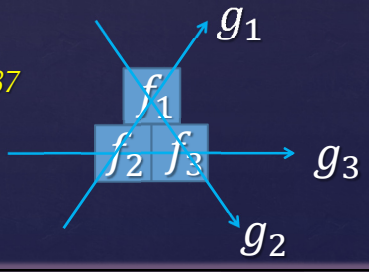
Caution: This book is not for beginners. Authors use pragmatic approach and mix Bayesian with frequentist approaches as needs arise. This may be quite confusing.
Make sure you have firm understanding of statistics before reaching for this book (do Berger first).

MCMC in ET

Applications of Bayesian inference using MCMC for more realistic tomographic problems (pretty easy extension of codes provided – at least conceptually).

Hierarchical priors (I have not talked about but learn it first from Berger)

Geman & McClure Bull Int Stat Inst 1987
Weir JASA 1996
Higdon et al. IEEE TMI 1997
Kupinski et al JOSA 2003

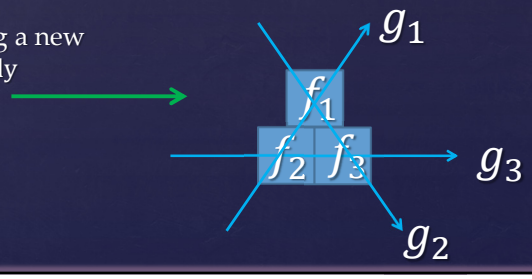


MCMC in ET

Unfortunately for each step of Markov chain the likelihood and prior needs to be recomputed. This is enormous cost even if computations are partial.

Yes lots of computing, but don't we have GPUs now …

(Partial recomputing: if creating a new Markov state by changing f_1 only projections for g_1 and g_2 needs to be recomputed)

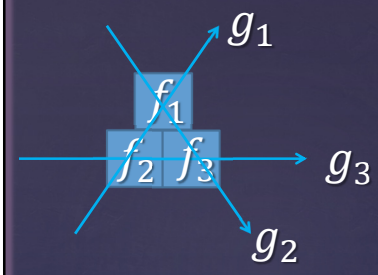


MCMC in ET Using Origin Ensembles

Use of Origin Ensembles obviates the need of calculation of the likelihood AND the prior in each Markov step. In fact the likelihood and prior are NEVER explicitly evaluated.

For more on the OE algorithm see
Sitek PMB 2008, Sitek IEEE TMI 2011
For statistical theory of OE
if interested send me email
asitek@bwh.harvard.edu

MCMC in ET Using Origin Ensembles

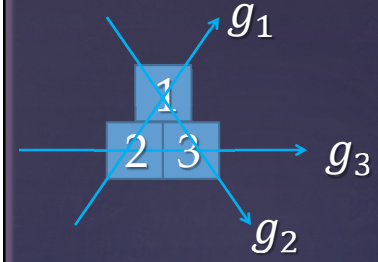


New RV (emission count)
Number of counts emitted in voxel i and detected in projection bin $j = h_{ki}$

h – vector of size 9 (3 by 3)

The idea: derive the posterior: $\mathcal{P}(\mathbf{h}|\mathbf{g})$ and use it for statistical inference (point estimators, marginalizations, Likelihood ratios) and for decision-theoretic approaches (loss functions).

MCMC in ET Using Origin Ensembles



New RV (emission count)
Number of counts emitted in voxel i and detected in projection bin $j = h_{ki}$

h – vector of size 9 (3 by 3)

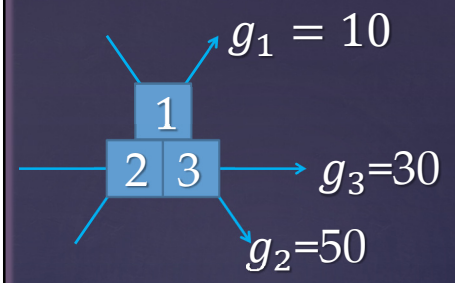
Posterior $p(\mathbf{h}|\mathbf{g})$ = No dependence on **f.** Discrete Distribution. Assuming non-informative prior:

$$\mathcal{P}(\mathbf{h}|\mathbf{g}) \propto \frac{c_1!c_2!c_3!}{h_{11}!h_{12}!h_{13}!h_{21}!h_{22}!h_{23}!h_{31}!h_{32}!h_{33}!} \quad \text{for} \quad \sum_{i=1}^{3} h_{ki} = g_k$$

where $c_i = \sum_{k=1}^{3} h_{ki}$

To calculate MMSE of $\hat{\mathbf{h}} = \sum_h \mathbf{h} p(\mathbf{h}|\mathbf{g})$ the OE can be used:

MCMC in ET Using Origin Ensembles



$$c_i = \sum_{k=1}^{3} h_{ki}$$

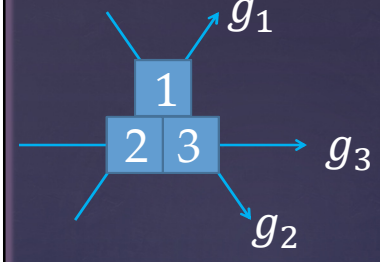
1. Randomly select detected event (1 out of 90) and note in which voxel it is contained (say i)
2. Randomly select a new voxel in which it could have been detected (say i')
3. Move from i to i' with a chance:

$$\frac{c_{i'} + 1}{c_i}$$

4. Goto 1

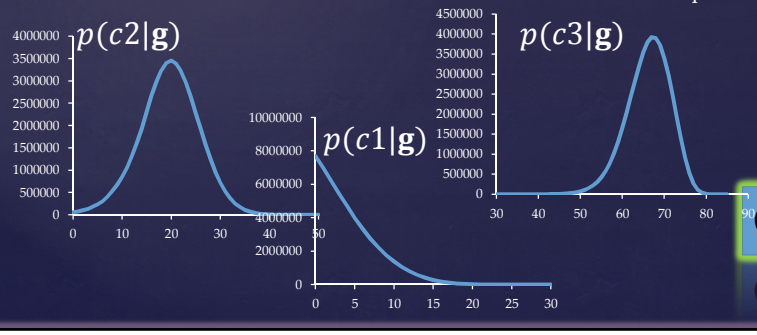
Code 7

MCMC in ET Using Origin Ensembles



	$\hat{f}_1$	$\hat{f}_2$	$\hat{f}_3$
ML	-10	30	70
ML+NN	0	22.5	67.5
MAP (NN and B90)	0	22.5	67.5
MMSE NN	5.8	20.2	67.2
MMSE (NN and B90)	5.8	20.3	66.8
MMSE (NN and B50)	6.8	26.9	46.6
OE MMSE (NN)	4.1*	19.6*	66.2*

*Estimates of number of emissions per voxel



Code 7

Summary

I have not shown a single image !!!

Is it an image worth a thousand words ?

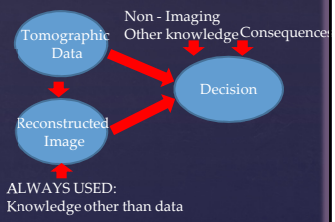
It is, but for each person it is worth different thousand words.

Assessment of the images by humans are just too subjective.

Summary

Reconstructed image should never be considered alone.

Measures of uncertainty derived from the data should also be used. This is make the decision less subjective.



However – there is no such thing as fully objective decision under uncertainty. Some level od subjectivity will always be present (eg: significance levels for classical statistics or priors for Bayesians etc.)

IEEE MIC Reconstruction Short Course 2011

Appendix

Demonstration C++ Codes

C:\Users\asitek\Desktop\Course\MaxLikelihood\MaxLikelihood.cpp 1

```
1  // Copyright (C) A. Sitek
2  // Finds ML for SINS
3  // Code 1
4  #include <stdio.h> // for printf
5
6  int main(int argc, char* argv[])
7  {
8      const double g1 = 10;
9      const double g2 = 30;
10     const double g3 = 50;
11
12     double f1_hat = 1;
13     double f2_hat = 1;
14     double f3_hat = 1;
15
16     const int iterations = 1000;
17
18     FILE *wsk = fopen("value.txt","w");
19     for(int it=0;it<iterations;it++)
20     {
21         // projection
22         double expectation_g1   = 0.5*(f1_hat+f2_hat);
23         double expectation_g2   = 0.5*(f1_hat+f3_hat);
24         double expectation_g3   = 0.5*(f2_hat+f3_hat);
25         // invert expectations to avoid division by zero
26         if(expectation_g1>0) expectation_g1 = 1/expectation_g1;
27         else expectation_g1=0;
28         if(expectation_g2>0) expectation_g2 = 1/expectation_g2;
29         else expectation_g2=0;
30         if(expectation_g3>0) expectation_g3 = 1/expectation_g3;
31         else expectation_g3=0;
32
33         //backprojection
34         double update_g1        =   0.5*(g1*expectation_g1 + g2*expectation_g2);
35         double update_g2        =   0.5*(g1*expectation_g1 + g3*expectation_g3);
36         double update_g3        =   0.5*(g2*expectation_g2 + g3*expectation_g3);
37
38         f1_hat *= update_g1;
39         f2_hat *= update_g2;
40         f3_hat *= update_g3;
41         fprintf(wsk,"%f %f %f\n",f1_hat,f2_hat,f3_hat);
42     }
43     fclose(wsk);
44     printf("f1=%.5f\nf2=%.5f\nf3=%.5f\n",f1_hat,f2_hat,f3_hat);
45     return 0;
46 }
47
48
```

C:\Users\asitek\Desktop\Course\CovarianceML\CovarianceML.cpp 1

```
1  // Copyright (C) A. Sitek
2  // SINS Covariance
3  // Code 2
4  #include <stdio.h>
5  #include <math.h>
6  #include <memory.h>
7  #include "..\Common\NumRecipes.h"
8
9  using namespace num_recipesAS;
10 // uncomment below for Gaussian statistics - otherwise Poisson
11 //#define GAUSS
12
13 int main(int argc, char* argv[])
14 {
15     // true values
16     const double f1 = 100;
17     const double f2 = 200;
18     const double f3 = 500;
19
20     // expectations
21     double g1_bar = 0.5 * (f1 + f2);
22     double g2_bar = 0.5 * (f1 + f3);
23     double g3_bar = 0.5 * (f2 + f3);
24
25     int noise_realizations = 100000;
26
27     double f1_mean  = 0;
28     double f2_mean  = 0;
29     double f3_mean  = 0;
30
31     double var1     = 0;
32     double var2     = 0;
33     double var3     = 0;
34
35     double cov12=0, cov21=0;
36     double cov13=0, cov31=0;
37     double cov23=0, cov32=0;
38
39     // init rnd
40     int seed = -123456;
41     ran1(&seed);
42
43     double *history = new double[noise_realizations*3];
44
45     int iterations = 1000;
46
47     for(int nr = 0; nr< noise_realizations; nr++)
48     {
49         if(nr%1000==0) printf("%d\n",nr);
50
51 #ifdef GAUSS
52         double g1 = g1_bar + sqrt(g1_bar) * gasdev(&seed);
53         double g2 = g2_bar + sqrt(g2_bar) * gasdev(&seed);
54         double g3 = g3_bar + sqrt(g3_bar) * gasdev(&seed);
55 #else
56         double g1 = poidev(g1_bar,&seed);
57         double g2 = poidev(g2_bar,&seed);
58         double g3 = poidev(g3_bar,&seed);
59 #endif
60
61         double f1_hat = 1;
62         double f2_hat = 1;
63         double f3_hat = 1;
64
65         for(int it=0;it<iterations;it++)
66         {
```

```
67              // projection
68              double expectation_g1   = 0.5*(f1_hat+f2_hat);
69              double expectation_g2   = 0.5*(f1_hat+f3_hat);
70              double expectation_g3   = 0.5*(f2_hat+f3_hat);
71              // invert expectations to avoid division by zero
72              if(expectation_g1>0) expectation_g1 = 1/expectation_g1;
73              else expectation_g1=0;
74              if(expectation_g2>0) expectation_g2 = 1/expectation_g2;
75              else expectation_g2=0;
76              if(expectation_g3>0) expectation_g3 = 1/expectation_g3;
77              else expectation_g3=0;
78              //backprojection
79              double update_g1        =   0.5*(g1*expectation_g1 + g2*expectation_g2);
80              double update_g2        =   0.5*(g1*expectation_g1 + g3*expectation_g3);
81              double update_g3        =   0.5*(g2*expectation_g2 + g3*expectation_g3);
82
83              f1_hat *= update_g1;
84              f2_hat *= update_g2;
85              f3_hat *= update_g3;
86          }
87
88          history[nr*3+0] = f1_hat;
89          history[nr*3+1] = f2_hat;
90          history[nr*3+2] = f3_hat;
91
92          f1_mean += f1_hat;
93          f2_mean += f2_hat;
94          f3_mean += f3_hat;
95      }
96
97      f1_mean /= (double)noise_realizations;
98      f2_mean /= (double)noise_realizations;
99      f3_mean /= (double)noise_realizations;
100
101     // calculate covariances
102     for(int nr = 0; nr< noise_realizations; nr++)
103     {
104         double f1_hat = history[3*nr+0];
105         double f2_hat = history[3*nr+1];
106         double f3_hat = history[3*nr+2];
107
108         var1 += (f1_hat - f1_mean) * (f1_hat - f1_mean);
109         var2 += (f2_hat - f2_mean) * (f2_hat - f2_mean);
110         var3 += (f3_hat - f3_mean) * (f3_hat - f3_mean);
111         cov12+= (f1_hat - f1_mean) * (f2_hat - f2_mean);
112         cov13+= (f1_hat - f1_mean) * (f3_hat - f3_mean);
113         cov23+= (f3_hat - f3_mean) * (f2_hat - f2_mean);
114     }
115
116     var1    /= (double)(noise_realizations-1);
117     var2    /= (double)(noise_realizations-1);
118     var3    /= (double)(noise_realizations-1);
119     cov12   /= (double)(noise_realizations-1);
120     cov13   /= (double)(noise_realizations-1);
121     cov23   /= (double)(noise_realizations-1);
122
123     return 0;
124 }
125
126
```

C:\Users\asitek\Desktop\Course\Covariance\Covariance.cpp 1

```
1  // Copyright (C) A. Sitek
2  // ML SINS Covariance
3  // Code 3
4  #include <stdio.h>
5  #include <math.h>
6  #include <memory.h>
7  #include "..\Common\NumRecipes.h"
8
9  using namespace num_recipesAS;
10 // uncomment below for Gaussian statistics - otherwise Poisson
11 //#define GAUSS
12
13 int main(int argc, char* argv[])
14 {
15     // for normal distribution integral under the curve: mean +- x * std
16     // 95.4% confidence interval for 2 standard deviations
17     double std1 = 0.682689492137;
18     double std2 = 0.954499736104;
19     double std3 = 0.997300203937;
20     double std4 = 0.999936657516;
21     double std5 = 0.999999426697;
22     double std6 = 0.999999998027;
23
24     // true values
25     const double f1 = 1;
26     const double f2 = 2;
27     const double f3 = 5;
28     // expectations
29     double g1_bar = 0.5 * (f1 + f2);
30     double g2_bar = 0.5 * (f1 + f3);
31     double g3_bar = 0.5 * (f2 + f3);
32
33     // expected variance of the ML estimation same for all and:
34     double expected_variance = g1_bar + g2_bar + g3_bar;
35     double sqrt_expected_variance = sqrt(expected_variance);
36
37     // estimation population variance of the estimates and check with prediction
38     int noise_realizations = 10000000;
39
40     double f1_mean  = 0;
41     double f2_mean  = 0;
42     double f3_mean  = 0;
43
44     double var1     = 0;
45     double var2     = 0;
46     double var3     = 0;
47
48     double cov12=0, cov21=0;
49     double cov13=0, cov31=0;
50     double cov23=0, cov32=0;
51
52     // init rnd
53     int seed = -123456;
54     ran1(&seed);
55
56     int ini_run_nr = 100000;
57     double *history = new double[ini_run_nr*6];
58
59     // histogram
60     const int histogram_size = 100;
61     double histogram_range[6][2];
62     int  *histograms[6];
63     for(int i=0;i<6;i++)
64     {
65         histograms[i] = new int[histogram_size];
66         memset(histograms[i],0,sizeof(int)*histogram_size);
```

```
67      }
68      // number of time true mean is within estimate+- sqrt(cov)
69      // This should be constant by frequentist statistics
70      // assuming true variace is known
71      double suc1_true_var=0;
72      double suc2_true_var=0;
73      double suc3_true_var=0;
74      // assming sample variance is used
75      double suc1_sample_var=0;
76      double suc2_sample_var=0;
77      double suc3_sample_var=0;
78
79      for(int nr = 0; nr< noise_realizations; nr++)
80      {
81          if(nr%10000==0) printf("%d\n",nr);
82
83  #ifdef GAUSS
84          double g1 = g1_bar + sqrt(g1_bar) * gasdev(&seed);
85          double g2 = g2_bar + sqrt(g2_bar) * gasdev(&seed);
86          double g3 = g3_bar + sqrt(g3_bar) * gasdev(&seed);
87  #else
88          double g1 = poidev(g1_bar,&seed);
89          double g2 = poidev(g2_bar,&seed);
90          double g3 = poidev(g3_bar,&seed);
91  #endif
92
93          // calculate estimates
94          double f1_hat = g1 + g2 - g3;
95          double f2_hat = g1 + g3 - g2;
96          double f3_hat = g2 + g3 - g1;
97
98          // sample variance
99          double f1_var = g1 + g2 + g3;
100         double f2_var = g1 + g3 + g2;
101         double f3_var = g2 + g3 + g1;
102
103         double sqrt_f1_var = sqrt(f1_var);
104         double sqrt_f2_var = sqrt(f2_var);
105         double sqrt_f3_var = sqrt(f3_var);
106
107         // assming the estimation is unbiased use true mean for covariace calculation
108         var1 += (f1_hat-f1)*(f1_hat-f1);
109         var2 += (f2_hat-f2)*(f2_hat-f2);
110         var3 += (f3_hat-f3)*(f3_hat-f3);
111
112         cov12 += (f1_hat-f1)*(f2_hat-f2);
113         cov13 += (f1_hat-f1)*(f3_hat-f3);
114         cov23 += (f2_hat-f2)*(f3_hat-f3);
115         // same for symmetrix terms
116
117         // see if this really is unbiased
118         f1_mean += f1_hat;
119         f2_mean += f2_hat;
120         f3_mean += f3_hat;
121
122         // is true value with the known std
123         if(f1_hat - sqrt_expected_variance * 1.96 <= f1 && f1_hat + sqrt_expected_variance* 1.96  >= ↙
    f1)
124         {
125             suc1_true_var ++;
126         }
127         if(f2_hat - sqrt_expected_variance* 1.96  <= f2 && f2_hat + sqrt_expected_variance* 1.96  >= ↙
    f2)
128         {
129             suc2_true_var ++;
130         }
```

```
131         if(f3_hat - sqrt_expected_variance* 1.96  <= f3 && f3_hat + sqrt_expected_variance* 1.96  >= f3)
132         {
133             suc3_true_var ++;
134         }
135
136         // is true value with the known std
137         if(f1_hat - sqrt_f1_var* 1.96  <= f1 && f1_hat + sqrt_f1_var* 1.96  >= f1)
138         {
139             suc1_sample_var ++;
140         }
141         if(f2_hat - sqrt_f2_var* 1.96  <= f2 && f2_hat + sqrt_f2_var* 1.96  >= f2)
142         {
143             suc2_sample_var ++;
144         }
145         if(f3_hat - sqrt_f2_var* 1.96  <= f3 && f3_hat + sqrt_f2_var* 1.96  >= f3)
146         {
147             suc3_sample_var ++;
148         }
149     }
150
151     suc1_true_var /=(double)noise_realizations;
152     double err_suc1_true_var = sqrt(suc1_true_var*(1-suc1_true_var))/sqrt((double)noise_realizations);
153     suc2_true_var/=(double)noise_realizations;
154     double err_suc2_true_var = sqrt(suc2_true_var*(1-suc2_true_var))/sqrt((double)noise_realizations);
155     suc3_true_var/=(double)noise_realizations;
156     double err_suc3_true_var = sqrt(suc3_true_var*(1-suc3_true_var))/sqrt((double)noise_realizations);
157
158     suc1_sample_var/=(double)noise_realizations;
159     double err_suc1_sample_var = sqrt(suc1_sample_var*(1-suc1_sample_var))/sqrt((double)noise_realizations);
160     suc2_sample_var/=(double)noise_realizations;
161     double err_suc2_sample_var = sqrt(suc2_sample_var*(1-suc2_sample_var))/sqrt((double)noise_realizations);
162     suc3_sample_var/=(double)noise_realizations;
163     double err_suc3_sample_var = sqrt(suc3_sample_var*(1-suc3_sample_var))/sqrt((double)noise_realizations);
164
165     var1 /= (double)noise_realizations;
166     var2 /= (double)noise_realizations;
167     var3 /= (double)noise_realizations;
168
169     cov12 /= (double)noise_realizations;
170     cov13 /= (double)noise_realizations;
171     cov23 /= (double)noise_realizations;
172
173     f1_mean /= (double)noise_realizations;
174     f2_mean /= (double)noise_realizations;
175     f3_mean /= (double)noise_realizations;
176
177 #ifdef GAUSS
178     printf("For Gauss statistics:\n");
179 #else
180     printf("For Poisson statistics\n");
181 #endif
182
183     printf("True means: %20.5f %20.5f %20.5f\n",f1,f2,f3);
184     printf("Esti means: %20.5f %20.5f %20.5f\n",f1_mean,f2_mean,f3_mean);
185     printf("Vari popul: %20.5f %20.5f %20.5f\n",var1,var2,var3);
186     printf("Cova popul: %20.5f %20.5f %20.5f\n",cov12,cov13,cov23);
187
188     printf("\nConfidence set theo: %20e\n",.95);
189     printf("Conf. set true   f1: %20e+-%20e\n",suc1_true_var, err_suc1_true_var);
```

```
190     printf("Conf. set true   f2: %20e+-%20e\n",suc2_true_var, err_suc2_true_var);
191     printf("Conf. set true   f3: %20e+-%20e\n",suc3_true_var, err_suc3_true_var);
192
193     printf("\nConfidence set samp: %20e\n",.95);
194     printf("Conf. set true   f1: %20e+-%20e\n",suc1_sample_var, err_suc1_sample_var);
195     printf("Conf. set true   f2: %20e+-%20e\n",suc2_sample_var, err_suc2_sample_var);
196     printf("Conf. set true   f3: %20e+-%20e\n",suc3_sample_var, err_suc3_sample_var);
197
198     return 0;
199 }
200
201
```

C:\Users\asitek\Desktop\Course\MCMCIntegration\MCMCIntegration.cpp 1

```
// Copyright (C) A. Sitek
// Markov Chain Metropolis Sapling Calulation of integral of sampling_func() over 2D xy [0:1]x[0:1].
// Sampling proportional to sampling_func(), and function evaluated 1/func()
// Since the Metropolis sampling is independet of normalization this procedure gives
// inverse of the integral
// I know ... it is cool
// Code 4
#include <stdio.h>
#include <math.h>
#include <memory.h>
#include "..\Common\NumRecipes.h"

using namespace num_recipesAS;

double sampling_func(const double x, const double y)
{
    return x*y; // change to whatever you wish to integrate over the domain
}
double integration_func(const double x, const double y)
{
    // sampling_func(x, y) cannot return 0 - no check
    return 1/sampling_func(x, y);
};
void doMCstep(double & x, double & y, double & prev_v, int & seed)
{
    const double step_size = 0.1;

    // update x,y
    double new_x = x + ( ran1(&seed) - 0.5 ) * step_size;
    // wrap it
    if(new_x<0)
    {
        new_x+=1.;
    }
    if(new_x>1)
    {
        new_x-=1.;
    }

    double new_y = y + ( ran1(&seed) - 0.5 ) * step_size;
    // wrap it
    if(new_y<0)
    {
        new_y+=1.;
    }
    if(new_y>1)
    {
        new_y-=1.;
    }

    double new_v = sampling_func(new_x,new_y);

    // reject if new_v less than prev_v and is unlucky
    if(new_v<prev_v)
    {
        if(ran1(&seed)>new_v/prev_v) // unlucky // chain do not advance
        {
            return;
        }
    }
    // advance the chain
    prev_v = new_v;
    x = new_x;
    y = new_y;
}
int main(int argc, char* argv[])
```

```
67 {
68     int seed = -123456;
69     // seed RNG
70     ran1(&seed);
71 
72     // start at 0,0
73     double x = 0;
74     double y = 0;
75     double v = sampling_func(x,y);
76 
77     const int buringin_steps = 100000;
78     const int measurement_steps = 50000000;
79 
80     // memory for saving the result will be used for the standard error calculations
81     double *memory = new double[measurement_steps];
82 
83     for(int s=0;s<buringin_steps;s++)
84     {
85         doMCstep(x,y,v,seed);
86     }
87 
88     double estimate = 0.;
89     for(int s=0;s<measurement_steps;s++)
90     {
91          doMCstep(x,y,v,seed);
92          memory[s] = integration_func(x,y);
93          estimate += memory[s];
94     }
95 
96     estimate /= (double)measurement_steps;
97 
98     double standard_error = 0.;
99 
100    for(int s=0;s<measurement_steps;s++)
101    {
102 
103         double diff = estimate - memory[s];
104         standard_error += diff*diff;
105    }
106 
107    standard_error /= (double)(measurement_steps-1)*(double)measurement_steps;
108    standard_error = sqrt(standard_error);
109 
110    printf("Calculated integral is: %.5f +- %.5f\n",1./estimate, standard_error/estimate/estimate);
111 
112    return 0;
113 }
114 
115 
```

```
1  // Copyright (C) A. Sitek
2  // SINS Marginalization
3  // This code determines the posterior of f_1, f_2, and f_3 conditioned on the acquired data
4  // it saves it in a file
5  // the distributions are saved in histogram form. See code for details of the setup.
6  // Code 5
7  #include <stdio.h> // for printf
8  #include <math.h>
9  #include <memory.h>
10
11 #include "..\Common\NumRecipes.h"
12
13 using namespace num_recipesAS;
14 class Histogram
15 {
16 public:
17     Histogram(double bw, double sv, int bn)
18     {
19         bin_width = bw;
20         start_value = sv;
21         bin_number = bn;
22         end_value = sv + (double)bin_number * bin_width;
23         histogram = new int[bin_number];
24         memset(histogram,0,sizeof(int)*bin_number);
25     };
26     ~Histogram()
27     {
28         delete [] histogram;
29     };
30     void DumpHistogramToFile(char *filename)
31     {
32         FILE *wsk = fopen(filename,"w");
33         if(!wsk) return;
34         for(int i=0;i<bin_number;i++)
35         {
36             fprintf(wsk,"%f %d \n",start_value + bin_width * (0.5 + (double)i),histogram[i]);
37         }
38         fclose(wsk);
39     }
40     void Add(double v)
41     {
42         if(v>=start_value && v < end_value )
43         {
44             int bin = (int)((v-start_value)/bin_width);
45             histogram[bin]++;
46         }
47     }
48 private:
49     double  bin_width;
50     double  start_value;
51     double  end_value;
52     int     bin_number;
53     int*    histogram;
54 };
55
56 enum PriorType {NONNEGATIVE, UPPERLIMIT};
57
58 // this calculation does not include division by a constant term due to factorial of the number of ↙
       counts
59 // Since the data is constant in Bayesian analysis we do not care about a constant term
60 // remember that Metropolis sampling does not care about the scaling factor for sampling distribution
61 double CalculateLikelihood(const double g1, const double g2, const double g3, const double f1, const ↙
       double f2, const double f3)
62 {
63     double expectation_g1   = 0.5*(f1+f2);
64     double expectation_g2   = 0.5*(f1+f3);
```

```
65      double expectation_g3   = 0.5*(f2+f3);
66
67      // calculate log for numberical stability
68      double logL1 = expectation_g1>0 ? g1 * log(expectation_g1) : 0;
69      double logL2 = expectation_g2>0 ? g2 * log(expectation_g2) : 0;
70      double logL3 = expectation_g3>0 ? g3 * log(expectation_g3) : 0;
71
72      double return_value =  exp(logL1 + logL2 + logL3 - expectation_g1 - expectation_g2 -
        expectation_g3);
73
74      return return_value;
75  }
76  double CalculatePrior(const double f1, const double f2, const double f3, PriorType type, const double
         prior_parameter)
77  {
78      double return_value = 0.;
79      switch(type)
80      {
81      case NONNEGATIVE:
82          {
83              if(f1>=0 && f2>=0 && f3>=0) return_value =  1;
84              else return_value = 0;
85              break;
86          }
87      case UPPERLIMIT:
88          {
89              if(f1>=0 && f2>=0 && f3>=0 && f1<=prior_parameter && f2<=prior_parameter && f3 <=
        prior_parameter) return_value =  1;
90              else return_value = 0;
91              break;
92          }
93      }
94      return return_value;
95  }
96  double CalculatePosterior(const double g1, const double g2, const double g3, const double f1, const
        double f2, const double f3, PriorType type, const double prior_parameter)
97  {
98      return CalculatePrior(f1, f2, f3, type, prior_parameter) * CalculateLikelihood(g1, g2, g3, f1, f2
        , f3);
99  }
100 void doMCStep(double & f1, double & f2,  double& f3, const double g1, const double g2, const double
        g3, double & prev_v, int & seed, PriorType type, const double prior_parameter)
101 {
102     const double step_size = 10.;
103
104     // some convinient way to generate new state
105     double new_f1 = 0;
106     do
107     {
108         new_f1 = f1 + ( ran1(&seed) - 0.5 ) * step_size;
109     } while(new_f1<0);
110
111     double new_f2 = 0;
112     do
113     {
114         new_f2 = f2 + ( ran1(&seed) - 0.5 ) * step_size;
115     } while(new_f2<0);
116
117     double new_f3 = 0;
118     do
119     {
120         new_f3 = f3 + ( ran1(&seed) - 0.5 ) * step_size;
121     } while(new_f3<0);
122
123
124     double new_v = CalculatePosterior(g1,g2,g3,new_f1,new_f2,new_f3, type, prior_parameter);
```

```
125
126     // reject if new_v less than prev_v and is unlucky
127     if(new_v<prev_v)
128     {
129         if(ran1(&seed)>new_v/prev_v) // unlucky // chain do not advance
130         {
131             return;
132         }
133     }
134     // advance the chain
135     prev_v = new_v;
136     f1 = new_f1;
137     f2 = new_f2;
138     f3 = new_f3;
139 }
140 int main(int argc, char* argv[])
141 {
142     int seed = -123456;
143     // seed RNG
144     ran1(&seed);
145
146     const double g1 = 10;
147     const double g2 = 30;
148     const double g3 = 50;
149
150     PriorType prior_type = NONNEGATIVE; // change to UPPERBOUND if prior with upper limit is desired
151     double  prior_parameter = 50;
152
153     double f1_current = 30.;
154     double f2_current = 30.;
155     double f3_current = 30.;
156     double v = CalculatePosterior(g1,g2,g3,f1_current,f2_current,f3_current, prior_type,
        prior_parameter);
157
158     const int burningin_steps = 100000;
159     const int measurement_steps = 50000000;
160
161     for(int i=0;i<burningin_steps;i++)
162     {
163         doMCStep(f1_current, f2_current, f3_current, g1, g2, g3, v, seed, prior_type,
        prior_parameter);
164     }
165     // We will determine the distributions of f_1 f_2 and f_3
166     Histogram histogramf1(0.1, 0, 1000);
167     Histogram histogramf2(0.1, 0, 1000);
168     Histogram histogramf3(0.1, 0, 1000);
169
170     for(int s=0;s<measurement_steps;s++)
171     {
172         doMCStep(f1_current, f2_current, f3_current, g1, g2, g3, v, seed, prior_type, prior_parameter
        );
173         // Note that all steps are added to the histogram altough they are highly correlated.
174         // It is assumed that in a long run these correlations are averaged out.
175         histogramf1.Add(f1_current);
176         histogramf2.Add(f2_current);
177         histogramf3.Add(f3_current);
178     }
179
180     histogramf1.DumpHistogramToFile("f1.txt");
181     histogramf2.DumpHistogramToFile("f2.txt");
182     histogramf3.DumpHistogramToFile("f3.txt");
183
184     return 0;
185 }
```

C:\Users\asitek\Desktop\Course\MMSE_1\MMSE_1.cpp 1

```
// Copyright (C) A. Sitek
// SINS MMSE using Metropolis
// Code 6
#include <stdio.h> // for printf
#include <math.h>
#include <memory.h>

#include "..\Common\NumRecipes.h"

using namespace num_recipesAS;

enum PriorType {NONNEGATIVE, UPPERBOUND};

// this calculation does not include division by a constant term due to factorial of the number of counts
// Since the data is constant in Bayesian analysis we do not care about a constant term
// remember that Metropolis sampling does not care about the scaling factor for sampling distribution
double CalculateLikelihood(const double g1, const double g2, const double g3, const double f1, const double f2, const double f3)
{
    double expectation_g1   = 0.5*(f1+f2);
    double expectation_g2   = 0.5*(f1+f3);
    double expectation_g3   = 0.5*(f2+f3);

    // calculate log end do exponent later for numerical stability
    double logL1 = expectation_g1>0 ? g1 * log(expectation_g1) : 0;
    double logL2 = expectation_g2>0 ? g2 * log(expectation_g2) : 0;
    double logL3 = expectation_g3>0 ? g3 * log(expectation_g3) : 0;

    double return_value =  exp(logL1 + logL2 + logL3 - expectation_g1 - expectation_g2 - expectation_g3);

    return return_value;
}
double CalculatePrior(const double f1, const double f2, const double f3, PriorType type, const double prior_parameter)
{
    double return_value = 0.;

    switch(type)
    {
    case NONNEGATIVE:
        {
            if(f1>=0 && f2>=0 && f3>=0) return_value =  1;
            else return_value = 0;
            break;
        }
    case UPPERBOUND:
        {
            if(f1>=0 && f2>=0 && f3>=0 && f1<=prior_parameter && f2<=prior_parameter && f3 <= prior_parameter) return_value =  1;
            else return_value = 0;
            break;
        }
    }

    return return_value;
}

double CalculatePosterior(const double g1, const double g2, const double g3, const double f1, const double f2, const double f3, PriorType type, const double prior_parameter)
{
    return CalculatePrior(f1, f2, f3, type, prior_parameter) * CalculateLikelihood(g1, g2, g3, f1, f2, f3);
}

```

```
60 void doMCStep(double & f1, double & f2,  double& f3, const double g1, const double g2, const double
       g3, double & prev_v, int & seed, PriorType type, const double prior_parameter)
61 {
62     const double step_size = 10.;
63 
64     // some convinient way to generate new state
65     double new_f1 = 0;
66     do
67     {
68         new_f1 = f1 + ( ran1(&seed) - 0.5 ) * step_size;
69     } while(new_f1<0);
70 
71     double new_f2 = 0;
72     do
73     {
74         new_f2 = f2 + ( ran1(&seed) - 0.5 ) * step_size;
75     } while(new_f2<0);
76 
77     double new_f3 = 0;
78     do
79     {
80         new_f3 = f3 + ( ran1(&seed) - 0.5 ) * step_size;
81     } while(new_f3<0);
82 
83     double new_v = CalculatePosterior(g1,g2,g3,new_f1,new_f2,new_f3, type, prior_parameter);
84 
85     // reject if new_v less than prev_v and is unlucky
86     if(new_v<prev_v)
87     {
88         if(ran1(&seed)>new_v/prev_v) // unlucky // chain do not advance
89         {
90             return;
91         }
92     }
93     // advance the chain
94     prev_v = new_v;
95     f1 = new_f1;
96     f2 = new_f2;
97     f3 = new_f3;
98 }
99 
100 int main(int argc, char* argv[])
101 {
102     int seed = -123456;
103     // seed RNG
104     ran1(&seed);
105 
106     const double g1 = 10;
107     const double g2 = 30;
108     const double g3 = 50;
109 
110     PriorType prior_type = NONNEGATIVE; // change to UPPERBOUND
111     double  prior_parameter = 50;
112 
113     double f1_current = 30.;
114     double f2_current = 30.;
115     double f3_current = 30.;
116     double v = CalculatePosterior(g1,g2,g3,f1_current,f2_current,f3_current, prior_type,
        prior_parameter);
117 
118     const int burningin_steps = 100000;
119     const int measurement_steps = 10000000;
120 
121     for(int i=0;i<burningin_steps;i++)
122     {
123         doMCStep(f1_current, f2_current, f3_current, g1, g2, g3, v, seed, prior_type,
```

```
    prior_parameter);
124     }
125
126     double f1_mmse_estimte = 0.;
127     double f2_mmse_estimte = 0.;
128     double f3_mmse_estimte = 0.;
129
130     for(int s=0;s<measurement_steps;s++)
131     {
132         doMCStep(f1_current, f2_current, f3_current, g1, g2, g3, v, seed, prior_type, prior_parameter↙
     );
133         f1_mmse_estimte += f1_current;
134         f2_mmse_estimte += f2_current;
135         f3_mmse_estimte += f3_current;
136     }
137
138     f1_mmse_estimte /= (double)measurement_steps;
139     f2_mmse_estimte /= (double)measurement_steps;
140     f3_mmse_estimte /= (double)measurement_steps;
141
142     printf("Calculated mmse estimate  is: %.5f %.5f %.5f\n",f1_mmse_estimte, f2_mmse_estimte,       ↙
    f3_mmse_estimte);
143
144     return 0;
145 }
146
147
```

C:\Users\asitek\Desktop\Course\OriginEnsemble\OriginEnsemble.cpp 1

```
1  // Copyright (C) A. Sitek
2  // SINS Marginalization
3  // This code does the orgin ensemble reconstruction for SINS with far prior
4  // It also does posterior marginalization
5  // Code 7
6  #include <stdio.h> // for printf
7  #include <math.h>
8  #include <memory.h>
9
10 #include "..\Common\NumRecipes.h"
11
12 using namespace num_recipesAS;
13
14 class DiscreteHistogram
15 {
16 public:
17     DiscreteHistogram(int sv, int s)
18     {
19         start_value = sv;
20         end_value = sv + s;
21         bin_number = s;
22         histogram = new int[bin_number];
23         memset(histogram,0,sizeof(int)*bin_number);
24     };
25     ~DiscreteHistogram()
26     {
27         delete [] histogram;
28     };
29     void DumpHistogramToFile(char *filename)
30     {
31         FILE *wsk = fopen(filename,"w");
32         if(!wsk) return;
33         for(int i=0;i<bin_number;i++)
34         {
35             fprintf(wsk,"%d %d \n",start_value + i,histogram[i]);
36         }
37         fclose(wsk);
38     }
39     void Add(double v)
40     {
41         if(v>=start_value && v < end_value )
42         {
43             int bin = (v-start_value);
44             histogram[bin]++;
45         }
46     }
47 private:
48     int     start_value;
49     int     end_value;
50     int     bin_number;
51     int*    histogram;
52 };
53
54 void doMCStep(int * c1, int * c2, int * c3, const int g1, const int g2, const int g3, int & seed, int↙
       *origin_lookup)
55 {
56     // for convinient access define pointer to c's
57     int *ptrc[3] = {c1, c2, c3};
58
59     // first randomly choose and event
60     int an_event = (int)(ran1(&seed)*(double)(g1+g2+g3));
61     // determine in which voxel the origin is currently located
62     int bin_i = origin_lookup[an_event];
63     // determine projection bin in which it was detected
64     int pr = origin_lookup[g1+g2+g3+an_event];
65     // randomly select new voxel (a candidate for a new location of the origin
```

```
66      int new_bin = -1;
67      if(pr==0)
68      {
69          if(ran1(&seed)<0.5)
70          {
71              new_bin = 0;
72          }
73          else
74          {
75              new_bin = 1;
76          }
77      }
78      else if(pr==1)
79      {
80          if(ran1(&seed)<0.5)
81          {
82              new_bin = 0;
83          }
84          else
85          {
86              new_bin = 2;
87          }
88      }
89      else
90      {
91          if(ran1(&seed)<0.5)
92          {
93              new_bin = 1;
94          }
95          else
96          {
97              new_bin = 2;
98          }
99      }
100     // same bin - always accept
101     if(new_bin == bin_i) return;
102
103     // check the total number of events
104     int c_i = *ptrc[bin_i];
105     int c_i_prime =  *ptrc[new_bin];
106
107     // Check for rejection
108     if(c_i_prime+1<c_i)
109     {
110         if(ran1(&seed)>(double)(c_i_prime+1)/(double)c_i)
111         {
112             return;
113         }
114     }
115
116     // move event orgin to new pixel
117     origin_lookup[an_event] = new_bin;
118     (*ptrc[bin_i])--;
119     (*ptrc[new_bin])++;
120 }
121
122 int main(int argc, char* argv[])
123 {
124     int seed = -123456;
125     // seed RNG
126     ran1(&seed);
127
128     const int g1 = 10;
129     const int g2 = 30;
130     const int g3 = 50;
131
```

```
132     // for each detected event we will mantain a lookup that will remember in which voxel the orgin ↙
        is located and in which projection element was detected
133     int *origin_lookup = new int[(g1+g2+g3)*2];
134
135     // total number of emissions per voxel
136     int         c1=0;
137     int         c2=0;
138     int         c3=0;
139
140     // initiate by guessing where detected events took place (in which voxel is the origin)
141     // events in g1 are either from 1 or 2 with 50/50 chance due to system matrix
142     for(int i=0;i<g1;i++)
143     {
144         if(ran1(&seed)<0.5)
145         {
146             c1++;
147             origin_lookup[i]=0;
148         }
149         else
150         {
151             c2++;
152             origin_lookup[i]=1;
153         }
154         //projection element 1 (C-notation 0)
155         origin_lookup[g1+g2+g3+i]=0;
156     }
157     for(int i=0;i<g2;i++)
158     {
159         if(ran1(&seed)<0.5)
160         {
161             c1++;
162             origin_lookup[g1+i]=0;
163         }
164         else
165         {
166             c3++;
167             origin_lookup[g1+i]=2;
168         }
169         origin_lookup[g1+g2+g3+g1+i]=1;
170     }
171     for(int i=0;i<g3;i++)
172     {
173         if(ran1(&seed)<0.5)
174         {
175             c2++;
176             origin_lookup[g1+g2+i]=1;
177         }
178         else
179         {
180             c3++;
181             origin_lookup[g1+g2+i]=2;
182         }
183         origin_lookup[g1+g2+g3+g1+g2+i]=2;
184     }
185     const int burningin_steps = 100000;
186     const int measurement_steps = 50000000;
187
188     for(int i=0;i<burningin_steps;i++)
189     {
190         doMCStep(&c1,&c2,&c3,g1,g2,g3,seed,origin_lookup);
191     }
192
193     DiscreteHistogram histogramc1(0, 100);
194     DiscreteHistogram histogramc2(0, 100);
195     DiscreteHistogram histogramc3(0, 100);
196
```

```
197     double mmse_estimate_c1 = 0;
198     double mmse_estimate_c2 = 0;
199     double mmse_estimate_c3 = 0;
200
201     for(int i=0;i<measurement_steps;i++)
202     {
203         doMCStep(&c1,&c2,&c3,g1,g2,g3,seed,origin_lookup);
204         mmse_estimate_c1 += (double)c1;
205         mmse_estimate_c2 += (double)c2;
206         mmse_estimate_c3 += (double)c3;
207         histogramc1.Add(c1);
208         histogramc2.Add(c2);
209         histogramc3.Add(c3);
210     }
211
212     mmse_estimate_c1 /= (double)measurement_steps;
213     mmse_estimate_c2 /= (double)measurement_steps;
214     mmse_estimate_c3 /= (double)measurement_steps;
215
216     histogramc1.DumpHistogramToFile("c1.txt");
217     histogramc2.DumpHistogramToFile("c2.txt");
218     histogramc3.DumpHistogramToFile("c3.txt");
219
220     return 0;
221 }
222
223
224
225
```

C:\Users\asitek\Desktop\Course\Common\NumRecipes.h 1

```
#include <stdio.h>
#include <math.h>
#include <memory.h>

namespace num_recipesAS
{
    #define IA 16807
    #define IM 2147483647
    #define IQ 127773
    #define IR 2836
    #define NTAB 32
    #define EPS (1.2E-07)
    #define MAX(a,b) (a>b)?a:b
    #define MIN(a,b) (a<b)?a:b

    double ran1(int* idum)
    {
           int j,k;
           static int iv[NTAB],iy=0;

           static double NDIV = 1.0/(1.0+(IM-1.0)/NTAB);
           static double RNMX = (1.0-EPS);
           static double AM = (1.0/IM);

           if ((*idum <= 0) || (iy == 0)) {
                   *idum = MAX(-*idum,*idum);
                   for(j=NTAB+7;j>=0;j--) {
                           k = *idum/IQ;
                           *idum = IA*(*idum-k*IQ)-IR*k;
                           if(*idum < 0) *idum += IM;
                           if(j < NTAB) iv[j] = *idum;
                   }
                   iy = iv[0];
           }
           k = *idum/IQ;
           *idum = IA*(*idum-k*IQ)-IR*k;
           if(*idum<0) *idum += IM;
           j = iy*NDIV;
           iy = iv[j];
           iv[j] = *idum;
           return MIN(AM*iy,RNMX);
    }
    #undef IA
    #undef IM
    #undef IQ
    #undef IR
    #undef NTAB
    #undef EPS
    #undef MAX
    #undef MIN


    double gasdev(int *idum)
    {

           static int iset=0;
           static double gset;
           double fac,rsq,v1,v2;

           if (*idum < 0) iset=0;
           if  (iset == 0) {
                   do {
                           v1=2.0*ran1(idum)-1.0;
                           v2=2.0*ran1(idum)-1.0;
                           rsq=v1*v1+v2*v2;
                   } while (rsq >= 1.0 || rsq == 0.0);
```

```
67                      fac=sqrt(-2.0*log(rsq)/rsq);
68                      gset=v1*fac;
69                      iset=1;
70                      return v2*fac;
71              } else {
72                      iset=0;
73                      return gset;
74              }
75      }
76
77      double gammln(double xx)
78      {
79              double x,tmp,ser;
80              static double cof[6]={76.18009173,-86.50532033,24.01409822,
81                      -1.231739516,0.120858003e-2,-0.536382e-5};
82              int j;
83
84              x=xx-1.0;
85              tmp=x+5.5;
86              tmp -= (x+0.5)*log(tmp);
87              ser=1.0;
88              for (j=0;j<=5;j++) {
89                      x += 1.0;
90                      ser += cof[j]/x;
91              }
92              return -tmp+log(2.50662827465*ser);
93      };
94      #define PI 3.141592654
95      double poidev(double xm, int *idum)
96      {
97              static double sq,alxm,g,oldm=(-1.0);
98              double em,t,y;
99
100
101             if (xm < 12.0) {
102                     if (xm != oldm) {
103                             oldm=xm;
104                             g=exp(-xm);
105                     }
106                     em = -1;
107                     t=1.0;
108                     do {
109                             em += 1.0;
110                             t *= ran1(idum);
111                     } while (t > g);
112             } else {
113                     if (xm != oldm) {
114                             oldm=xm;
115                             sq=sqrt(2.0*xm);
116                             alxm=log(xm);
117                             g=xm*alxm-gammln(xm+1.0);
118                     }
119                     do {
120                             do {
121                                     y=tan(PI*ran1(idum));
122                                     em=sq*y+xm;
123                             } while (em < 0.0);
124                             em=floor(em);
125                             t=0.9*(1.0+y*y)*exp(em*alxm-gammln(em+1.0)-g);
126                     } while (ran1(idum) > t);
127             }
128             return em;
129     }
130 };
```