aprendtech.com >> blog >> this post

If you have trouble viewing this, try the pdf of this post. You can download the code used to produce the figures in this post.

Image SNR with energy-selective detectors

This is the last post in my series discussing my paper, “Near optimal energy selective x-ray imaging system performance with simple detectors”[1]. In the last post I showed plots of the signal to noise ratio (SNR) of images with different types of energy-selective detectors. In this post, I show images illustrating these differences. These images were not included in the paper but they are based on its approach. The images are calculated from a random sample of the energy spectrum at each point in a projection image. These data are then used to make images with (a) the total energy, which are comparable to the detectors now used in commercial systems, (b) the total number of photons, (c) an N2Q detector, and (d) an optimal full spectrum by weighting the spectrum data before summing, as described in Tapiovaara and Wagner (TW)[2]. I use the theory developed in my paper, to make images from A-space data using data from the N2Q detector. In order to do this, I need an estimator that achieves the Cramèr-Rao lower bound (CRLB). For this I use the A-table estimator I introduced in my paper “Estimator for photon counting energy selective x-ray imaging with multibin pulse height analysis”[3] available for download here.

The Monte Carlo experiment

The experimental set up is shown in Fig. 1 and the details are described in the caption. Poisson distributed random data are computed with expected value equal to the transmitted spectrum at each pixel. The random data are stored in a matrix with number of rows equal to the number of pixels in the image and number of columns equal to the number of energies in the spectrum. The matrix is used to compute the data for the images. For example, the photon count image is the sum of each of the rows. This gives a column vector, that can be “re-shaped” to form the image. Similarly, the N2Q data can be generated by summing along the rows from the lowest energy to the threshold energy, which is the red, dashed, vertical line shown in the exit spectrum in the bottom panel, and from the threshold energy plus one to the highest energy. This gives the N2 data at each row. The Q data is the sum of the photon counts each multiplied by the energy for that point in the spectrum.

The simulated object was generated with my sim2d.m function, which is included with the code package. The relevant code from P36SNRimages.m is in Listing 1

Listing 1: Compute simulated object

```
1  %%% init the simulated object
2  thicknesses = [1 -.1 .1]; % gm/cm^2 -- will assume density is equal to 1
3  nx = 256; ny = 256;
4  zsiz = nx + 1i*ny; % dimensions of images
5  z1 = 1 + 1i;
6     % boxes coords in unit based coords
7  zcorners = nx*[
8       .1*z1, .9*z1; % background material region
9       .2*z1,.2*z1 + .2+.6i;
10      .2*z1,.2*z1 + .2+.6i;  % hole for feature
11      ];
12    % make object out of 3 planes: background,
13    % -feature from background slab,
14    % fill in feature material in background slab
15 tplanes = zeros(ny,nx,2); % one plane per background and feature
16    % combine background plane with hole for feature
17 for kt = 1:2
18      timg = sim2d(zsiz,'rect',[zcorners(kt,:),thicknesses(kt)]);
19      tplanes(:,:,1) =  tplanes(:,:,1) + timg;
20 end
21 % add in feature
22 tplanes(:,:,2) = sim2d(zsiz,'rect',[zcorners(3,:),thicknesses(3)]);
23    % reformat into columns (ny*nx,2)
24 ts_object = reshape(tplanes,ny*nx,2);
```

The object, shown in the middle part of Fig. 1 on the right, is generated as three planes that are summed pixel by pixel to form it. The three planes are the overall background material slab, a hole in the background for the feature and the feature alone. The code from lines 1-11 of Listing 1 defines the sim2D.m specifications for the image in each plane. See the "help" for sim2D.m for more details on the specification. The tplanes 3D array holds the background and feature thicknesses in each plane respectively. This is computed in lines 12-22. The last line, 24, reshapes tplanes so each row is the thickness of the feature and background material.

The random transmitted spectrum is generated using the code shown in Listing 2 below.

Listing 2: Generate random transmitted data

```
1  %%% make the incident spectrum and  transmitted spectrum
2  kV = 120;
3  [dum,dum,specdat] = XrayTubeSpectrumTasmip(kV,'number_spectrum','clipzeros');
4  specdat.mus = [xraymu(ztback,specdat.egys) xraymu(ztfeat,specdat.egys)];
5
6  %%% make the random transmitted data
7  transmission = exp(-ts_object*specdat.mus');
8     % the expected values of the transmitted spectrum
9  lambdas = bsxfun(@times,transmission,specdat.specnum_norm'); % siz = (ny*nx,nenergy)
10 N = 10^3; % total number of photons in trasnmitted spectrum
11 offset2counts = 1/numel(specdat.egys); % add small number so no zeros
12    % generate the random data
13 ns = poissrnd(N*lambdas) + offset2counts;
```

The spectrum and the attenuation coefficients of the background and feature material are computed in lines 1-4. The expected values of the transmitted spectrum are generated in lines 6-10. The random data are computed in line 13. The ns matrix has one row per pixel in the image and the columns are the values for each spectrum energy. Note that a small

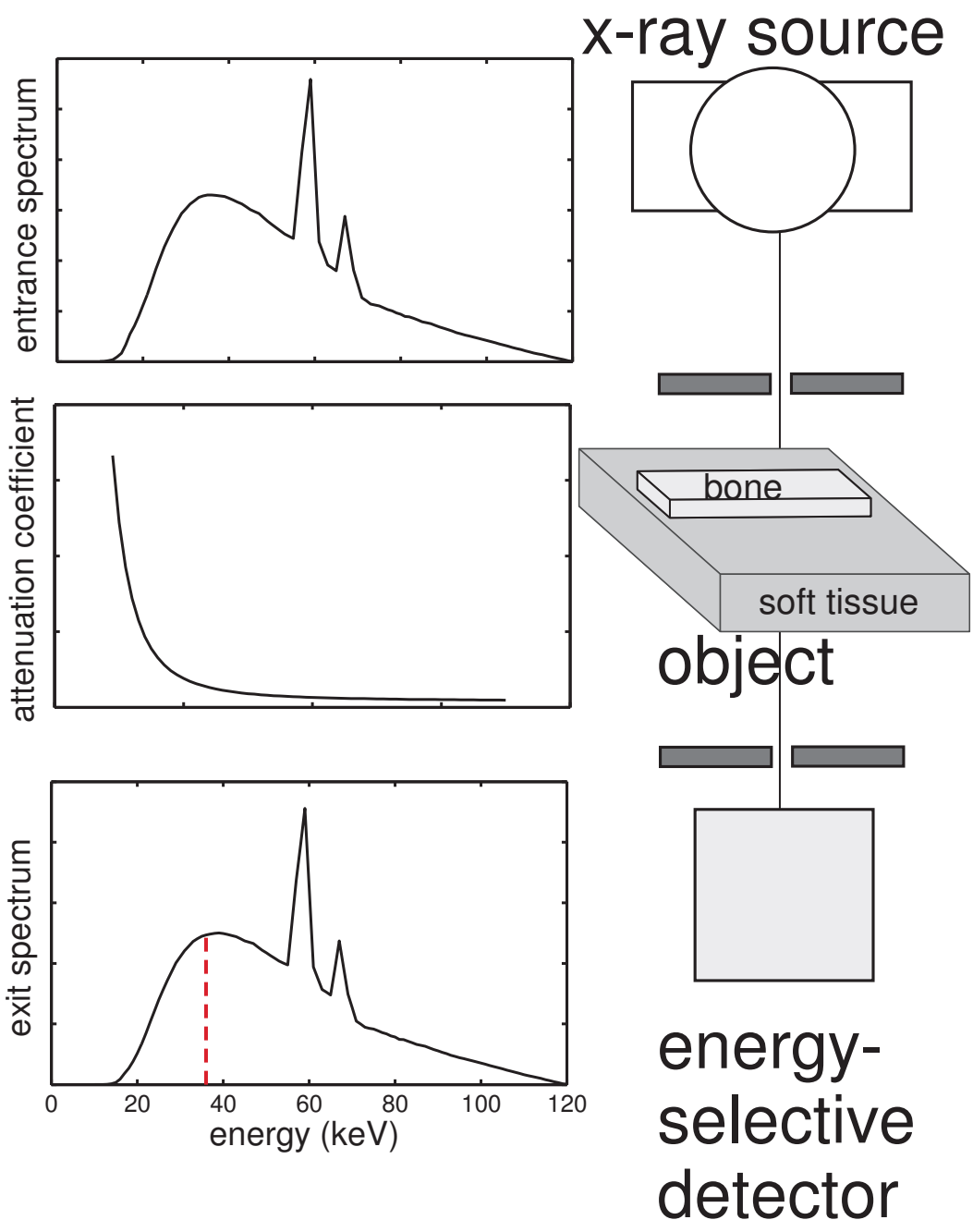

Figure 1: Block diagram of the experiment. An x-ray tube generates the entrance spectrum. The collimated beam goes through every pixel of the object with "good geometry" so scatter is negligible. The object consists of the background, a slab of tissue material, and the feature, a box of bone-material. The feature is embedded in the background so there is a hole in the tissue slab to fit the bone box. The background is 1 gm/cm^2 and the feature is 0.1 gm/cm^2. The transmitted spectrum is measured at each point in the image.

constant is added to the data so there are no zero values. This would lead to problems in the computation of the log data for the N2Q image.

Making the N, Q, and optimal, full-spectrum images

The N, Q, and optimal, full-spectrum TW images were computed directly from the random transmitted data generated with the code in Listing 2. The code to generate the images is in Listing 3. The $bsxfun$ function is a fast way to compute element-by-element products of a matrix and a vector (see the $Matlab$ help).

Listing 3: Compute N, Q, and TW images

```
%% N image
Ns = sum(ns,2);
imdatN = reshape(Ns,ny,nx);
snrN = SNRfeature(imdatN);

%% Q image
Qs = sum(bsxfun(@times,ns,specdat.egys'),2);
imdatQ = reshape(Qs,ny,nx);
snrQ = SNRfeature(imdatQ);

%% Make the TW optimal image by weighting the full spectrum data at each point in image
   % for a thin object, the optimal weighting function is proportional to
   % diff of the mus at each  energy
dmus = diff(specdat.mus,1,2);
dmus_norm = dmus/sum(dmus);
imTW = reshape(sum(bsxfun(@times,ns,dmus_norm'),2),ny,nx);
snrTW = SNRfeature(imTW);
```

Making the N2Q data

Computing the N2Q image is more complex but the result has advantages over the other images including the full-spectrum TW. It is a two step process: first, compute the A-vector data, then compute a low-noise image from the A-vectors. I will discuss the first step in this section and leave the computation of the image for the next section. The steps to compute the A-vector data are (each step refers to the lines of code in Listing 4):

1. Compute the N2Q data from the ns matrix and the threshold index (1-5)
2. Make the “air data” i.e. the expected value of the data with no object in the beam (6-9).
3. Compute $-\log\left({}^{N2Q}/_{N2Q_0}\right)$ (10-12)
4. Compute noise-free calibration data for a two element step wedge made of the background and feature material. Use of these materials is unrealistic but the results would be the same for more realistic materials such as aluminum and plastic and would make the code needlessly complex. (14-28)
5. Compute the covariance of the log data. This is required for the A-table algorithm. This can be done experimentally from the log N2Q image using the same approach. (30-31)

6. Solve for the A-vector image using my $AtableSolveEquations.m$ function. This is included with the code package. (33-38)

Listing 4: Compute the N2Q image

```
%% N2Q data -- will process using my A-table method
  % first compute object data
Ethreshold = 35; % keV
[dum,idxthreshold] = min(abs(specdat.egys - Ethreshold));
N2Qs = [sum(ns(:,1:idxthreshold),2) sum(ns(:,(idxthreshold+1):end),2) Qs];
   % 'air data' i.e. no object in beam. Assume has no noise
N2Qs0 = N*[sum(specdat.specnum_norm(1:idxthreshold)) ...
                   sum(specdat.specnum_norm((idxthreshold+1):end)) ...
                   sum(specdat.specnum_norm.*specdat.egys)];
  % add offset to air data so it matches the offset added to noisy data
N2Qs0(1:2) = N2Qs0(1:2) + offset2counts;
logN2Qs = bsxfun(@minus,log(N2Qs0),log(N2Qs)); % these are positive numbers

%% calibration data -- assume noise free
ncalib_points = 20;
t1s_calib = linspace(-.5,4*thicknesses(1),ncalib_points);
% allow negatives so fewer out of range points
t2s_calib = linspace(-.3,4*thicknesses(3),ncalib_points);
idxs = combinator(ncalib_points,2,'p','r'); % permutations taken 2 at a time with repitition
ts_calib = [t1s_calib(idxs(:,1))' t2s_calib(idxs(:,2))'];
transmission_calib = exp(-ts_calib*specdat.mus');
lambdas_calib = bsxfun(@times,transmission_calib,specdat.specnum_norm'); % siz = (ncalib steps,ne
ns_calib = N*lambdas_calib + offset2counts; % no noise for calibration data
Qs_calib = sum(bsxfun(@times,ns_calib,specdat.egys'),2);
N2Qs_calib = [sum(ns_calib(:,1:idxthreshold),2) sum(ns_calib(:,(idxthreshold+1):end),2) Qs_calib]
   % air data does not change so use results from previous cell
logN2Qs_calib = bsxfun(@minus,log(N2Qs0),log(N2Qs_calib)); % these are positive numbers
cdat.As = ts_calib;
cdat.logI = logN2Qs_calib;

%% compute covariance of log(N2Q) data from background region of image
covstats.RL = cov(logN2Qs(idxs4back,:));

%% solve for A vector image data
tdat.As = ts_object;
tdat.logI = logN2Qs;
% N2Qdat = PolyASolveEquations(cdat,tdat);
N2Qdat = AtableSolveEquations(cdat,tdat,'stats',covstats);
zAs_back = zcomp(N2Qdat.Astest(idxs4back,:));
```

The final steps are to compute a low noise image from the A-vector data. This is discussed in the next section.

Compute low-noise image from N2Q A-vector data

The output of the A-space method is a set of A-vectors for every pixel in the image or line in CT. We can make a scalar image comparable to the other images by doing a vector dot product of the A-vector at every pixel with a unit vector. This is called a generalized projection. By varying the angle of the unit vector, we can cancel tissues from images or make images comparable to a conventional projection image. This was first discussed in Section 4.6 of my dissertation and in more detail in Sections 2.2 and 2.3 of this paper available on my website.

A problem with forming images from A-vector data is that the noise is highly negatively correlated. As a result, the noise will depend on the angle of the unit vector. My my paper described an alternate approach, which solved this problem. The approach is to transform with data by using a whitening transform. My paper showed that with this approach, you can make images with the best SNR available for a given detector type.

Whitening transforms in general are described in one of my previous posts, which is available in Ch. 25 of my ebook. The whitening transform is computed with the code in Listing 5. Line 2 computes the A-vector covariance from the data in the background region. Line 3 computes the transform and line 4 applies it to the whole image. Line 5 converts the whitened image data to complex numbers to make it easy for the following computations

Listing 5: Whiten code

```
1 %% compute whiten xform -- see my blog posts ebook Eq. 25.23
2 [V,D] = eig(cov(N2Qdat.Astest(idxs4back,:)));
3 Pwhiten = V*D^(-.5);
4 Aswhiten = N2Qdat.Astest*Pwhiten;
5 zAswhiten = zcomp(Aswhiten);
```

Once we have the whitened data, we can compute images by taking a vector dot product with a unit vector at each pixel and choose the angle of the unit vector to maximize the SNR. This is done with the code in Listing 6. The angle for maximum SNR is chosen by computing multiple images for a large number of angles from 0 to 2π and selecting the one that gives the maximum value. Fig. 2 plots the results.

Listing 6: Computing generalized projection angle to maximize SNR

```
1  %% find projection angle for maximum SNR
2  nprojangles = 100;
3  projangles = linspace(0,2*pi,nprojangles+1); % will remove 1 so no wraparound
4  projangles = projangles(1:nprojangles);
5  snrs = zeros(nprojangles,1);
6
7  for kangle = 1:nprojangles
8     improj = zdot(zAswhiten,exp(1i*projangles(kangle)));
9     snrs(kangle) = SNRfeature(improj);
10 end
11 idxpeak = findpeaks(snrs);
12 assert(~isempty(idxpeak));
13 imN2Q = reshape(zdot(zAswhiten,exp(1i*projangles(idxpeak(1)))),ny,nx);
14 snrN2Q = SNRfeature(imN2Q);
```

Results

The images with different detectors are shown in Fig. 3. These were all computed from the same random data; the only difference is the processing to emulate the characteristics of each detector type. As expected from the results in the last post, the Q image has the lowest SNR, the N is somewhat better and the N2Q is close to the Tapiovaara-Wagner SNR. The N2Q and TW images have different visual appearance with the N2Q seeming to have higher contrast but the numerical value of the N2Q image SNR is close to the TW image.

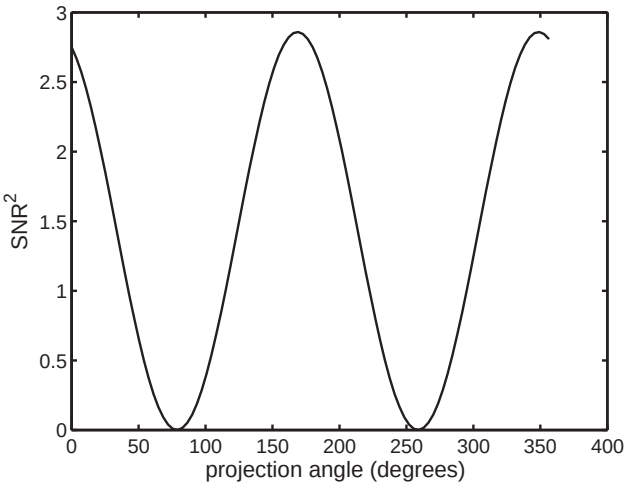

Figure 2: SNR vs. generalized projection angle.

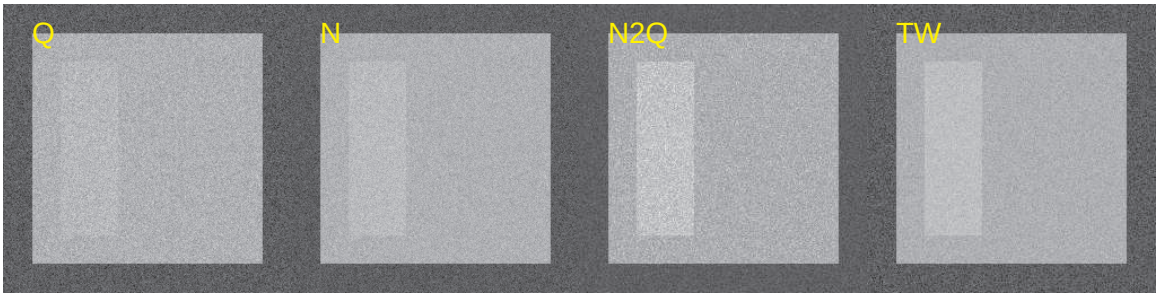

Figure 3: Images with Q, N, N2Q, and TW detectors.

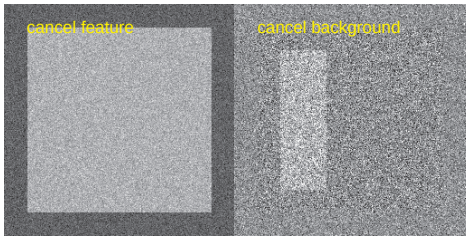

Figure 4: Feature and background material cancelled images. These were computed from the whitened data by selecting the generalized projection angle that results in zero SNR for each type of material. The result shows only the background or only the feature material in this two material case.

Conclusions

The N2Q detector with A-space processing has SNR close to the TW optimal. I showed in my paper that images with high SNR can also be computed from N2 (two-bin PHA) and NQ (simultaneous photon counts and integrated energy) detectors. The fact that these low energy-resolution detectors give such good performance is due to the simplicity of the underlying information, which can be completely summarized by the coefficients of two basis functions. This information was not used by Tapiovaara and Wagner or other people who have used their approach.

Notice that there are generalized projection angles in Fig. 2 with 0 SNR. These are the angles that cancel the feature material from the image. The images, computed from the whitened data, with zero SNR angles for the feature and background materials respectively, are shown in Fig. 4. With the A-space method, we not only get high SNR images but also have the additional A-space information that we can use for selective material imaging. This is not possible with the Tapiovaara-Wagner approach.

–Bob Alvarez

Last edited April 13, 2013

References

[1] R. E. Alvarez, “Near optimal energy selective x-ray imaging system performance with simple detectors,” *Med. Phys.* **37**, 822–841, (2010).

[2] M. J. Tapiovaara and R. Wagner, “SNR and DQE analysis of broad spectrum X-ray imaging,” *Phys. Med. Biol.* **30**, 519–529, (1985).

[3] R. E. Alvarez, “Estimator for Photon Counting Energy Selective X-ray Imaging with Multi-bin Pulse Height Analysis,” *Med. Phys.* **38**, 2324–2334, (2011).