NOISE POWER SPECTRUM CALCULATOR

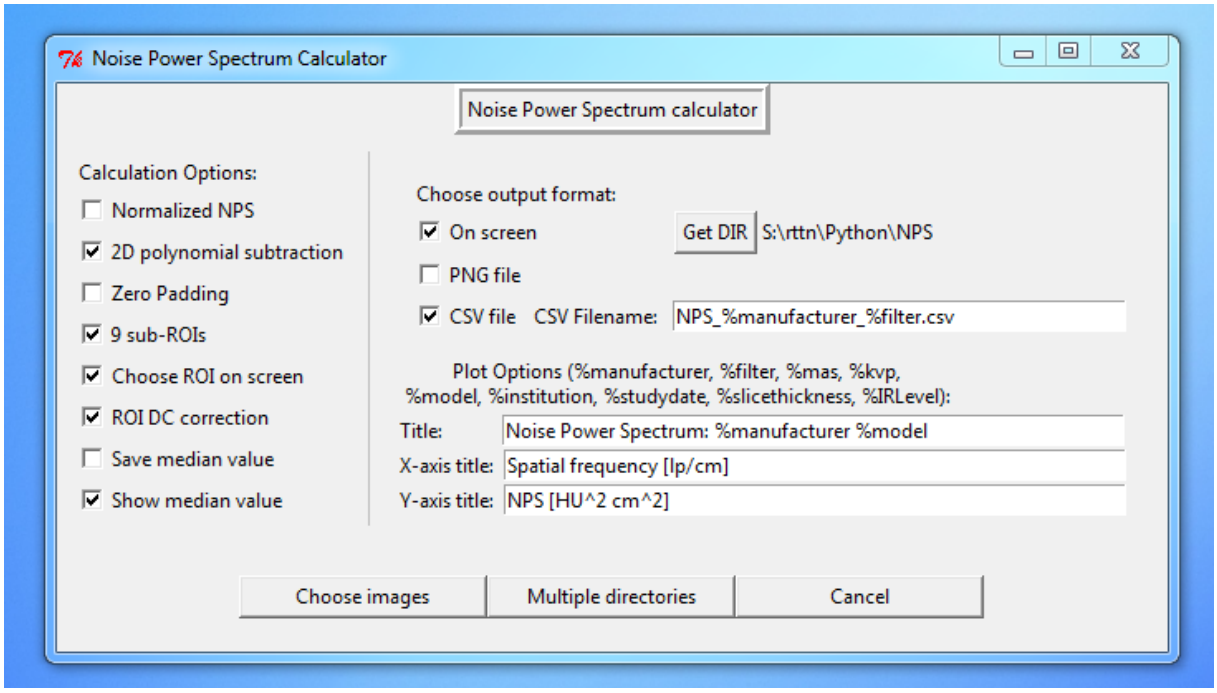
Noise Power Spectrum (NPS) er et mål som sier noe om støyteksturen i et bilde. Se Boedeker et al. Application of the NPS in MDCT, del 1 og 2 for mer informasjon om NPS.

Det finnes mange programmer som kan regne ut NPS og normalisert NPS (NNPS) for ett og ett bilde eller flere bilder i én serie (CTQA-CP, IQWorks osv.), f.eks. som del av et QA-program. Dette programmet sammenlikner bildeserier med hverandre, f.eks. om man ønsker å se hvordan NPS varierer over forskjellige scannere, rekonstruksjonsfiltre, rekonstruksjonsalgoritmer osv.

Man trenger støybilder uten struktur (f.eks. vann, PMMA eller subtraherte bilder) i DICOM-format. Det er to måter å lagre bildene på:

- I samme mappe
    - Hvor man lager én NPS-kurve, jo flere bilder jo bedre statistikk.
    - Bruk knappen **Choose images**
- I forskjellige mapper
    - Hvor man lager én NPS-kurve av alle bildene i hver mappe, og sammenlikner mappene mot hverandre.
    - Bruk knappen **Multiple directories**

Når man kjører programmet (gjennom `main.py`) får man flere valgmuligheter opp. Se Figur 1.



FIGUR 1: ÅPNINGSVINDUET

Når man har valgt alle innstillingene (se under for beskrivelse), velger man bilder enten med **Choose images** eller **Multiple directories**, hhv. etter om man ønsker å lage én NPS-kurve fra enkeltbilder eller sammenlikne alle bildene i én mappe mot alle bildene i en annen mappe (opp til 30 mapper).

Legg merke til at dersom man ønsker å velge flere mapper må man velge én og én mappe i dialogen som kommer fra (velg mappe -> OK, ny dialog, velg mappe -> OK), helt til man er fornøyd (og da må man trykke på **Avbryt**).

Når det er gjort starter beregningene, og om det ikke oppstår feil vil resultatene dannes, etter hvordan man har valgt å få dem opp (On screen, PNG file og/eller CSV file).

Hver kurve vil få farge etter en permutert liste: Med linje: Rød, grønn, svart, blå, lyseblå, gul. Med stiplet linje: Samme fargesekvens. Så kommer kortstiplet, prikk-stiplet og prikket kurve. Totalt 30 kombinasjoner, dvs. 30 mulige kurver i samme plot.

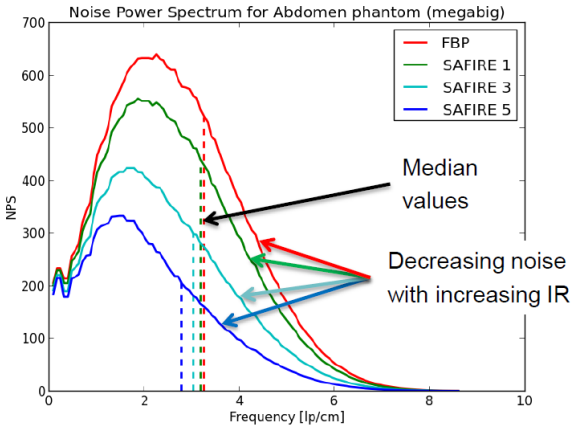
En beskrivelse av kurvene vil dukke opp i høyre hjørne. Den vil vise filter, samt egenskaper som er *ulike* blant kurvene: Modellnavn, IR-styrke og mAs.

BESKRIVELSE AV INNSTILLINGENE

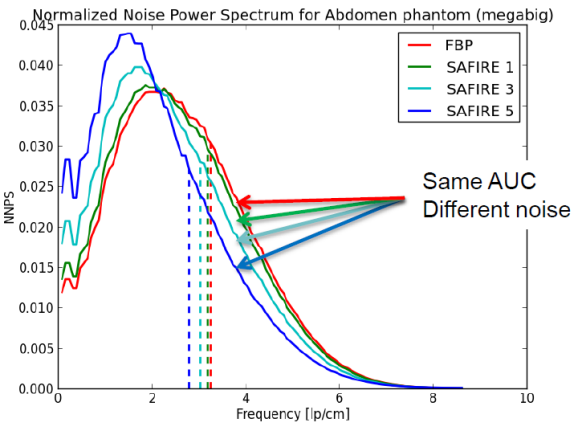
NORMALIZED NPS

Et nyttig verktøy dersom man vil sammenlikne systemer mot hverandre. I utgangspunktet (unormalisert NPS) vil arealet under NPS-kurven svare til standardavviket eller støymengden i bildet. To forskjellige filtre med forskjellig støynivå vil derfor representeres som to NPS-kurver med forskjellig areal og høyde.

Dersom man aktiverer **Normalized NPS** vil kurven normaliseres på arealet under kurven. Siden enheten til NPS er $\mathrm{HU}^2\mathrm{cm}^2$, og enheten til frekvens er cm^{-1}, vil arealet ha enheten $\mathrm{HU}^2\ \mathrm{cm}$. Det betyr at NNPS normalisert på denne måten har arealet cm.



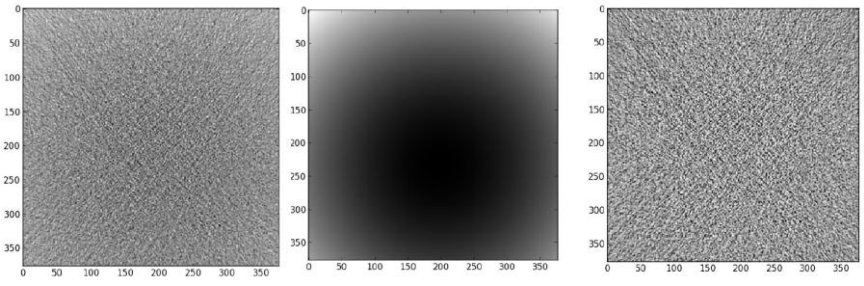
Noise Power Spectrum

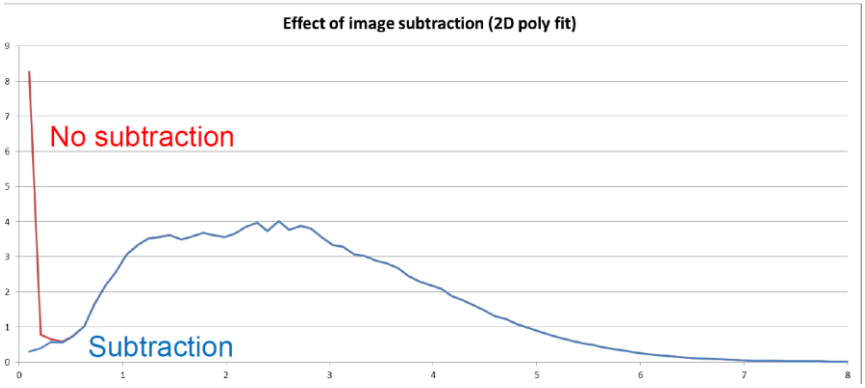


Normalized NPS = NPS/AUC
(AUC = Area Under Curve)

2D POLYNOMIAL SUBTRACTION

Dersom området man regner ut NPS fra ikke er flatt – dvs. at middelverdien varierer forskjellige steder på bildet – kan man aktivere dette valget for å trekke fra et 2D polynom fra valgt Region Of Interest (ROI). Vær forsiktig med dette valget, da det er bruker mye CPU-tid.



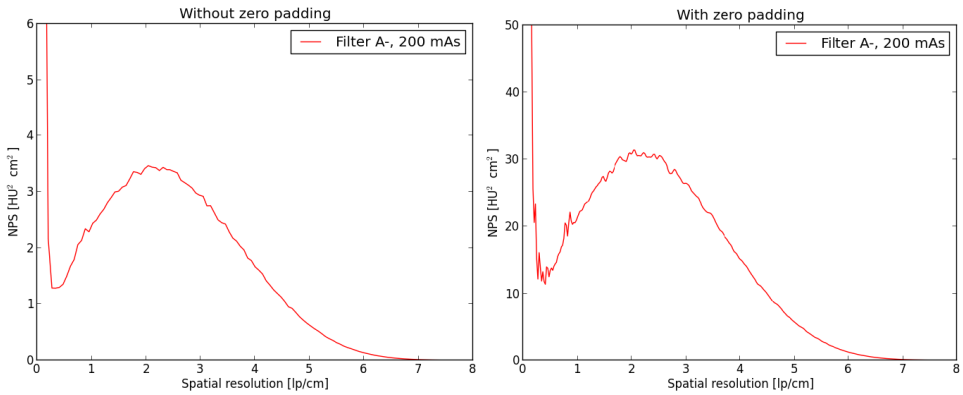


Effekten av en slik subtraksjon er at den kan fjerne NPS-toppen nær null-frekvensen.

ZERO PADDING

Et triks for å kjøre Fouriertransformasjonen med høyere nøyaktighet. Dersom opprinnelig ROI er på $m \times n$ piksler, vil den danne et nytt bilde på $3m \times 3n$ piksler, hvor originalbildet er sentrert i midten og de nye verdiene er satt til 0. Dette trikset er velkjent, se denne siden.

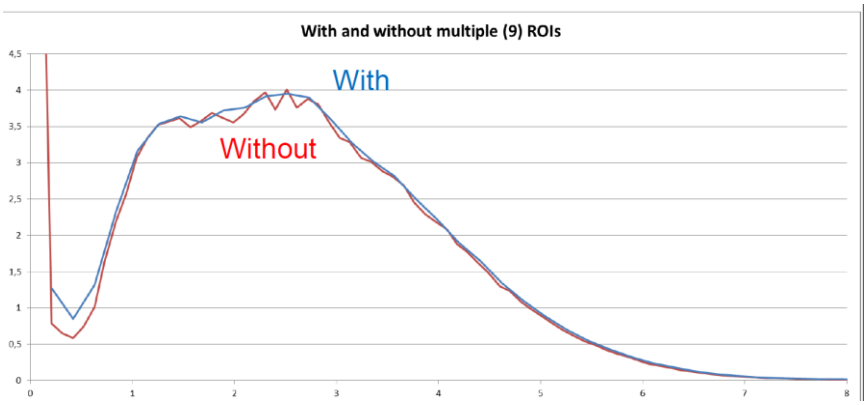
Resultatet er at NPS-kurven får høyere oppløsning, men med litt høyere støy.



9 SUB-ROIS

En måte å få mer statistikk (og en mykere kurve) fra bildedataene er å dele opp én ROI i flere mindre ROI-er, og finne gjennomsnittet til slutt. Med dette valget aktivert vil valgt ROI deles opp i 9 mindre ROI-er på følgende måte:

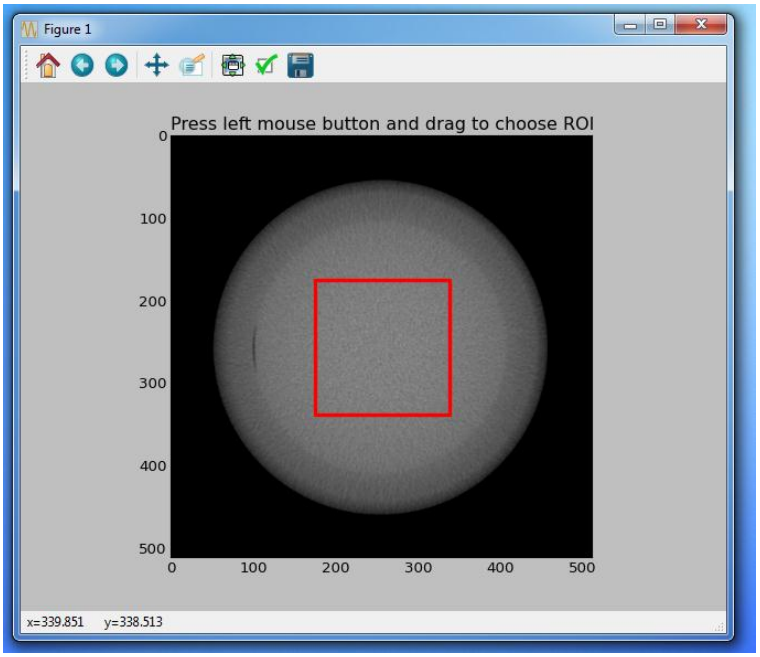
- Del ROI opp i 4 horisontalt og vertikalt, slik at man får 16 ikke-overlappende ROI'.
- Sett sammen fire slike stykker (2×2 ROI') øverst i venstre hjørne, øverst i midten, øverst i høyre hjørne, i midten på venstre side, i midten, i midten på høyre side og nederst på venstre side, nederst i midten og nederst til høyre.
- Resultatet er 9 sub-ROI med delvis overlapp
- Gjennomsnittet av 9 NPS-kurver legges sammen.



CHOOSE ROI ON SCREEN

Med dette valget kan man selv velge hvilken ROI som skal brukes i NPS-utregningen. Utvalget må være kvadratisk. Uten dette valget brukes hele bildet i utregningen.

Når man aktiverer dette valget vil det dukke opp et nytt vindu. Her drar man et rødt kvadrat for å plassere ROI, på *det første bildet i den første mappen*.



Jo større ROI man velger, jo bedre statistikk får man (og mykere kurve), men dess lengre tid tar programkjøringen.

ROI DC Correction

(Nesten) obligatorisk valg. Trekker fra gjennomsnittsverdien fra bildet, slik at FFT utføres på et «zero mean»-bilde. Legg merke til at dette er en enklere utgave av 2D polynomial subtraction.

Save Median Value

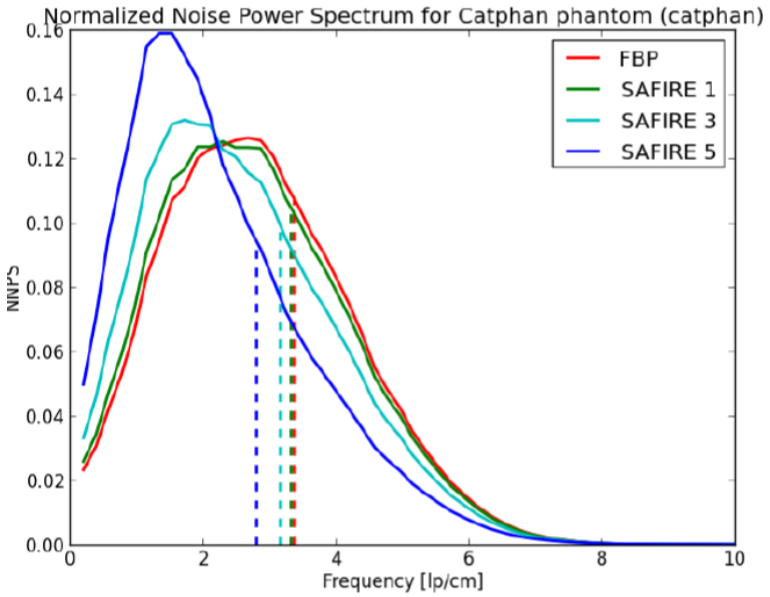
Om man er interessert i hvor «massen» til NPS-kurven ligger vil dette valget regne ut ved hvilken frekvens 50 % av arealet er under, og 50 % av arealet er over. Dette er median-verdien til NPS-kurven. Dersom den ligger mellom to frekevenspunkter, vil den interpolerte verdien brukes.

Slike verdier er enklere å bruke for å sammenlikne hvor skarpe forskjellige filtre er.

Verdiene lagres i en semikolon-separert fil `median_values.csv`, som inneholder «Filter; Iterative Strength Level; Median Value [lp/cm]; Date [YYYY-MM-DD HH:MM:SS]».

Show Median Value

Se «Save Median Value». Dersom denne er aktivert vil det dannes en vertikal linje i plottet som viser hvor medianverdien er.



Andre innstillinger

I tillegg til forskjellige måter å gjøre NPS-beregningene på, er det flere innstillinger når det gjelder visning og lagring av resultatene.

CHOOSE OUTPUT FORMAT

Under denne overskriften velger man hvordan man ønsker å lagre resultatene. Man kan hake av flere muligheter samtidig. Dersom man velger output som PNG eller CSV, vil det komme opp et spørsmål om hvor man ønsker å lagre filen. Det er mulig å bruke variabelnavn som sykehus (%institute), modell (%model) og rekonstruksjonsfilter (%filter) i filnavnet, se diskusjon under plot-tittel.

- On screen: Resultatene kommer opp i et interaktivt vindu, og man kan velge størrelse / farger og akseinnstillinger selv, i tillegg til å lagre resultatet til en bildefil.
- PNG file: Lagrer plottet som en bildefil (PNG). Man kan ikke gjøre endringer i akser, farger o.l. når dette er utført. **Tips: Det går også an å lagre som PDF-filer, da skriver man bare inn filnavn.pdf.**
- CSV file: Det lages én semikolon-separert fil for hver NPS-kurve. Her inngår x-akse- og y-akse-verdier. Husk å bruke variabelnavn (som nevnt over) for å skille filnavnene.

PLOT OPTIONS

Man kan velge tittel på plottet, samt på x- og y-aksene. Dersom man velger normalisert NPS vil navn (NNPS) og enhet (cm) endres automatisk. Her er det mulig å bruke variabelnavn som automatisk henter verdiene fra den *siste* DICOM-filen som ble kjørt (dersom man har flere leverandører og velger %manufacturer som tittel, vil den siste være gjeldende).

Følgende variabelnavn er mulige, hvor verdiene hentes fra DICOM-merkelapper:

- %filter: Rekonstruksjonsfilter
- %exposure: mAs
- %kvp: kV_p
- %model: Modell (f.eks. Somatom Definition AS+)
- %institution: Sykehus
- %studydate: Dato for opptak
- %slicethickness: Snittykkelse i mm
- %IRLevel: Iterativ rekonstruksjonsstyrke, som et tall fra 1-5 (SAFIRE), 1-7 (iDose) og 1-3 (AIDR3D). Om flere muligheter er ønsket her, finn ut hvordan den lagres i DICOM-filen (f.eks. en privat merkelapp) og legg til i `calc.py`, under funksjonen `getIRLevel()`.

KRAV TIL PYTHON:

Man trenger følgende pakker for å kjøre programmet:

- `dicom`
- `matplotlib`
- `tkFileDialog`
- `Tkinter`
- `numpy / scipy`

BESKRIVELSE AV PROGRAMFILENE

Noise Power Spectrum Calculator består av fem filer.

GUI.PY

I denne filen ligger det grafiske grensesnittet, dvs. hovedskjermen hvor man setter alle innstillingene. Den inneholder én klasse, som arver `Frame`-klassen til Tkinter. Jeg har prøvd å holde den noenlunde kommentert.

I tillegg er det noen funksjoner for å velge ROI grafisk, som laster klassen fra `classes.py`.

CALC.PY

Denne filen inneholder flere hjelpefunksjoner.

- `func()`, `easyfunc()` og `poly_sub()` for å gjøre 2D polynom-subtraksjon av et bilde (ved hjelp av `scipy.optimize.curve_fit`)
- `getIRLevel()` for å finne styrken på iterativ rekonstruksjon fra DICOM-header
- `noisePowerRadAv` for å finne det radielle gjennomsnittet av et 2D array

CLASSES.PY

Inneholder en modifisert utgave av `matplotlib.widgets.RectangleSelector`, hvor det kreves at høyden og bredden på valgt ROI er lik.

NPS.PY

Denne filen gjør selve NPS-beregningen. Den tar argumentene `input_file_list` som inneholder filnavnene til alle filene som skal inngå i én kurve, `rectXY` som er ROI-dimensjonene (eller `False` dersom hele bildet skal brukes) og listen over `options` fra GUI-et, som bestemmer hvilke beregningsinnstillinger som skal brukes.

Grovt fortalt fungerer programmet slik:

1. Den laster inn én DICOM-fil
2. Pikseldata lastes inn i et numpy-array, og det gjøres korreksjoner for RescaleSlope og RescaleIntercept (for å få HU-tall) + DC korreksjon om det er valgt
3. Dersom ROI er valgt, innsnevrer man bildet til de dimensjonene
4. Dersom 9 sub-ROI er valgt, deles bildet inn i mange små ROI-er, hvor NPS-analysen gjøres separat og summeres til slutt
5. Dersom 2D poly er valgt, gjøres det nå
6. Dersom Zero Pad er valgt, gjøres det nå
7. numpy-funksjonene for fft2 og fftfreq brukes på resultatet, samt formelen for NPS: $\mathrm{NPS} = |\mathrm{FFT}(img)|^2 \times \mathrm{pixelSpacing}^2 / N_{pixel\_x} N_{pixel\_y}$
8. Funksjonen noisePowerRadAv lager en 1D kurve fra 2D FFT-bildet
9. Dette gjentas for alle bildene, og snittresultatet er NPS-kurven
10. Om median-verdien skal regnes ut gjøres det her. Den finner totalt areal med trapes-regelen (fra `scipy.optimize.trapz`). Så gjentar den prosessen med del-

arealet, og finner ut når $\text{delareal}(x) = 0.5 \times \text{totalareal}$ gjelder. Det er vår medianverdi.

11. Så returnerer den NPS-kurven, bildeinfo (DICOM) samt median-verdien.

MAIN.PY

Selve programfilen. Den begynner med å kjøre gui.py for å få alle innstillingene, og så spør den etter hvilke bilder eller mapper som skal brukes.

Den ordner mappene etter filer, og for hver mappe gir den alle filene til nps.py. Når den får tilbake NPS-kurvene, finner den litt info til plotting, før den plotter og lagrer som spesifisert i innstillingene.

Helge Pettersen