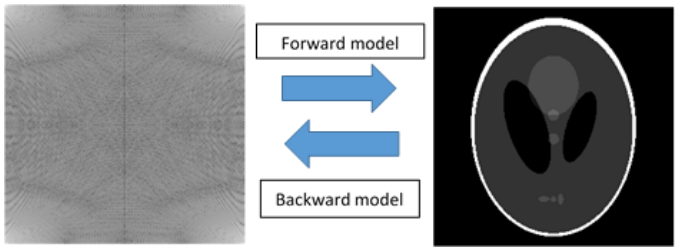
Projekt 4: Rekonstruktion af medicinsk scanningsdata
02525 - Januar 2018

1 Projektbeskrivelse

Dette projekt omhandler de modeller, der ligger til grund for rekonstruktioner af billeder fra medicinske scannere. Fysikken og matematikken bag er forskellig for alle typer af scannere, og denne opgave vil således kun omhandle rekonstruktion af MR billeder. Hovedspørgsmålet er altså, hvordan kommer vi fra detekteret data til et billede af, hvad der er inde i kroppen?

Projektet bygger på den diskrete Fourier transformation (DFT), som blev introduceret i starten af 13-ugers perioden. Vi vil udvide formuleringen af denne til to dimensioner, og derfor kan vi altså benytte transformationen på billeddata. Det skal nævnes, at denne opgave betragter problemet i generelle termer. Software i nutidens MR scannere har langt flere detaljer i rekonstruktionsprocessen.

Figur 1: MR scanner ved Herlev Hospital.



Figur 2: Forward-backward models.

1.1 Rekonstruktion af scannet data

I en MR scanner detekteres radiobølger, som udsendes når visse atomkerner sættes i svingninger vha. magnetiske radiobølger. MR står for *magnetisk resonans*.

Rekonstruktionsmetoden, altså at gå fra detektioner af radiobølger til et signal, der repræsenterer, hvad der er inde i kroppen, bygger i høj grad på den inverse diskrete Fourier transformation (DFT). Dette skyldes, at det den detekterede data er DFT transformeret data.

Forward og backward modeller I rekonstruktionsproblemer taler vi ofte om en forward og en backward model. Her vil forward modellen være den model der ligger til grund for detektioner, hvorimod backward modellen er rekonstruktionen. Figur 2 illustrerer dette. I tilfældet med rekonstruktion af MR scanninger vil forward modellen være en todimensionel DFT af et billede, og backward modellen er således den inverse, todimensionelle DFT.

Detekteret vs. simuleret data Det er normal praksis, at man tester rekonsturktionsalgoritmer på kendt eller simuleret data. Data fra scannere vil altid være påvirket af støj, hvorimod simuleret data oftest vil være rent. Dette kan man tage højde for ved at addere forskellig støj til det simulerede data. Ved at addere støj af forskellig art og niveau til vores simulerede data, vil vi kunne vurdere hvor god rekonsturktionsmetoden er til at løse problemet under forskellige forhold.

Når der arbejdes med simuleret data er det muligt at sample frekvensspektret meget, meget fint. Dette vil ikke nødvendigvis være tilfældet når det er

data fra en MR scanner som er til rådighed. Derfor indeholder denne opgave en undersøgelse af, hvordan andelen af samplinger influerer rekonstruktionen.

2 Rekonstruktion af MR data

Selve opgaven fremgår herunder. Opgaven indeholder syv forskellige delopgaver. Seks delopgaver, hvor rekonstruktionsmetoden bliver undersøgt, og slutteligt en delopgave, hvor ukendt data skal rekonstrueres.

2.1 Testbilleder - simuleret data

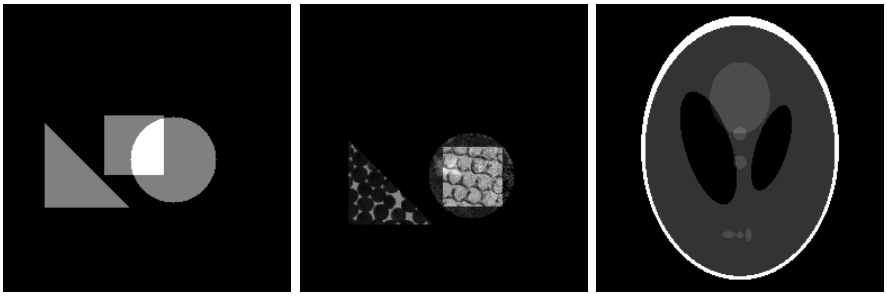
Første skridt i denne opgave er, at generere to testbilleder, som kan bruges, når rekonstruktionsmetoden skal testes. Disse billeder vil fungere som simuleret data i resten af opgaven.

Generering af testbilleder Det første testbillede som skal genereres, består af forskellige geometriske objekter. I billedet skal der således både være en trekant, en firkant og en cirkel, som det ses i figur 3. Et sådant billede vil give en indikation af, hvilke former/kanter i et billede, som rekonstruktionsmetoden har let eller svært ved at rekonstruere.

Implementér en funktion, der returnerer et kvadratisk billede af størrelsen $K \times K$ pixels, der indeholder en cirkel, en trekant og en firkant. Kald funktionen `generate_simdata`. *Funktionen skal tage billeddimensionen K som inputparameter. Centrum eller evt. et givet hjørne for de tre geometriske figurer skal genereres tilfældigt og størrelsen af dem skal være afhængig af billeddimensionen. Tankegangen i genereringen af billedet skal fremgå af kodens dokumentation.*

Vi udvider nu vores geometriske billede således, at hvert objekt i billedet repræsenterer en tekstur.

Download zip-filen med tre teksturbilleder fra Campusnet. Pak dem ud, så de ligger i en mappe samme sted som I har jeres scripts og functions liggende. De steder i det første testbillede, hvor de geometriske figurer er til stede, skal nu kunne erstattes med de tre teksturer såfremt dette er ønskes. Integrer dette i funktionen ovenfor vha. inputparametre, så funktionen kan



Figur 3: Simulerede billeder med geometriske figurer uden og med tekstur og *Shepp-logan* fantom.

generere geometriske figurer både med og uden tekstur afhængigt af antal input parametre. Den første input-parameter skal stadig være en skalar `K`*, mens den anden valgfrie input-parameter,* `texFiles`*, skal være en string, der angiver stien til teksturbillederne. Det kan være en fordel at tjekke hvad funktionerne* `nargin` *og* `dir` *gør, når teksturbillederne skal indlæses. Sørg for at gemme dit testbillede lokalt på din computer. Se* `imwrite` *for dette.*

Vi vil også gerne kunne teste vores rekonstruktionsalgoritme på data, som ligner virkelig data. Dette kan vi bl.a. gøre vha. Shepp-logan fantomet, som er indbygget i Matlab med funktionen `phantom`.

Generer et Shepp-logan fantom af størrelsen 256×256*. Sørg for at gemme dit testbillede lokalt på computeren.*

2.2 2D DFT

Den diskrete Fourier transformation bliver ofte anvendt i signalbehandling, da Fourier-repræsentationen beskriver hvilke frekvenser, der er tilstede i data. Dette så vi også i arbejdet med audiotuneren i 13-ugersperioden. Frekvenser er lette at tolke når vi snakker om lydsignaler, hvor vi ved at høj frekvente signaler svarer til høje toner. I billeder er der også både høj- og lavfrekvente dele. De højfrekvente dele af et billede er der, hvor der er skarpe kanter og overgange i et billede, hvorimod de lavfrekvente områder er der, hvor der er bløde overgange eller ensartethed i billedet.

Den kontinuerte todimensionelle Fourier transformation af en funktion $f(x, y)$

er givet ved

$$F(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-2\pi i(ux+vy)}\,dx\,dy. \tag{1}$$

Da vi arbejder med billeddata, bliver vi nødt til at have en diskret formulering af Fourier transformationen. For et billede med endelig størrelse, $M \times N$ pixels, er den diskrete Fourier transformation i to dimensioner givet ved:

$$F(u,v) = \frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n)e^{-2\pi i(\frac{um}{M}+\frac{vn}{N})}. \tag{2}$$

I denne opgave arbejdes udelukkende med kvadratiske signaler af størrelsen $K \times K$, og den todimensionelle DFT kan da formuleres som

$$F(u,v) = \frac{1}{K^2}\sum_{m=0}^{K-1}\sum_{n=0}^{K-1} f(m,n)e^{-2\pi i(\frac{um}{K}+\frac{vn}{K})}. \tag{3}$$

Transformation af data I Matlab kan vi beregne 2D DFT vha. funktionen `fft2`. Yderligere benytter vi også ofte `fftshift`, som gør DFT signalet circulært. Når det transformerede data skal visualiseres, kan man med fordel bruge funktionerne `abs`, `log` og `imagesc`.

Beregn 2D DFT på testbillederne. Vis størrelsen (`abs`) af DFT signalerne som billeder. Kommentér på resultaterne.

2.3 Støj

Vi vil i dette projekt kigge på normalfordelt støj. Vi betragter det som en additiv model. Dette betyder, at det endelige støjfyldte signal er givet i form af det oprindelige billede (altså DFT signalet), $\boldsymbol{F}$, og et billede af støj, $\boldsymbol{N}$:

$$\boldsymbol{F}_{noise} = \boldsymbol{F} + \sigma\boldsymbol{N}, \tag{4}$$

hvor σ angiver niveauet af støj.

Normalfordelt støj Normalfordelt støj, eller *hvid støj* som det også kaldes, er givet ud fra en normalfordeling med et gennemsnit på0 og en varians på1. I Matlab genereres standard normalfordelte variable med funktionen `randn`. Når støjen adderes til et billede er det vigtigt at tage højde for pixelværdierne i billedet. For et billede bestående af pixels med reelle værdier kan vi addere normalfordelt støj ved et givet procent niveau τ med følgende kode:

```
r = randn(n,n); r = r/norm(r,'fro');
e = tau*r*norm(im,'fro');
im_noisy = im + e;
```

Forklar hvad de tre ovenstående linjer kode beregner. DFT signalerne der i denne opgave skal adderes støj til er komplekse. Hvordan vil koden, der generer et støjbillede til disse se ud? Hvad er det vigtigt at tage højde for?

Implementer en funktion `addnoise`*, som tager et billede og procentvis støjniveau som input parametre. Funktionen skal returnere det støjfyldte billede. Det er vigtigt at funktionen tager højde for om billedet er komplekst. Yderligere skal det være muligt at angive procentdel som enten et decimaltal eller på et interval fra 0 til 100.*

For hvert DFT transformeret testbillede, generer et billede af normalfordelt kompleks støj. Sæt støjniveauet til 1% og addér støjbilledet til DFT signalet for testbilledet. Vis de støjfyldte Fourier signaler.

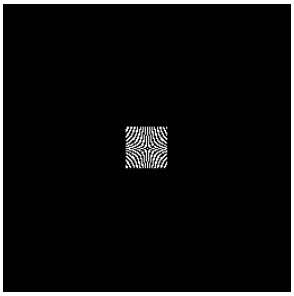
2.4 Rekonstruktion af simuleret data

Tilsvarende den todimensinelle DFT givet i (2.2), er den inverse todimensionelle DFT (IDFT) givet ved

$$f(m,n) = \sum_{u=0}^{K-2} \sum_{v=0}^{K-1} F(u,v) e^{2\pi i \left(\frac{u}{K}m + \frac{v}{K}n\right)}. \tag{5}$$

Her antages igen, at der arbejdes med kvadratiske billeder af størrelsen $K \times K$. 2D IDFT skal bruges til at transformere fra Fourier rummet til det oprindelige koordinatsæt.

Rekonstruktion I Matlab kan den inverse DFT udføres med funktionen `ifft2`. Vær opmærksom på, at rekonstruktionen sagtens kan indeholde



Figur 4: *Zero-padding* af DFT signal.

komplekse tal, og for at vise rekonstruktionen er det således størrelsen af rekonstruktionen, der skal vises (`abs`).

Rekonstruer de støjfrie testbilleder.

Rekonstruer de støjfyldte testbilleder.

Vis resultaterne af de to ovenstående rekonstruktioner. Kommenter på resultaterne.

2.5 Sampling i DFT rummet

Det er kostbart og tidskrævende at sample alle frekvenser, så derfor vælger man i nogle tilfælde kun at sample en vis andel af frekvenserne i DFT rummet. Det er en balancegang at få tilstrækkeligt data til at kunne opnå en brugbar rekonstruktion og tiden det tager at optage signalet.

Denne sampling af frekvensrummet skal vi nu forsøge at imitere. Dette gøres ved at sætte en stor del af DFT koefficienterne til 0, og herefter rekonstruere. Se figur 4 for en illustration af hvordan DFT signalet herefter vil se ud.

Implementer en funktion, `signal_limited` *, som tager to input parametre;* `signal` *og* `frac`*. Funktionen skal returnere et DFT signal, hvor den givne andel af DFT koefficienter er sat til 0. Andelen af koefficienter bliver bestemt af* `frac`*, som beskriver andelen af samplinger fra det originale signal, som vi gerne vil beholde. Det antages, at input-signalet er kvadratisk og at andelen af samplings er mindre end 1.*

Indfør tjek af variable i din implementering af funktionen ovenfor. Hvilke variable skal der testes på?

Test implementeringen af funktionen for de to testbilleder. Evt. både med og uden støj. Kommenter på resultaterne. Hvad sker der med rekonstruktionen når der ikke samples i det fulde DFT signal?

2.6 Fejlmål

For at kunne vurdere, hvor godt metoden til rekonstruktion virker, indføres et normaliseret fejlmål. Her defineres $\boldsymbol{X}$ som det kendte data (*ground truth*), og $\hat{\boldsymbol{X}}$ er det rekonstruerede billede. Fejlmålet defineres som

$$\varepsilon = \frac{\|\boldsymbol{X} - \hat{\boldsymbol{X}}\|_{fro}}{\|\boldsymbol{X}\|_{fro}}, \tag{6}$$

hvor $\|\boldsymbol{A}\|_{fro}$ betegner Frobenius normen. Frobenius normen, som tidligere er blevet benyttet ved normalisering af støjsignal, er givet ved

$$\|\boldsymbol{A}\|_{\text{fro}} = \left(\sum_{i=1}^{M} \sum_{j=1}^{N} |a_{ij}|^2 \right)^{1/2}, \tag{7}$$

hvor a_{ij} er elementet i den i'te række og j'te søjle i matricen $\boldsymbol{A}$.

Rekonstruktionsfejl I Matlab kan man nemt beregne normer ved hjælp af funktionen `norm`.

Implementer fejlmålet i (2.6) *i en Matlab funktion. Kald denne function* `error_measure`. *Hvad skal funktionen have af input variable og hvordan skal disse tjekkes?*

Test implementeringen af `error_measure` *ved at beregne rekonstruktionsfejlen på en af rekonstruktionerne fra et af de to foregående afsnit.*

Lav en vektor, som repræsenterer forskellige niveauer af støj. For alle støjniveauer, skal du nu lave en rekonstruktion og herefter finde rekonstruktionsfejlen. Visualisér fejlen ved de varierende støjniveauer med en graf og billeder.

Lav en vektor, som repræsenterer forskellige andele af samplings af det 'rene' DFT signal. For alle andele, skal du nu lave en rekonstruktion og herefter finde rekonstruktionsfejlen. Visualisér fejlen ved de varierende samplinger med en graf og billeder.

Genbrug nu de arrays, som repræsenterer forskellige andele af samplings af DFT signaler og støjniveauer. For alle andele og hvert støjniveau beregnes rekonstruktionen og rekonstruktionsfejlen. Visualisér fejlen ved de varierende samplinger med en graf og udvalgte billeder af rekonstruktionerne. Forklar tendenserne i fejlkurverne.

2.7 Rekonstruktion af ægte data

Denne sidste delopgave omhandler rekonstruktion af et MR scannet volumen. På Campusnet findes der flere .mat-filer, som indeholder et antal slices af 2D DFT data. To af filerne indeholder kendte objekter, hvorimod de to sidste er ukendte. Lav evt. et nyt script for denne del af opgaven.

Rekonstruktion af volumen Et scan af et volumen består af en stak af 2D billeder. Når flere slices af et volumen skal rekonstrueres skal der udføres en 2D DFT beregning for hvert slice for at rekonstruere.

Implementer en funktion, `recon_volume`, *som kan rekonstruere en række af MR signaler.Inputparametrene skal være et tredimensionelt array, der indeholder slices af et MR scan, og en vektor, der beskriver hvilke slices, der skal rekonstrueres. Sørg selv for at lave korrekte variabeltjek. De rekonstruerede slices skal returneres i et tredimensionalt array.*

Musehjerte og hoved På Campusnet findes to filer, mouse.mat og head.mat. Disse to filer indeholder data fra MR scanninger af henholdsvis et musehjerte og en menneskelig hjerne. Slices i scan fra musehjertet er en tidsserie af det samme sted på hjertet.

Rekonstruer alle slices i MR scanningen af musehjertet vha. funktionen `recon_volume`.

Rekonstruer alle slices i MR scanningen af hovedet.

For en volumetrisk rekonstruktion er det nyttigt at kunne se rekonstruktion af forskellige planer i volument. Derfor skal der nu implementeres en funktion, som kan returnere ønskede slices i xy-, xz-, og yz-planerne. Dette kaldes normalt *ortho-slices*.

Implementér en funktion `ortho_slices`*, som returnerer tre ortho-slices,* `O1, O2, O3`*. Hver af disse orthoslices er et todimensionelt array. Funktionen*

skal uderover et volumen tage tre inputparametre; `s1, s2` *og* `s3`*, som hver repræsenterer, hvilket slice i de tre planer, som skal udtages.Undersøg hvad funktionen* `squeeze` *gør og brug den evt. i implementeringen.*

Test implementeringen af `ortho_slices` *på rekonstruktionerne af hovedet. Vis de tre slices vha.* `subplots` *i én figur.*

Ukendt data På Campusnet finder du også filerne A.mat og B.mat, som hver indeholder et MR scan. Scanningen af objekt A har en voxelstørrelse på 0.273438 mm $\times$ 0.273438 mm $\times$ 0.273438 mm, og for objekt B er voxelstørrelsen 0.51 mm $\times$ 0.51 mm $\times$ 0.51 mm.

Rekonstruér nu disse datasæt. Hvad vil I gætte på, at objekterne er? Vurdér i gruppen hvordan det er bedst at visualisere disse rekonstruktionsresultater. Er der noget af det teoretiske fra første delopgave som kan testes på et eller begge af disse datasæt?

3 Rapport

Rapporten må maksimalt være 10 sider lang. Den skal indeholde følgende:

- Introduktion, kort teoribeskrivelse, analyseafsnit, diskussion og konklusion.
- Illustrerede resultater, hvor figurer indgår som en aktiv del af opgavebesvarelsen.
- Veldokumenteret kode i bilag - husk at skrive header på funktioner og ellers hold kommentarerne i koden kortfattede. (Hvis I benytter LaTeX, så inkluder evt. på mcode.sty (findes online), der kan embedde Matlab-kode i pdf'en med korrekt formatering.)