DMIP – Exercise

Sinograms and Filtered Backprojection (FBP) for Parallel Beam

Yan Xia, Marco Bögel
Pattern Recognition Lab (CS 5)

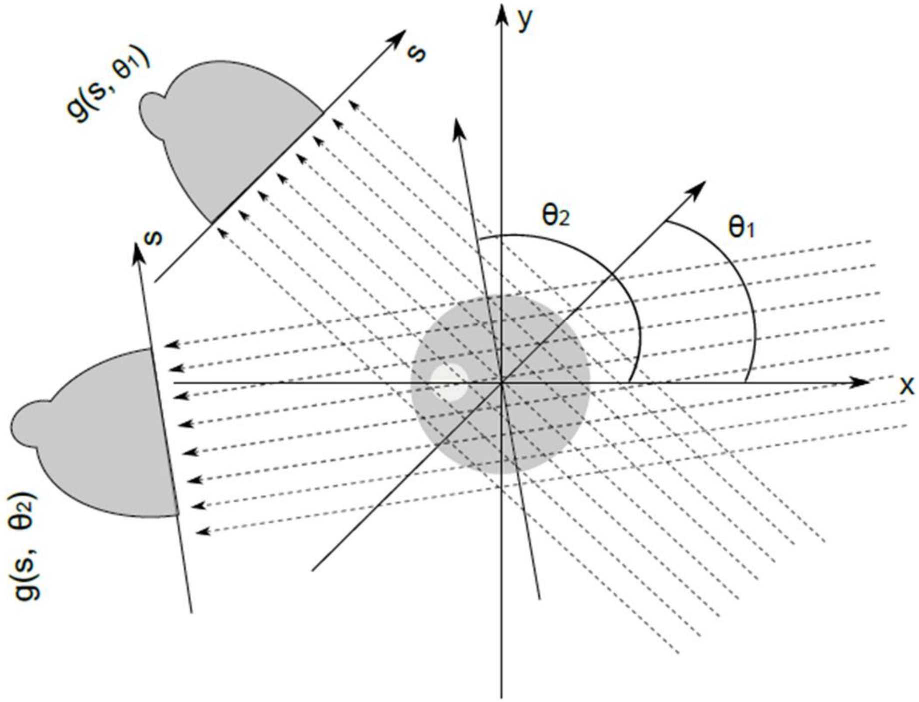
FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG
TECHNISCHE FAKULTÄT

Sinograms

- What is a projection?

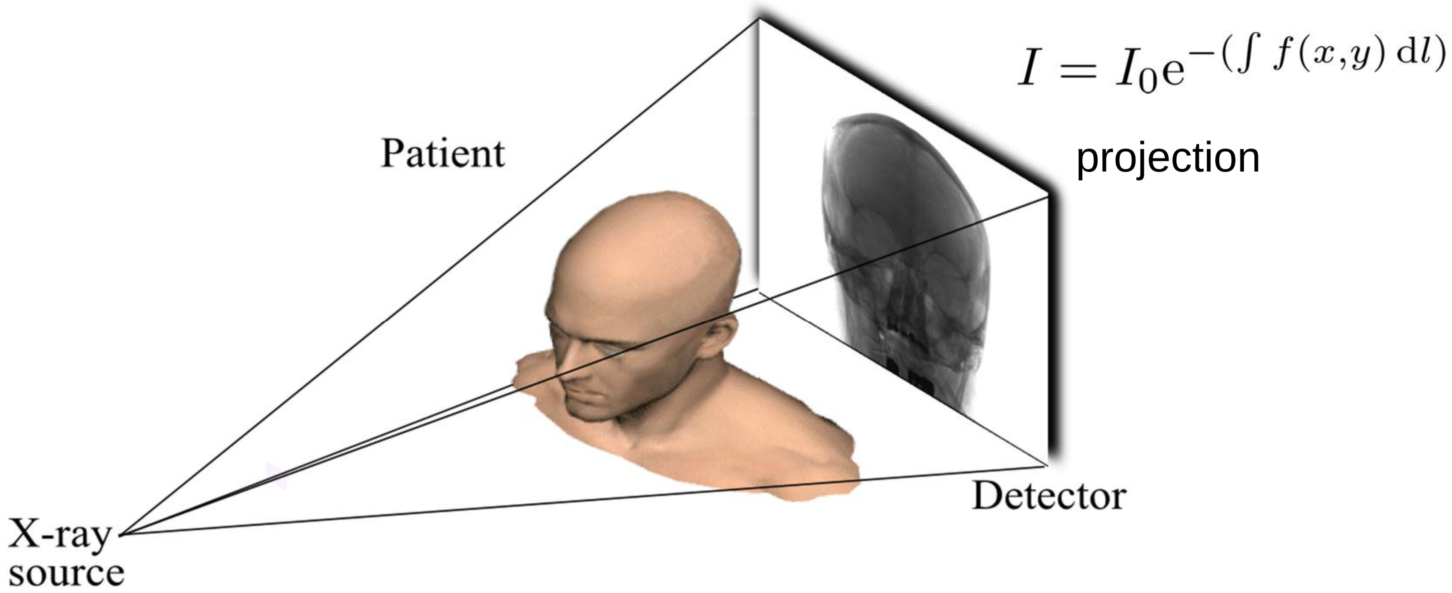
Sinograms

- What is a projection?
  - Mathematically, a projection is a line integral of a function



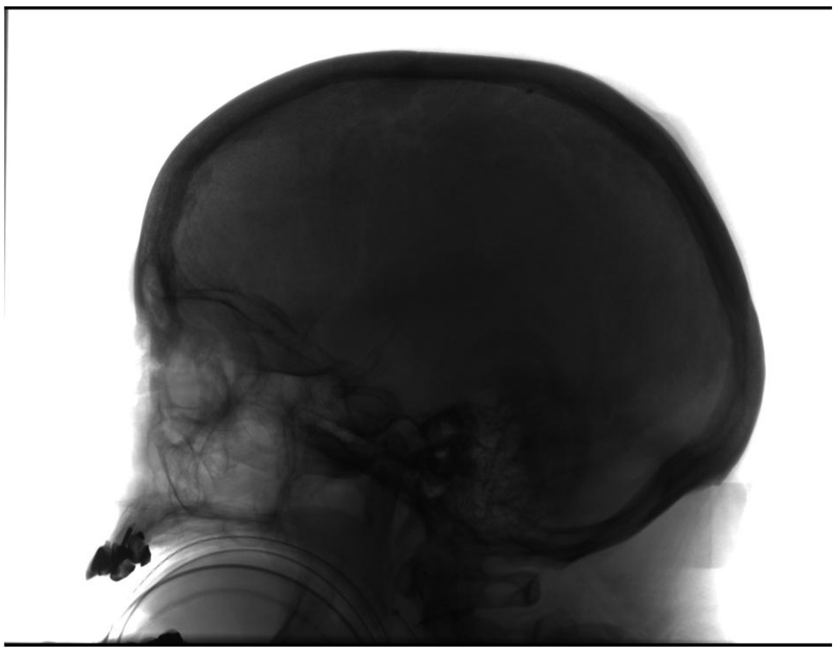
Sinograms

- What is a projection?
  - Mathematically, a projection is a line integral of a function
  - We use projection synonymous with X-ray projection



$$I = I_0 \mathrm{e}^{-\left(\int f(x,y)\,\mathrm{d}l\right)}$$

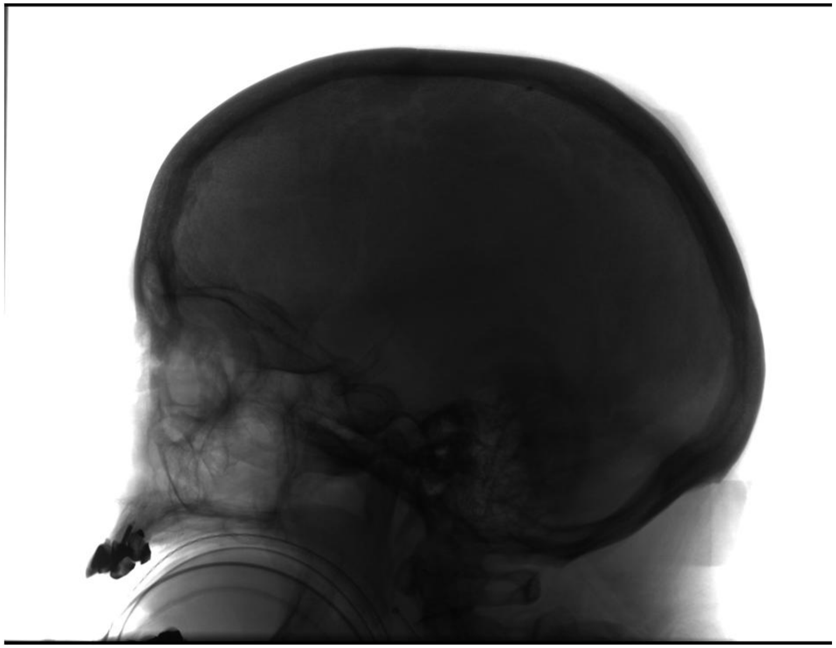
?



We get from detector

Line integral used for recon

$$I = I_0 \mathrm{e}^{-\left(\int f(x,y)\,\mathrm{d}l\right)} \Rightarrow \int f(x,y)\,\mathrm{d}l = -\ln\left(I/I_0\right)$$



We get from detector

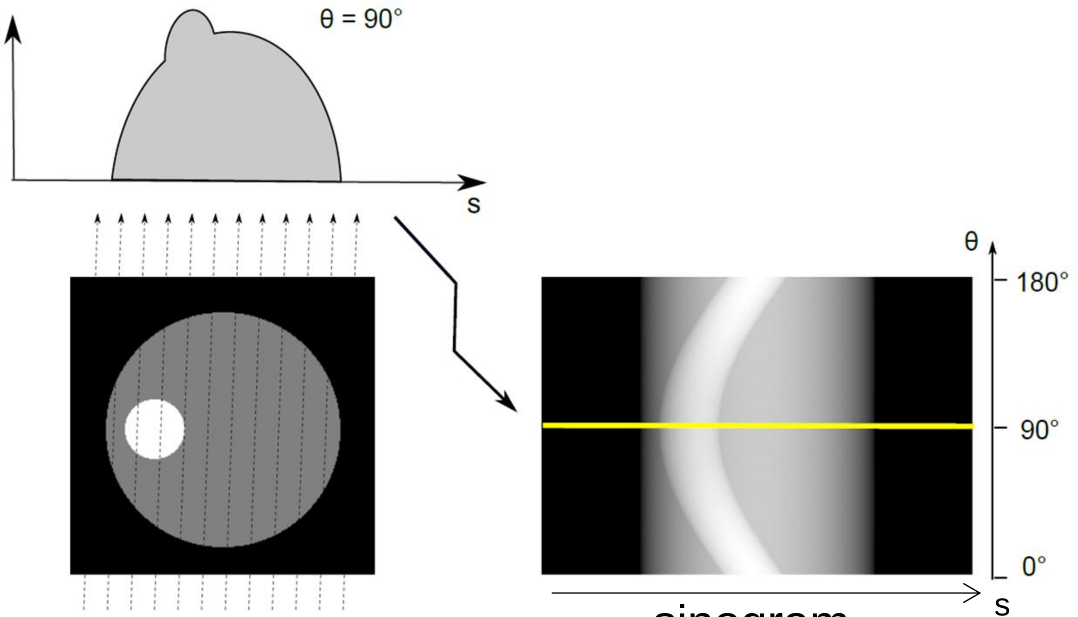
Line integral used for recon

Sinograms

- What is a sinogram and how does it relate to projections?

Sinograms

- What is a sinogram and how does it relate to projections?
  - A stack of all acquired projections sorted by their angle
  - A 2-D sinogram contains information from 1-D projections, i.e. all necessary information to reconstruct one 2-D slice



sinogram

Sinograms

- What is a sinogram and how does it relate to projections?
  - A stack of all acquired projections sorted by their angle
  - A 2-D sinogram contains information from 1-D projections, i.e. all necessary information to reconstruct one 2-D slice

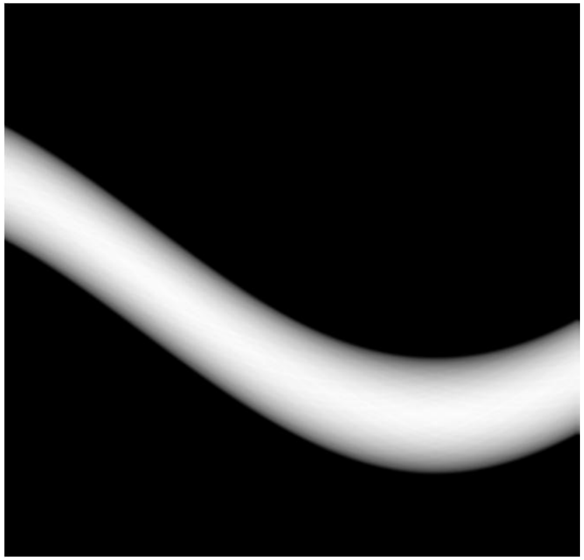
- Why is it called sinogram?

Sinograms

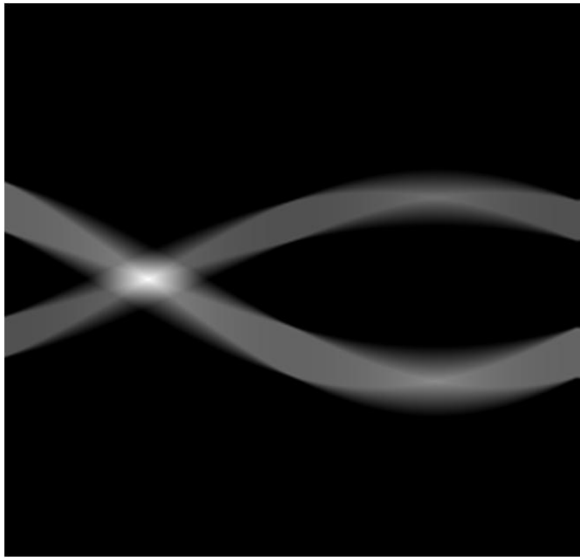
- What is a sinogram and how does it relate to projections?
  - A stack of all acquired projections sorted by their angle
  - A 2-D sinogram contains information from 1-D projections, i.e. all necessary information to reconstruct one 2-D slice

- Why is it called sinogram?
  - Because an off-centred object creates a trace that looks like a sine-wave

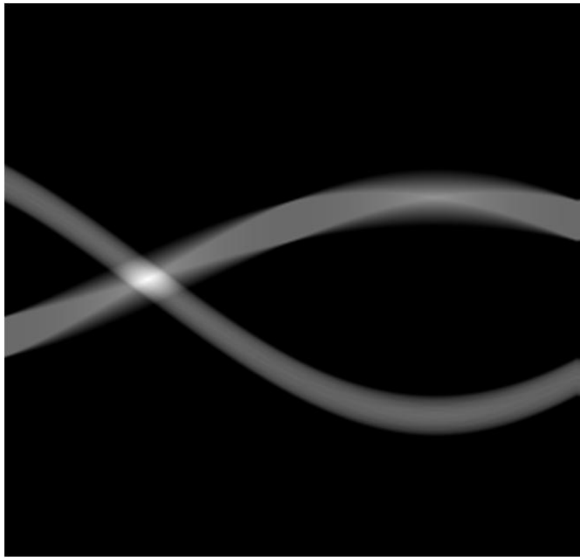
Sinograms



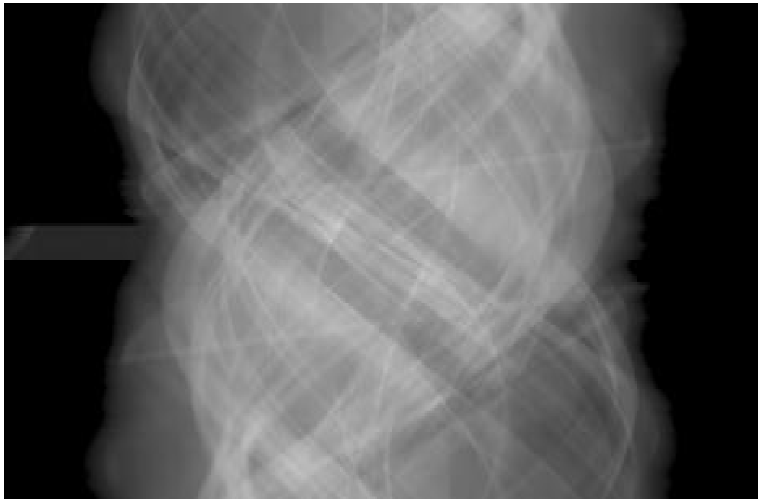
Sinograms



Sinograms



Sinograms



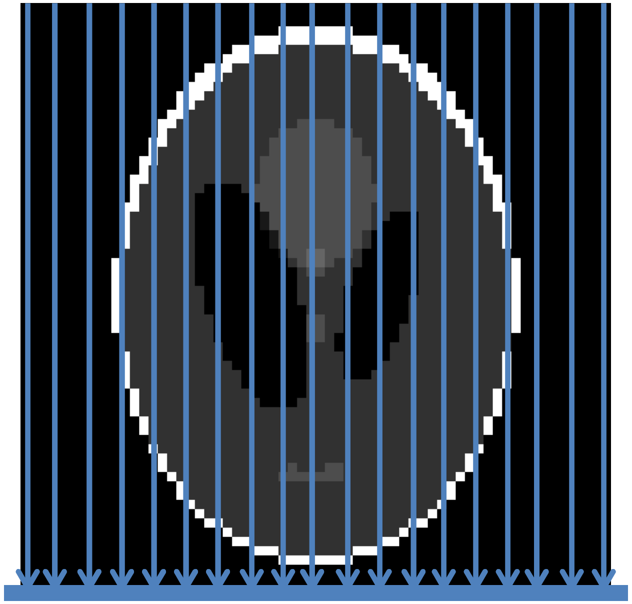
Scan Simulation

- We will scan a Shepp-Logan phantom

```
im = phantom(64);
```

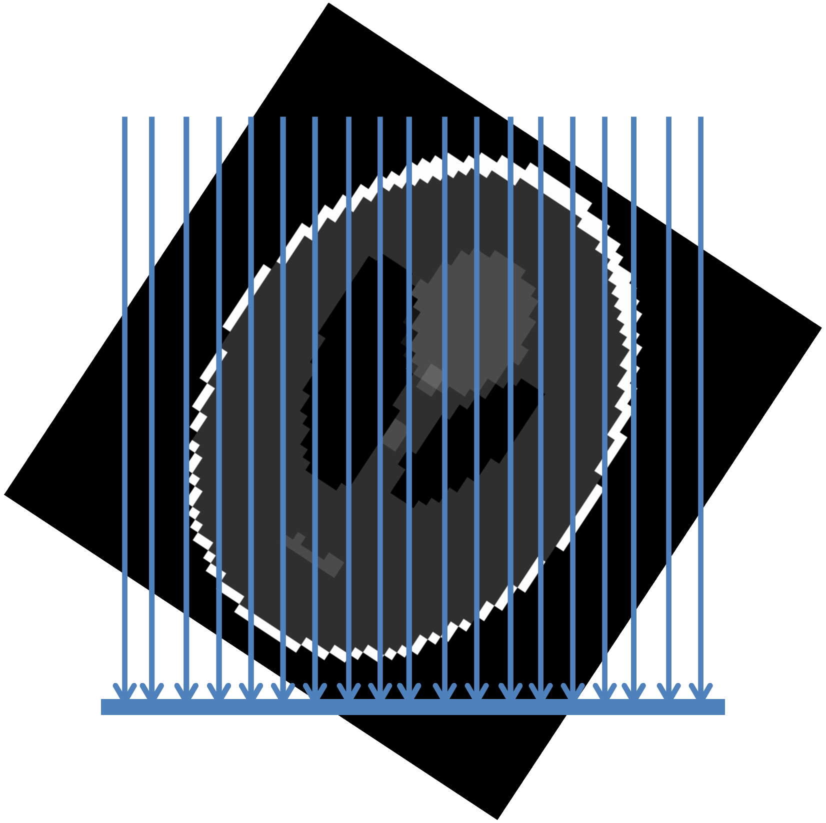
Scan Simulation

- Scanning is simulated by summing up columns



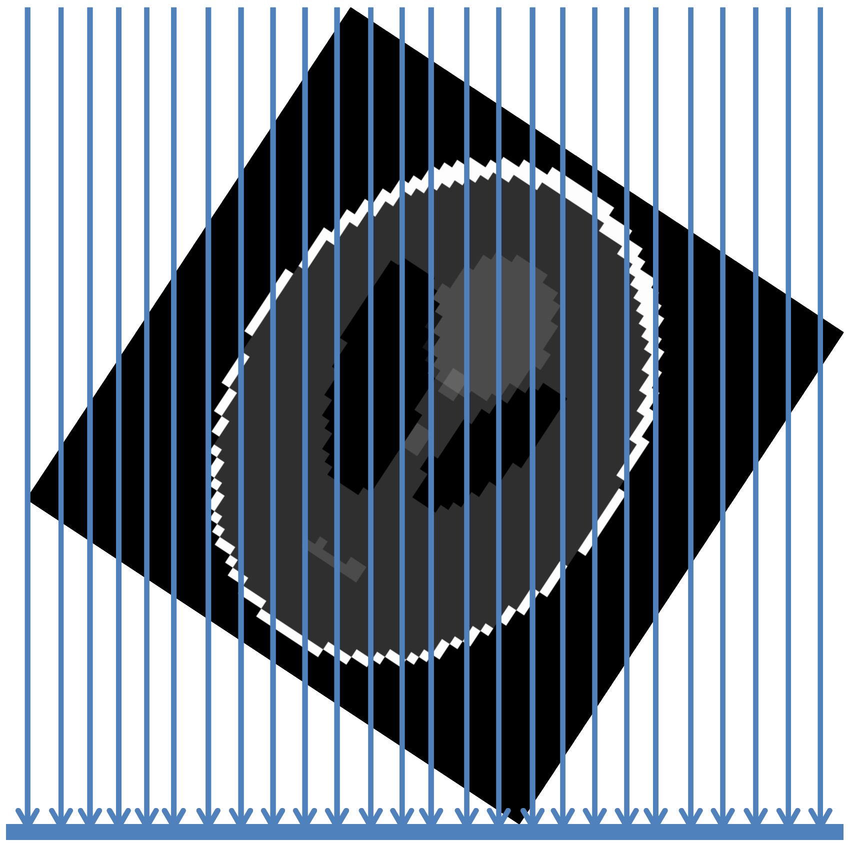
Scan Simulation

- We rotate the image instead of the detector



Scan Simulation

- Note: If we don't crop after rotation, we get different scan sizes and (unnatural) translations of the detector



Scan Simulation

- Task: Setup the scan parameters

```
% angleIncrement = ???;
% startAngle = ???;
phi = startAngle;
% numberOfProjections = ???;
```

Scan Simulation

- Task: Implement the actual scan simulation: Rotate the image, sum up the columns, save the current projection and set the next rotation angle.

```
for i=1:numberOfProjections
     ...
     % rI = ???;
     ...
     % Sum up columnwise -> parallel beam
     % projs{i} = ???;

     % Compute the next rotation angle
     % phi = ???;
end
```

Reconstruction

- Used Filtered Backprojection scheme:

Convolution in spatial domain

↓

Backprojection onto detector

Reconstruction

- Used Filtered Backprojection scheme:

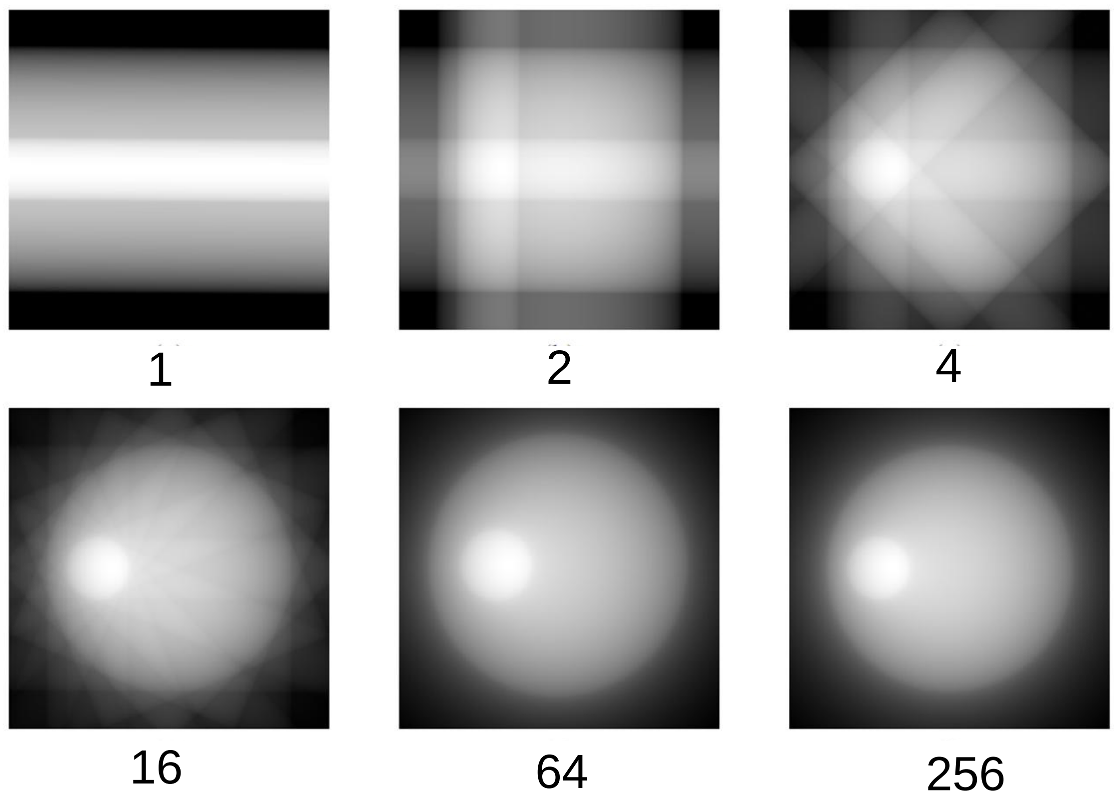
Convolution in spatial domain

↓

Backprojection onto detector

Filtered Backprojection for Parallel Beam

- For the Filtered Backprojection, why do we need a high pass filter? What would the reconstruction look like without filter?

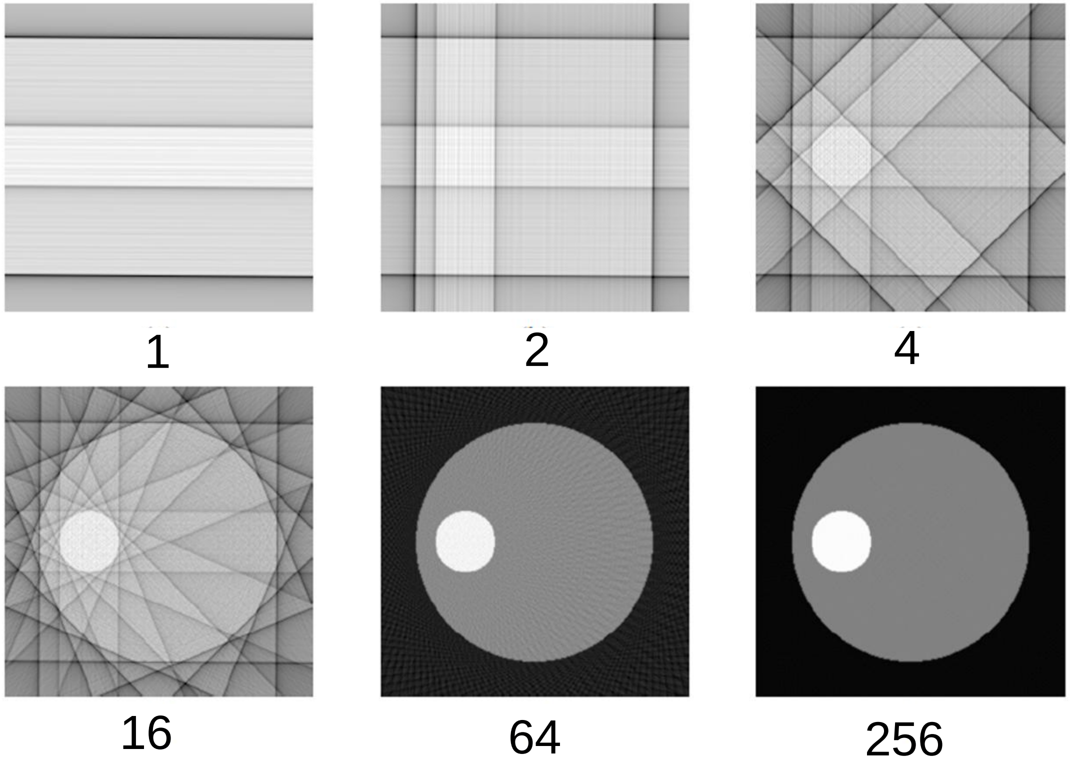


Filtered Backprojection for Parallel Beam

- For the Filtered Backprojection we can use different filter kernels. List them!

Filtered Backprojection for Parallel Beam

- For the Filtered Backprojection we can use different filter kernels. List them!
  - Most important are Ram-Lak and Shepp-Logan



Reconstruction

- Three convolution options are implemented.

```
if(fltr == 1)
    fltm = RamLak(60);
    proj = conv(proj, fltm, 'same');
elseif(fltr == 2)
    fltm = SheppLogan(60);
    proj = conv(proj, fltm, 'same');
else
    proj = proj;
end
```

Reconstruction

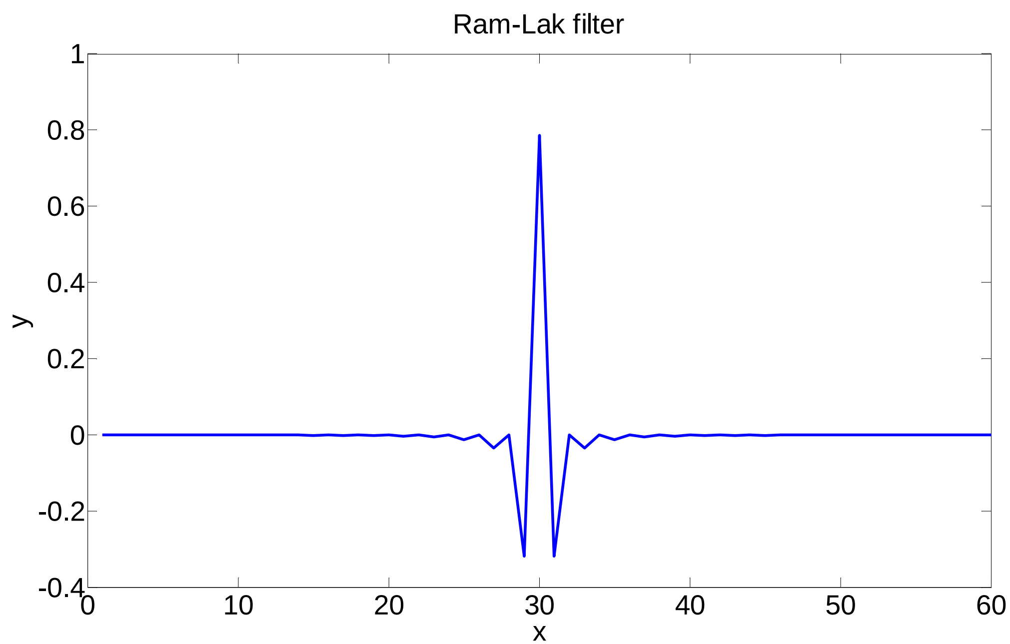
- Task: Implement the discrete spatial version of the RamLak filter.

```
function [ramlak] = RamLak(width)
%
%
%
% ???
%
%
%
%
end
```

Reconstruction

- Task: Implement the discrete spatial version of the RamLak filter.

$$h_t = \begin{cases} \frac{1}{4} & t = 0 \\ 0 & t \text{ even}, \neq 0 \\ -\frac{1}{\pi^2 t^2} & t \text{ odd} \end{cases}$$



Reconstruction

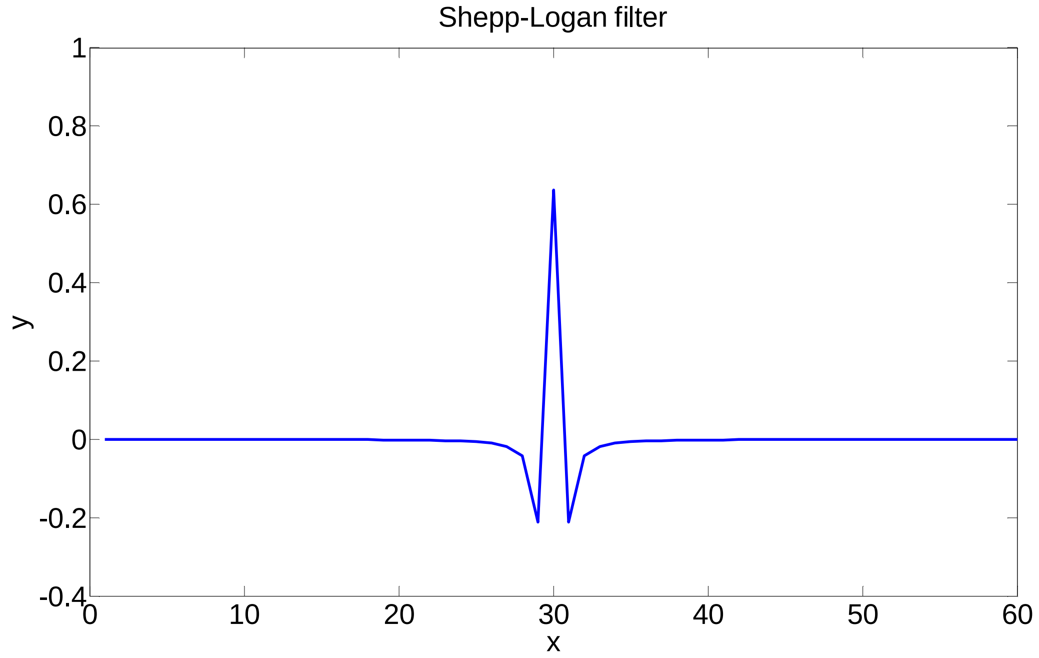
- Task: Implement the discrete spatial version of the Shepp-Logan filter.

```
function [shepp] = SheppLogan(width)
%
%
% ???
%
%
%
end
```

Reconstruction

- Task: Implement the discrete spatial version of the Shepp-Logan filter.

$$h_t = -\frac{2}{\pi^2} \cdot \frac{1}{(4t^2 - 1)}$$



Filtered Backprojection for Parallel Beam

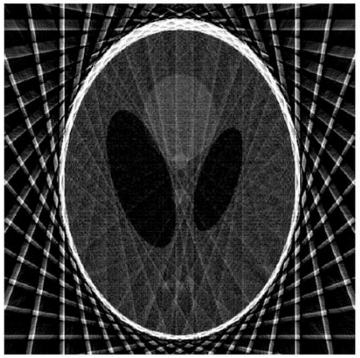
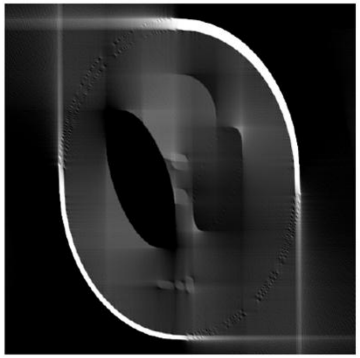
- What is the maximal angle that makes sense to acquire projections at?

Filtered Backprojection for Parallel Beam

- What is the maximal angle that makes sense to acquire projections at?
  - 180 ° – after that, the same data is acquired twice
- Which artefacts appear if you use 110 projections at 1 ° increment?

Filtered Backprojection for Parallel Beam

- What is the maximal angle that makes sense to acquire projections at?
  - 180 ° – after that, the same data is acquired twice
- Which artefacts appear if you use 110 projections at 1 ° increment?
  - View-undersampling artefacts
  - Manifestation in CT: Streaks, “rough” edges, wrong grey values and (most important) missing parts

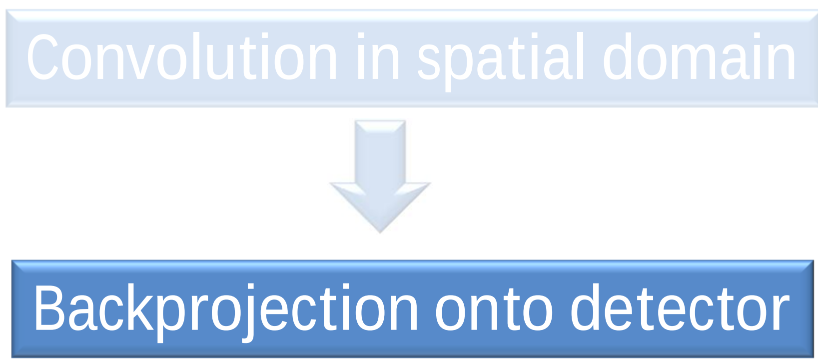




Filtered Backprojection for Parallel Beam

- Which artefacts appear if data gets truncated?
  - Cupping artefacts, bright ring artefacts
  - Wrong grey values

Reconstruction

- Used Filtered Backprojection scheme:



Backprojection

- Two different approaches are common

1. Detector driven: “Smear” detector values over the image.
   - Problem: Interpolation in 2D!
2. Pixel driven: Sample where you expect the outcome!
   - Go over all pixel centers
   - Project center points to the detector
   - Interpolate on the detector and assign to corresponding pixel

- Both approaches are repeated for each projection
- Output is the mean over all results

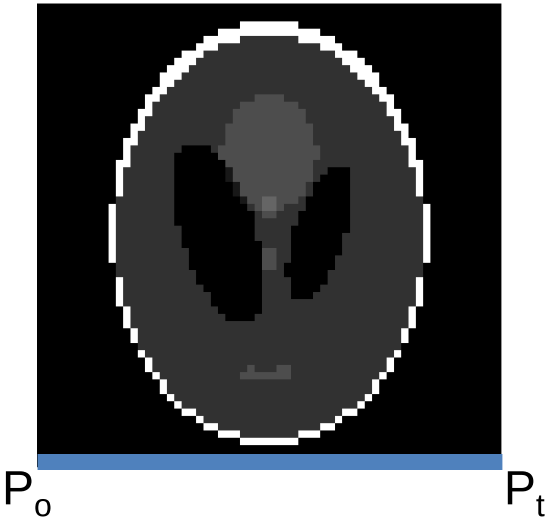
Backprojection

- We rotate the detector border points. The coordinate system's origin is shifted according to the rotation center.

```
Po = [-dimensions(1)/2; -dimensions(2)/2];
Pt = [-dimensions(1)/2; dimensions(2)/2];

R = [cos(rad), -sin(rad);
     sin(rad), cos(rad)];

pPo = (R*Po);
pPt = (R*Pt);
```



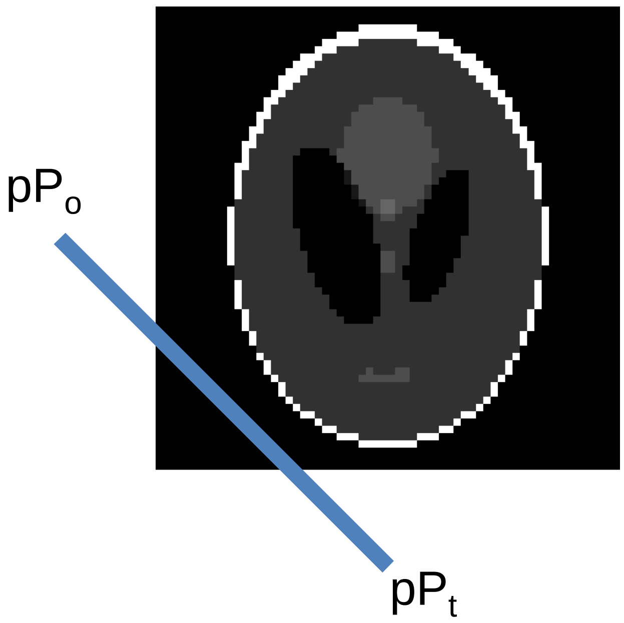
Backprojection

- We rotate the detector border points. The coordinate system's origin is shifted according to the rotation center.

```
Po = [-dimensions(1)/2; -dimensions(2)/2];
Pt = [-dimensions(1)/2; dimensions(2)/2];

R = [cos(rad), -sin(rad);
     sin(rad), cos(rad)];

pPo = (R*Po);
pPt = (R*Pt);
```



Backprojection

- We use the Hesse normal form to calculate the distance of each point to the detector.

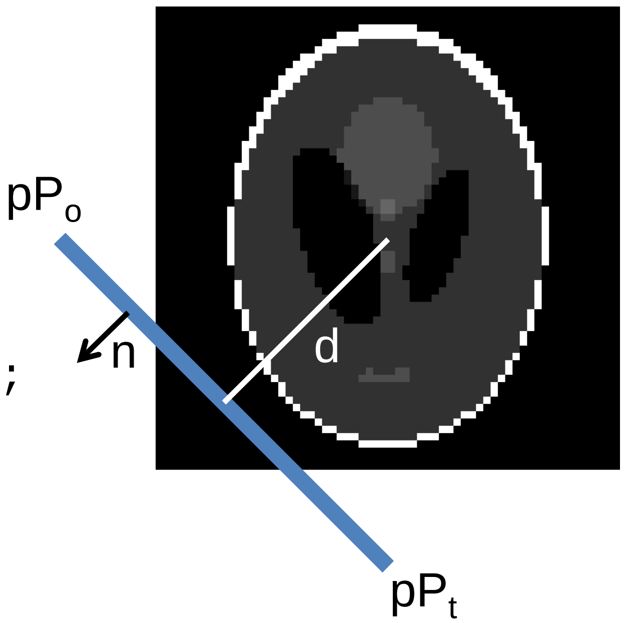
1. Derive the normal form for the detector

$$\boldsymbol{x}^{\boldsymbol{T}}\boldsymbol{n} - d = 0$$

```
dirDet = pPt-pPo;
dirDet = dirDet/norm(dirDet);

normalDet = [-dirDet(2);dirDet(1)];
normalDet = normalDet/norm(normalDet);

d = pPo'*normalDet;
```

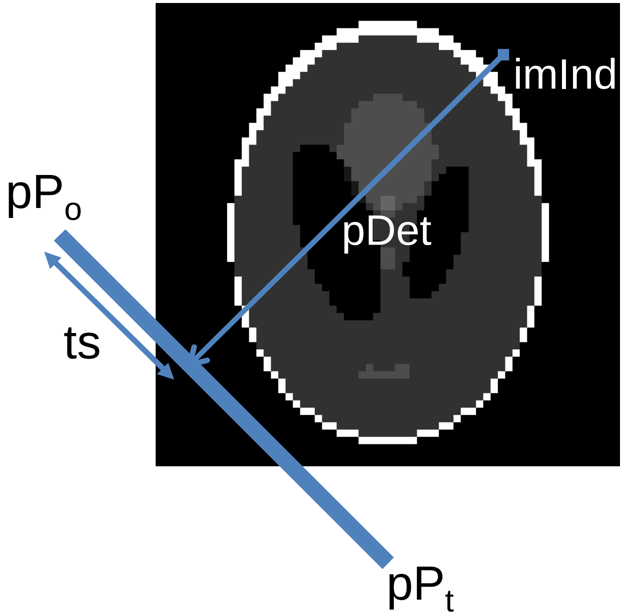


Backprojection

- Projection is done using the detector normal and the distances

```
dd = imInd*normalDet-d*ones(nInd,1);
pDet = imInd-repmat(normalDet',nInd,1).*repmat(dd,1,2);

dis = pDet-repmat(pPo',nInd,1);
ts = sqrt((dis(:,1).*dis(:,1))+
        (dis(:,2).*dis(:,2)));
```

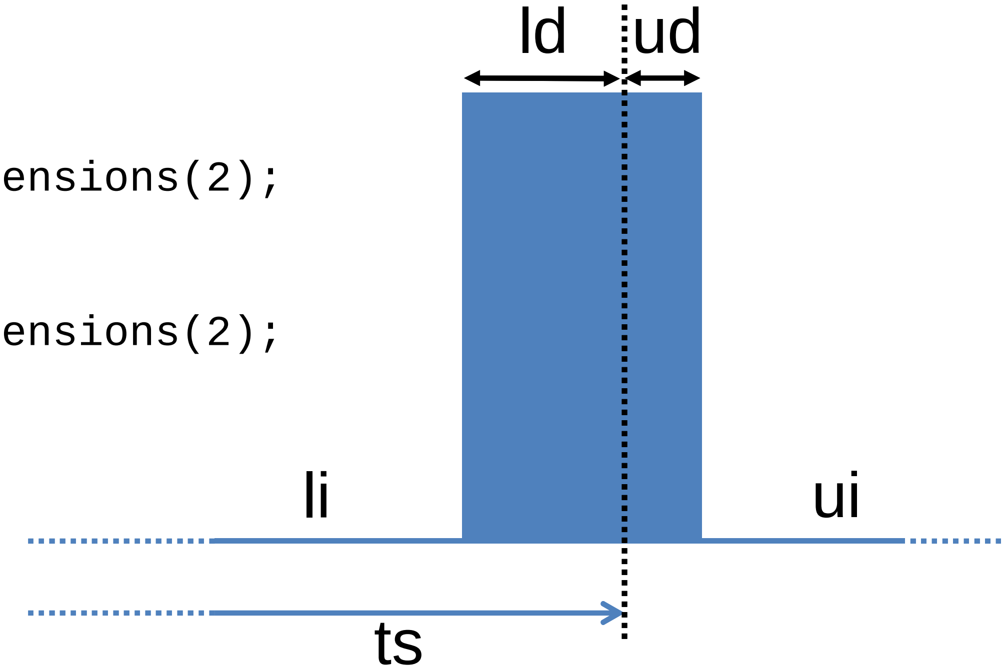


Backprojection

- Points do not necessarily hit a detector cell. Thus, we have to interpolate between the lower cells `li` and the upper cells `ui` using distance weights `ld` and `ud`.

```
li = floor(ts);
li(li<1)=1;
li(li>dimensions(2))=dimensions(2);
ui = li+1;
ui(ui>dimensions(2))=dimensions(2);

ld = abs(ts-li);
ud = abs(ts-ui);
```




```
fbpv = (ones(nInd,1)-ld).*proj(li)'+(ones(nInd,1)-ud).*proj(ui)';
```