# DMIP Summary

## Untertitel

## Art der Arbeit

im Fachgebiet Fachgebiet

vorgelegt von: Christopher Syben

Studienbereich: Studienbereich

## Abstract

Abstract

# Inhaltsverzeichnis

# Abkürzungsverzeichnis

**Abk.** Abkürzung

# Abbildungsverzeichnis

# Tabellenverzeichnis

# 1. Math

## 1.1. Matricies

### Nullspace

The nullspace of a matrix is:

$$\mathbf{A}\vec{x} = \vec{0} \tag{1.1}$$

which means all vectors $\vec{x}$ which fullfill the equation are the nullspace of $\mathbf{A}$.
All Matrices have the trivial nullspace: $\vec{0}$.
If a Matrix have a non-trivial nullspace, than this is a clear idicator that this matrix cannot be inverted because the matrix have a rank deficiency.

### Rank

The rank of a Matrix is the dimension of the collumn-space. The rank tells you the number of linearly independent Collumn-Vectors.

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = (\vec{a_1}, \vec{a_2}) \tag{1.2}$$

### pseudo-inverse

If a Matrix cannot be inverted you can compute the pseudo-inverse, which is as close as possible to the potential inverse.

## 1.2. SVD

The SVD can compute the pseudo-inverse of a Matrix. You can ignore the rank and all the important stuff you have to know wether a matrix can be inverted or not.

- If the Matrix can be inverted then the SVD will give you the inverse of the Matrix !
- If the Matrix cannot be inverted then the SVD will give you the pseudo-inverse of the Matrix !

SVD is a perfect tool for the:

- computation of singular values
- computation of null space
- computation of (pseudo-)inverse
- solution of overdetermined linear equations
- computation of condition numbers
- enforcing rank criterion (numerical rank)

## 1.3. Fourier Transform

### 1.3.1. Symmetry Property of Fourier Transform

There exists a nice property for a real valued discrete signal of length $N$:

$$F(\xi) = \bar{F}(N - \xi) \tag{1.3}$$

If $F(\xi)$ is real-valued Function the Fourier transformed goes from $[0; N-1]$ elements. The Fourier transform at the point $\xi$ is the conjugate at the Point $\bar{F}(N - \xi)$.

### 1.3.2. Dirac's $\delta$-function

The $\delta$-function is defined as:

$$\delta(n) = \begin{cases} 1, & \text{if n=0} \\ 0, & \text{otherwise} \end{cases}$$

You can use Dirac's $\delta$-function to select single values in the frequency domain:

$$F(k) = \hat{F}(s) \cdot \delta(k - s)$$

Only when $s = k$ Dirac's $\delta$-function will be not Zero and you get the value at position $k$.

### 1.3.3. Hilbert Transform

the Hilberttransform $h(\tau)$ is used for convolution and the spatial/frequency domain pair is:

$$h(\tau) = \frac{1}{\pi\tau}$$
$$H(\omega) = -i\ sgn(\omega)$$

## 1.4. Statistics

### 1.4.1. Entropy and Kulback-Leibler Divergence

#### Entropy

The Entropy measures how close your pdf of the intensity values are to the uniform distribution. The higher the Entropy the closer your pdf of your intensity values look like the pdf of the uniform distribution. The Entropy of a discrete random variable X is defined by:

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log p(x_i)$$

If you measures the KL-Divergence from an pdf to the uniform distribution then you get the Entropy formula above.

#### KL - Kulback Leibler Divergence

With the Kulback Leiber Divergence you can measure the similarity of two pdfs. You could use:

$$\int \Big( (p(x) - q(x))^2 \Big) dx$$

but with this equation you have the problem that you run into problems if your random variables have different quantisations.
So instead you should use the KL-Divergence:

$$KL(p, q) = \sum_{i=1}^{n} p(x_i) \log \frac{p(x_i)}{q(x_i)} \tag{1.4}$$

Properties of the KL-Divergence:

- $KL(p,q) \neq KL(q,p)$
- $KL(p,q) \geq 0$
- $KL(p,q) = 0 \Leftrightarrow p = q$
- $KL(p,q) \to 0$ if $p \to q$

### 1.4.2. Independency

When are two random Variables $X, Y$ statistically independent?

- $p(x)$ pdf of $X$
- $q(y)$ pdf of $Y$
- $p(x,y)$ joint pdf
- $p(x) \cdot q(y) = p(x,y)$

They are independent when you can factorize the joint pdf.

### 1.4.3. Regularizer

A closer look to the regularize in the probabilistic context:
where $c$ is the estimated and $x$ the observation:

$$p(c|x) = \frac{p(x,c)}{p(x)} = \frac{p(c) \cdot p(x|c)}{p(x)}$$

for the maximization w.r.t. $c$ $p(x)$ can be ignored, because it does not affect the position of the maximum only the value:

$$\arg\max_c p(c|x) = \arg\max_c p(c)p(x|c) = \arg\max_c \underbrace{\log p(c)}_{\text{regularizer}} + \underbrace{\log p(x|c)}_{\text{data term}}$$

where the data Term $p(x|c)$ contains the information, the observation and $p(c)$ only depends on the class that we considering, which is prior or the regularize.
The regularize only holds information on the classes we want to estimate. So a regularize is just use the objective function (data Term) and adds another term to the objective functions which depends not on the observation only on the parameters ! (is also called prior knowledge because you don't look at the measurements) **take of using prior knowledge in medicine, because patients are usually persons which are not belong to the average**

### 1.4.4. Marginalization - Hidden Random Variables

If you have a PDF $p(X, Y, Z)$ then the following counts:

$$\int\int\int p(X, Y, Z)dXdYdZ = 1 \tag{1.5}$$

to get the probability of observing $X$ without knowing some thing about $Y$ and $Z$. You can integrate over $Y$ and $Z$ to get $p(X)$:

$$\int\int p(X, Y, Z)dYdZ = p(X) \tag{1.6}$$

in other words, you can eliminate random Variables by marginalization.

## 1.5. Homogeneous Coordinates

A two-dimensional point in Cartesian coordinates $p = (x, y)^T \in \mathbb{R}^2$ is represented by $(wx, wy, w)^T \in \mathbb{P}^2$ in homogeneous coordinates, where $w \in \mathbb{R}$ is an arbitrary real valued constant.
A Vector in homogeneous coordinates can be transformed back to Cartesian coordinates by dividing the components with the last component ($\neq 0$).
A 2-D point $(x, y)^T$ in Cartesian coordinates corresponds to a line in 3-D:

$$\begin{pmatrix} x \\ y \end{pmatrix} \rightarrow w \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \qquad \text{where } w \in \mathbb{R} \tag{1.7}$$

There exists an infinite number of homogeneous points that correspond to one and the same 2-D point!

### 1.5.1. Rotation and Translation

normally you have to multiply the rotation matrix from the left side and add a 3D translation vector $t$:

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \mathbf{R} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + t \tag{1.8}$$

if you use homogeneous coordinates you can incorporate the translation:

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} R_{11} & R_{12} & R_{13} & t_x \\ R_{21} & R_{22} & R_{23} & t_y \\ R_{31} & R_{32} & R_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{1.9}$$

# 2. PreProcessing

## 2.1. Disortion & Interpolation

## 2.2. Detectors

The Image quality is defined by three major components.

- Noise
- spatial Resolution
- contrast resolution

### 2.2.1. DQE - Measurement

The detective quantum efficiency (DQE) is a important measurement of a Detector.
$\mathbf{DQE} = \frac{\mathrm{SNR}^2(f)_{out}}{\mathrm{SNR}^2(f)_{in}}$

## 2.3. Defect Pixel Interpolation

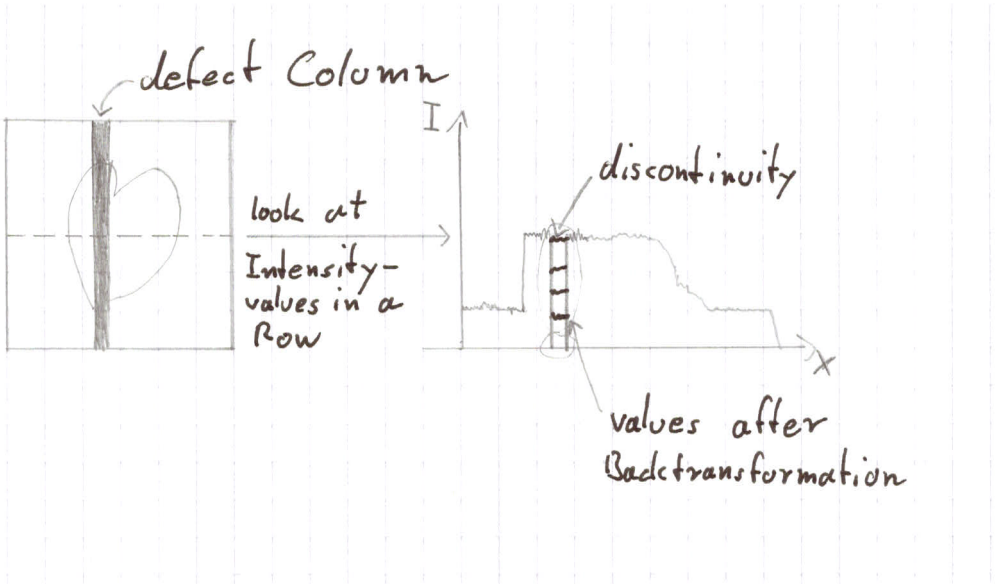
In Interpolation you hit the sampling points exactly, not like regression where you fit a line with the smallest error. Or like extrapolation, where you try to find a sampling point outside your interval of sampling Points.

### 2.3.1. Defect Pixel

The mathematical model for defect generation is just the multiplication of the original image with a defect mask. Let $f_{i,j}$ denote the intensity value at

$$w_{i,j} = \begin{cases} 1, & \text{if pixel is defect} \\ 0, & \text{otherwise} \end{cases}$$

### Defect Pixel Interpolation by band limitation

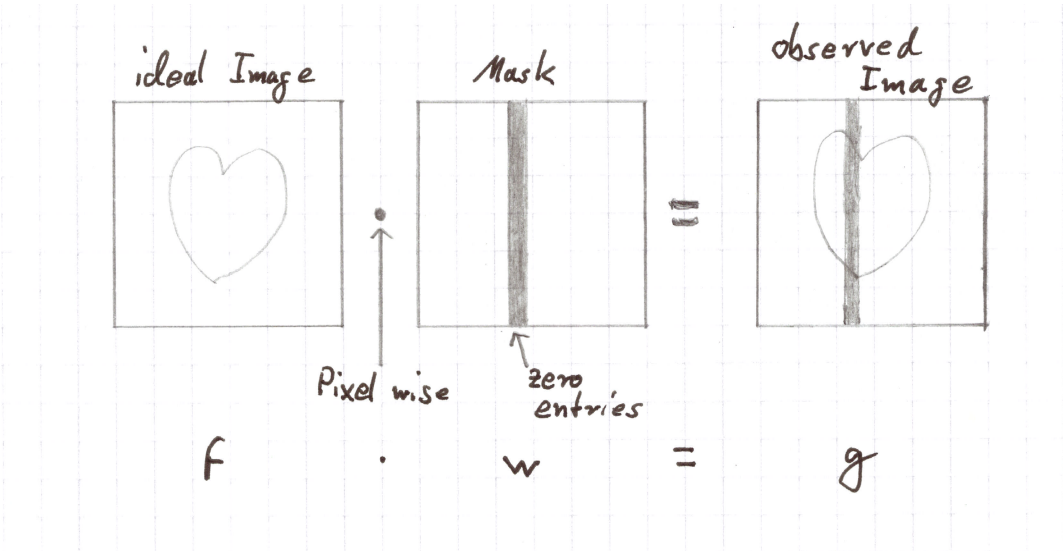



the discontinuity leads to a Signal which have high frequencies and is not band-limited any more ! So the Fourier transform run into the problem that the Fourier transform will not achieve zero values for a certain threshold $\xi$.
So take the Fourier transform of the signal with the defect and cut off all the frequencies that are larger than the Band-limitation $b$ of the Detector (this is given), back transform the signal and fill in the new values at the defect column.
This Values are not zero but you still have a huge difference. Repeat this proceeder iteratively, at the end you will get a signal which do not violate the band-limitation of the Signal.
Which is nothing else than a low pass filter but it is very important that you do not filter the whole image; only at the defect positions.

| |
|---|
| compute FT of input signal $g(n)$ |
| set $G(\xi) = 0$ for $\xi > B_u$ or $\xi < B_l$, |
| compute inverse FT of corrected $G(\xi)$ |
| Replace defect samples in $g(n)$ by values of inverse FT |
| UNTIL changes are below a threshold |

- Band limitation must be known
- its computationally expensive, because each iteration requires twice the Fourier transform.
- If the defect is at the border of the observed interval, its a case of extrapolation the results aren't that good.

### Frequency Domain Defect Pixel Interpolation




We can interpret our defect Image $g$ as a product of the ideal Image $f$ and a defect mask $w$. This cant be inverted because the mask contains zero entries (defect Pixels).

$$f(n) \cdot w(n) = g(n)$$
$$F(\xi) * W(\xi) = G(\xi)$$

But we can look in the frequency Domain, where we can compute a deconvolution! All three, the ideal, the mask and the defect Image the Fourier transform satisfies the symmetry property (all three are real valued):

$$F(\xi) = \bar{F}(N - \xi)$$
$$W(\xi) = \bar{W}(N - \xi)$$
$$G(\xi) = \bar{G}(N - \xi)$$

Instead of looking at the whole Fourier transform we only look on the symmetric pairs $G(s)$ and $G(N-s)$ from the corrupted Image and $F(s)$ and $F(N-s)$ from the ideal Image. With these Pairs we can rewrite the Fourier transform using Dirac's $\delta$-function:

$$F(k) = \hat{F}(s)\delta(k - s) + \hat{F}(N - s)\delta(k - N + s)$$

So the Fourier transformation just consists of the too values at $s$ and $N-s$ selected with the $\delta$-function.
You can put that into the convolution; this gets reduced to two elements:

$$G(s) = \underbrace{\frac{1}{N}}_{\text{scaling}} \left(\hat{F}(s)W(0) + \hat{\bar{F}}(s)W(2s)\right)$$

The conjugate complex of this is:

$$\bar{G(s)} = \frac{1}{N}\left(\hat{\bar{F}}(s)\bar{W}(0) + \hat{F}(s)\bar{W}(2s)\right)$$

The $\hat{\bar{F}}(s)$ can be replaced by the original function $F(N-s)$.
Then we have two linear equations($G(s)$ and $\bar{G}(s)$) with two unknowns $F(s)$ and $F(N-s)$ so we can solve it.

$$\hat{F}(s) = N\frac{G(s)\bar{W}(0) - \bar{G}(s)W(2s)}{|W(0)|^2 - |W(2s)|^2}$$

where $|.|$ is the absolute value of the complex number.
This is not for the whole Fourier transform, its only for one pair. We can solve this equations for different Pairs and get estimates for the whole Fourier transform; transform $F$ back and compare it to our measured Image.
So we can iterate over these different pairs and minimize the error between the measured Image and the back transformed Image $f$ multiplied with our mask Image:

$$\Delta_\epsilon = \frac{1}{N}\sum_{n=0}^{N-1}(g(s) - w(n)\cdot f(n))^2 \tag{2.1}$$

| | | |
|---|---|---|
| compute FT of input signal $g(n)$ | | |
| Initialize $\hat{F}^0(k) = 0$, $G^0(k) = G(k)$, $i = 1$ | | |
| | Randomly select a line pair $s \neq 0$ $G^{i-1}(s) = G^{i-1}(N-s)$ | |
| | Estimate $\hat{F}^i(s)$, $\hat{F}^i(N-s)$ using (6) | |
| | Update spectrum $\hat{F}^i(k)$, compute error $\Delta_\epsilon$ using (8) | |
| | IF | error $\Delta_\epsilon$ above a threshold |
| | THEN Compute error spectrum $G^i(k)$, increment $i$ | |
| UNTIL error are below a threshold | | |
| Compute inverse FT of $\hat{F}^i(k)$ | | |

## 2.4. MRT

The magnetic Field in space should be different in each spatial point. The MR-Scanner needs a gradient System which generates this special magnetic field and a RF-System. The RF-System contains a transmission and a receiver coil. The transmitter coil generates a rotating magnetic field for the excitation of a spin system(the nucleus have beside mass and charge their spin as a basic property), and the receiver coil converts magnetic charges in electrical system. So the transmitter causes changes in the System and the receiver coil measures this changes and convert this into intensity values.

**Pros:**

- patient care, no ionising Rays
- high spatial resolution($50\mu m$)
- excellent contrast resolution (discrimination of soft tissues)
- ...

**Cons:**

- inhomogeneities caused by radio frequency coil.
- intensity inhomogeneities produce spatial changes in tissue statistics. That means in an MR image the intensity value of water can be 5 but also 5000. Different intensities at different points in the Image may characterize the same molecular structure.
- inhomogeneities can change with different acquisition parameters, from patient to patient and from slice to slice

### 2.4.1. Intensity Inhomogeneities (IIH) in MRI

Their are different Reasons for inhomogeneities in MRI:

- non-uniform radio-frequency
- inhomogeneity of the static main field
- patient motion

There exists different mathematical models to describe the IIH:

- **Low-Frequency model**: It is assumed that IIH is caused by low-frequency components; the IIH map can be recovered by low-pass filtering

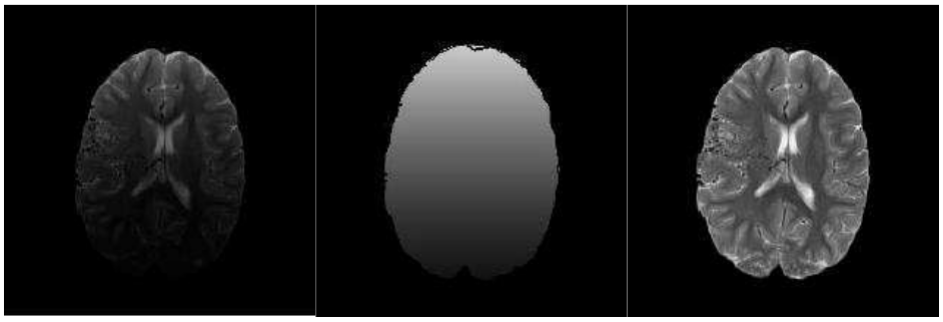


Abbildung 2.1.: Original MR Image(left),gain field (middle), restored image (right)

- **Hypersurface model**:IIH map is represented by a smooth (low-frequency) parametric function; the IIH map can be recovered by least-square-fitting (regression)
- **Statistical model**:IIH map is represented by a stochastic process; the IIH map can be recovered dependent on the selected statistical model by parametric or non-parametric statistical estimation

This multiplicative model is now a days the accepted model, earlier used additive models are not close enough the reality:

$$g_{ij} = f_{ij} \cdot b_{ij} + n_{ij} \tag{2.2}$$

where $g_{ij}$ is the observed Image, $f_{ij}$ the ideal Image and $b_{ij}$ is the gain field and $n_{ij}$ is the noise mostly set as a Gaussian noise. Correction methods mostly will applied to the product of $f$ and $b$, the noise should be removed before. You can use a homomorphism and apply the logarithm to the equation above to replace the product with a summation.
**Hint:** if you use the logarithm you have to be aware of that the resulting noise is no Gaussian noise any more !

### High pass filtering

The main Idea is to apply a high pass filter on the Image, because the bias field is generated by low frequencies. For a faster filtering transform the measured Image into the frequency domain using the Fourier transform and then multiply it with a high-pass filter Kernel $H$:

$$H_{k,l} = 1 - \beta \cdot e^{-\frac{k^2+l^2}{2\sigma^2}} \tag{2.3}$$

where $\beta$ is a scaling factor that ensure that $H_{k,l}$ for all $k, l = 0, .., N-1$ and $\sigma^2$ is closely related to the bandwidth of the filter-kernel.
So you get your corrected Image $f$ with:

$$f = \mathrm{FT}^{-1}\left\{G_{k,l} \cdot H_{k,l}\right\} \tag{2.4}$$

### Homomorphic Filtering

This type of filtering assume IIH is:

- an artefact with low frequencies
- the anatomic structure contribute to the high frequencies in the image

The same assumptions like in the filtering before, but this time we go another way. The main idea is to do a low-pass filtering to get the bias field and subtract this from the measured Image.
Homomorphic filtering is applied to log-transformed images:

- low-pass filterung of the log-transformed image (LFP denotes a low pass filter):

$$[h_{i,j}] = \mathrm{LPF}([\log g_{i,j}]) \tag{2.5}$$

- IIH corrected log-transformed image results from the difference:

$$[\log f_{i,j}] = [\log g_{i,j}] - [h_{i,j}] + \mu \tag{2.6}$$

  where $\mu$ assures that IIH correction is mean preserving. So you add here a mean-value, to make sure that you have a mean normalisation, you make sure that the final mean intensity value is in a certain range.

### Homomorphic Unsharp Masking

Is apply a mean normalization. The idea is, that when you have a Image of the brain, for example, then you should have in patches of the Image the same mean-value, but as shown in Figure 2.1 that not true:

$$f_{i,j} = g_{i,j} \cdot \frac{\mu}{\mu_{i,j}} \tag{2.7}$$

where $\mu$ is the global mean of the Image and $\mu_{i,j}$ is the mean of the neighbourhood at pixel $i, j$.
**Hint:** take in consideration that this method implies that the local mean is the same

as the gloab mean, which is only true for some cases. A full body scan for example would violate this assumption!

#### Polynomial filtering

#### KL Divergence Minimization

It allows the incorporation of prior knowledge

#### Fuzzy C-means Clustering

The main idea is to use a probabilistic k-means algorithm. So you calculate the distance to the Clusters but incorporate the probability that the feature belongs to a Cluster.
The fuzzy C-means objective function for partitioning the observation into $N_c$ clusters and allows one data point belong to more than one class:

$$J(x_1, x_2, ...., x_n) = \sum_{i=1}^{N_c} \sum_{k=1}^{n} a_{i,k}^d ||x_k - c_i||^2 \tag{2.8}$$

- Prior: number of tissue classes
- $c_1, c_2, ..., c_{N_c}$ are the prototypes of the clusters
- $x_1, x_2, ..., x_n$ are the data points, in ours case the logarithms of ideal intensities

The objective function has to be minimized with respect to $c_i, a_{i,k}$. The optimization needs the constraint that the probability that a feature belongs to a cluster sum up to one:

$$\text{minimize} \sum_{i=1}^{N_c} \sum_{k=1}^{n} a_{i,k}^d ||x_k - c_i||^2 \tag{2.9}$$

$$\text{subject to} \sum_{i=1}^{N_c} a_{i,k} = 1 \tag{2.10}$$

This can be solved with the Lagrange multiplier method.
There are few drawbacks we have to consider and incorporate into the Method, so we can apply it on the bias field correction:

- the current objective function with the probabilistic assignment of data points to classes does not consider dependencies of neighboring data points (intuition: neighboring data points most probably belong to the same class)

- probabilistic approach required mutually independent intensities The question now is, how can we incorporate dependencies of neighboring data points?

A solution for the first drawback is to incorporate a regularize (more Information about regularizer: 1.4.3):

$$J(x_1, x_2, ...., x_n) = \underbrace{\sum_{i=1}^{N_c}\sum_{k=1}^{n} a_{i,k}^d ||x_k - c_i||^2}_{\text{data Term}} + \underbrace{\sum_{i=1}^{N_c}\sum_{k=1}^{n} \frac{\lambda}{\#\mathcal{N}_k} a_{i,k}^d \sum_{x_r \in \mathcal{N}_k} ||x_r - c_i||^2}_{\substack{\text{data term whith prior;} \\ \text{incorporates neighbourhood}}} \tag{2.11}$$

The neighbourhood is considered in the following way: we can say, we sum over all the pixels($x_r$) in a local neighbourhood ($\mathcal{N}_k$) and we compute the distance between the assigned class $c_r$ to the currently considered class/cluster $c_i$, then we weight them also by probabilities ($a_{i,k}^d$) and scale them by the size of the neighbourhood ($\frac{\lambda}{\#\mathcal{N}_k}$).
Next step is to incorporate the bias field:
We replace the logarithm of ideal intensity value $x_k$ using $x_k = y_k - \beta_k$ and minimize the optimization problem with respect to the probability $a$, the cluster $c_i$ and the bias $\beta$:

$$\{\hat{\mathbf{A}}, \hat{c}_i, \hat{\beta}_i\} = \underset{\mathbf{A}, c_i, \beta_i}{\arg\min} \sum_{i=1}^{N_c}\sum_{k=1}^{n} a_{i,k}^d ||x_k - c_i||^2 + \sum_{i=1}^{N_c}\sum_{k=1}^{n} \frac{\lambda}{\#\mathcal{N}_k} a_{i,k}^d \sum_{x_r \in \mathcal{N}_k} ||x_r - c_i||^2$$

$$\text{subject to: } \sum_{i=1}^{N_c} a_{i,k} = 1 \text{ for all } k = 1, 2, ..., n$$

with this we try to compute the bias field $\beta_k$ of pixel $x_k$.
This can be done with setting up the optimization with a Lagrange multiplier and at the end of the day we get close form solutions for all three parameters:

$$J_R = \sum_{i=1}^{N_c}\sum_{k=1}^{n}\left(a_{i,k}^d D_{i,k} + \frac{\lambda}{\#\mathcal{N}_k} a_{i,k}^d E_{i,k}\right) + \sum_{k=1}^{n} \eta_k (1 - \sum_{j=1}^{N_c} a_{j,k}) \tag{2.12}$$

$$\text{where } D_{i,k} = \|y_k - \beta_k - c_i\|^2 \tag{2.13}$$

$$E_{i,k} = \sum_{(y_r - \beta_r) \in (N)_k} \|y_r - \beta_r - c_i\|^2 \tag{2.14}$$

the computation of the zero crossings of the gradient results in the following estimator for the partition matrix:

$$\hat{a}_{i,k} = \frac{1}{\sum\limits_{j=1}^{N_c} \left( \frac{\#\mathcal{N}_k D_{i,k} + \lambda E_{i,k}}{\#\mathcal{N}_k D_{j,k} + \lambda E_{j,k}} \right)^{\frac{1}{d-1}}} \tag{2.15}$$

..leads to Cluster Prototype Update:

$$\hat{c}_i = \frac{\sum\limits_{k=1}^{n} a_{i,k}^d \left( (y_k - \beta_k) + \frac{\lambda}{\#\mathcal{N}_k} \sum\limits_{y_r \in \mathcal{N}_k} (y_r - \beta_r) \right)}{(1+\lambda) \sum\limits_{k=1}^{n} a_{i,k}^d} \tag{2.16}$$

and to the Bias Field Estimator:

$$\hat{\beta}_k = y_k - \frac{\sum\limits_{i=1}^{N_c} a_{i,k}^d \cdot c_i}{\sum\limits_{i=1}^{N_c} a_{i,k}^d} \tag{2.17}$$

#### Probabilistic Correction of Bias Fields

Like in the fuzzy C-means-clustering this idea combines the bias field correction with an simultaneously image segmentation step.
But in this approach we want to compute the unbiased image where tissue classes of pixels (i.e. segmentation) and the bias field itself are unknown! So we don't know which pixel belongs to which class and at the same time we do not know which bias is present in this particular pixel.
The core Idea of this approach is to use the trick of Marginalization 1.4.4: We will characterize our Image by assuming the tissue classes are known, the bias field is known and we have an given observation and at the end we compute a probability for the given Image by integrate out the bias and integrating out the tissue classes with this marginalisation trick.
To use the **E**xpectation and **M**aximization Algorithm have to fit this idea into the following framework:

- observable random measurement: bias logarithmic intensity value
- hidden random measurement: tissue class for each pixel
- parameter estimation problem: computation of the bias field

Instead of saying the bias field is an unknown parameter, we can also say the bias field is an hidden random variable, depends on how we want to Setup the estimation

problem. This is an incomplete data estimation problem.
The used probabilistic model consists of the following components:

- logarithmic intensity Values $x_{i,j} = \log g_{i,j}$ belonging to tissue classes $\Gamma$ are normally distributed:

$$p(x_{i,j}|\Gamma;\beta_{i,j}) = \mathcal{N}(x_{i,j};\mu_\Gamma + \beta_{i,j}, \Sigma_\Gamma) \tag{2.18}$$

$$= \frac{1}{\sqrt{|2\pi\Sigma_\Gamma|}} e^{-\frac{1}{2}(x_{i,j}-\mu_\Gamma-\beta_{i,j})^T\Sigma_\Gamma^{-1}(x_{i,j}-\mu_\Gamma-\beta_{i,j})} \tag{2.19}$$

  where

  - $x_{i,j}$: observed log intensity
  - $\Sigma_\Gamma$: covariance matrix of tissue class $\Gamma$
  - $\beta_{i,j}$: bias at point (i,j)
  - $\mu_\Sigma$ mean log intensity of tissue class $\Sigma$

- prior probability of tissue class,i.e. without considering any observation (you get this from an anatomic Atlas):

$$p(\Gamma), \text{for} \qquad \Gamma = 1, 2, ..., N \tag{2.20}$$

- prior density of bias field:

$$p(\beta_{i,j} = \mathcal{N}(\beta_{i,j}; 0, \Sigma_\beta)) \tag{2.21}$$

  that means the parameters we want to estimate, they also underlay a certain PDF. For example if you look at the low frequencies changes in MR Images and if you see its brighter on one side and darker on the other side and fit in a PDF that characterizes this behaviour.
  We Assume the bias field is Normally-Distributed (it works !)

- elimination of unknown tissues class $\Gamma$ by marginalization:

$$p(x_{i,j};\beta_{i,j}) = \sum_{\Gamma=1}^{N} p(\Gamma)p(x_{i,j}|\Gamma;\beta_{i,j}) \tag{2.22}$$

  which means we built up a probabilistic model where we assume we know the class assignment $p(\Gamma)p(x_{i,j}|\Gamma;\beta_{i,j})$. But we don't know the class assignment, so we use marginalization to get rid of the class assignment. For this we multiply our model with the prior knowledge that a certain tissue class occurs and sum up over all tissue classes.

Due to the fact that $p(\beta_{i,j})$ is assumed to be Gaussian, the bias field is now considered as a random variable. We do estimate the bias field $\beta = [\beta_{i,j}]$ by the maximization of posteriors given the logarithmic image $\mathbf{x} = [x_{i,j}]$:

$$\hat{\beta} = \arg\max_{\beta} p(\beta|\mathbf{x}) = \arg\max_{\beta}(\log p(\beta) + \log p(\mathbf{x}|\beta)) \tag{2.23}$$

with the Assumption that the Intensities are **mutually independent** we can compute the joint density for the whole Image by multiply over all pixels the probability that we observe this intensity $x_{i,j}$ given the bias $\beta_{i,j}$ (commonly used simplification !):

$$p(\mathbf{x}|\beta) = \prod_{i,j} p(x_{i,j}|\beta_{i,j}) = \prod_{i,j} \sum_{\Gamma=1}^{N} p(\Gamma) p(x_{i,j}|\Gamma, \beta_{i,j}) \tag{2.24}$$

The estimation of the bias field can be done iteratively, but the current model includes hidden variables. In total, we have to estimate the following parameters using the EM-Algorithm:

- bias field $\beta$
- covariance $\Sigma_\Gamma$
- priors $p(\Gamma)$
- means $\mu_\Gamma$ and covariances $\sigma_\Gamma$

Once all the parameters are estimated, the segmentation result is required. The final tissue class of each pixel can be estimated by the maximization of the posteriors:

$$\hat{\Gamma} = \arg\max_{\Gamma} p(\Gamma|\vec{x}_{i,j}; \beta_{i,j}) \tag{2.25}$$

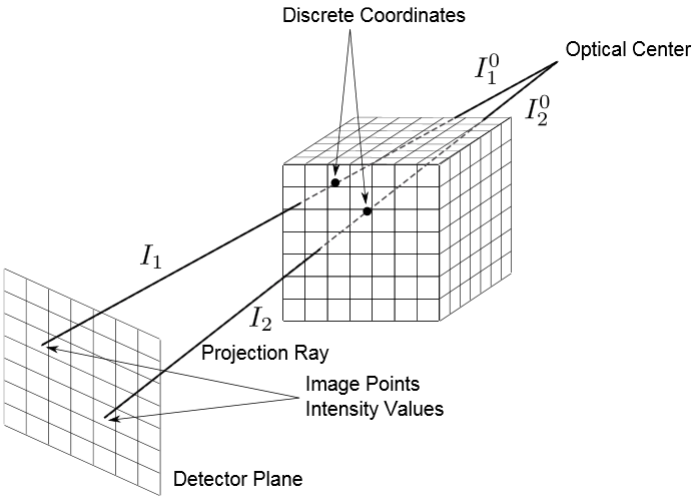
$$= \arg\max_{\Gamma}(\log p(\Gamma) + \log p(x_{i,j}|\Gamma; \beta_{i,j})) \tag{2.26}$$

**Remark:** in practice the bias field is not estimated for all components $\beta_{i,j}$. but usually approximated by a parametric function:

$$\beta_{i,j} = \sum_{k=0}^{M} \theta_k \phi_k(i,j) \tag{2.27}$$

where we have $\theta_k \in \mathbb{R}$ and $\phi_k$ are proper base functions. The bias field estimation thus reduces to the computation of the $\theta_k$'s. The Parameters can be estimated with the EM-Algorithm

# 3. Projection and Homogeneous Coordinates




We have to find a representation of the Projection Ray's to handle the reconstruction Problem.

### 3.0.2. Projections

There existing different Projection Models:

- Orthographic projection

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \rightarrow \begin{pmatrix} x \\ y \end{pmatrix} \tag{3.1}$$

  which is a linear mapping and can be written in homogeneous coordinates.

- Weak perspective projection

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \rightarrow \begin{pmatrix} k \cdot x \\ k \cdot y \end{pmatrix} \tag{3.2}$$

  which is a linear mapping and can be written in homogeneous coordinates.

- Perspective projection

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \rightarrow \begin{pmatrix} f \cdot x/z \\ f \cdot y/f \end{pmatrix} \tag{3.3}$$

  where $f$ is the distance to the image plane to origin.
  This is a **non-linear mapping** of the points !

The projection model of X-Ray systems can be approximated by perspective projection. A downside is that this model is not a linear one. But we can fix this with Homogeneous Coordinates.
We will now formulate projections from 3D to 2D using Homogeneous coordinates:

- Orthographic Projection in homogeneous coordinates id defined by:

$$\tilde{\mathbf{P}} = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \qquad \tilde{\mathbf{P}}' = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tilde{\mathbf{P}} \tag{3.4}$$

  this mapping from $\mathbb{P}^3 \rightarrow \mathbb{P}^2$ can be simply written in matrix form as:

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} x \\ y \\ y \end{pmatrix} \Rightarrow \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \boxed{0} & \boxed{1} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{3.5}$$

- Perspective Projection in homogeneous coordinates is defined by:

$$\tilde{\mathbf{P}} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} f \cdot \frac{x}{z} \\ f \cdot \frac{y}{z} \end{pmatrix} \rightarrow \begin{pmatrix} fx \\ fy \\ z \end{pmatrix} \qquad \tilde{\mathbf{P}}' = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tilde{\mathbf{P}} \tag{3.6}$$

  where $f$ is the focal length!
  this mapping from $\mathbb{P}^3 \rightarrow \mathbb{P}^2$ can be simply written in matrix form as:

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} x \\ y \\ z \end{pmatrix} \Rightarrow \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \boxed{1} & \boxed{0} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{3.7}$$

Both orthographic Projection and Perspective Projection can be written in terms of a linear Mapping, both matrices differ only in the red marked elements.

## 3.1. Extrinsic Camera Parameters

Extrinsic Parameters are the translation and rotation which are applied to the Camera in the 3D-World. There existing 6 extrinsic Parameters: 3 Translation $(x, y, z)^T$ and 3 rotations.
To get an affine linear mapping we can use homogeneous coordinates (1.5.1). Then it is just a matrix multiplication from the left side:

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \mathbf{D} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{3.8}$$

where $\mathbf{D}$ include the rotation and the translation and is a $4 \times 4$ Matrix for the extrinsic Parameters.

## 3.2. Intrinsic Camera Parameters

The intrinsic Camera parameters have 5 degrees of Freedom:

- the angle of the CCD-Chip (1 degrees of Freedom)
- the scaling in X-Y Direction, because we have non-rectangular Pixels (in this scaling the focal length included) (2 degrees of Freedom)
- the ideal coordinate System, which is used for the discussion, has an offset $(u, v)^T$ regarding to the coordinate System of the CCD-Chip(2 degrees of Freedom). The origin of the ideal coordinate system is defined by the optical axes (the X-Ray which hits the image plane orthogonal), this is the offset.

The intrinsic parameters do not change if the camera moves. the mapping for the

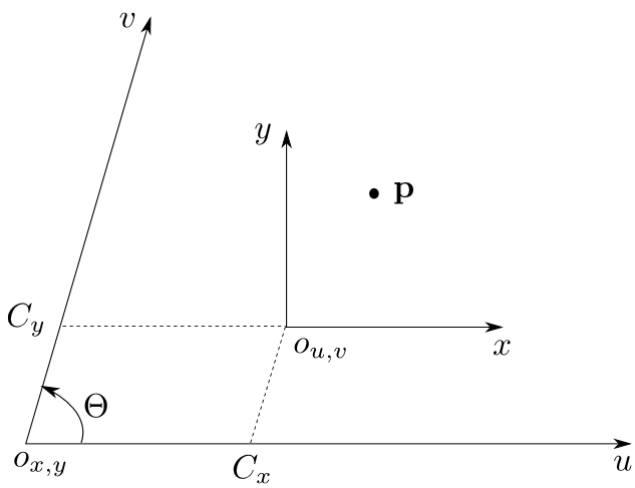



Abbildung 3.1.: $(u; v)$ detector and $(x; y)$ image coordinate system

figure above we get by looking at the base-vectors and their difference:

$$\vec{x} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \curvearrowright \begin{pmatrix} \frac{1}{k_x} \\ 0 \end{pmatrix} \qquad \vec{y} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \curvearrowright \begin{pmatrix} \frac{1}{k_y} \cos\theta \\ \frac{1}{k_y} \sin\theta \end{pmatrix} \tag{3.9}$$

the required transform from $(x, y)$ to the $(u, v)$ coordinate system is given by the inverse of the matrix:

$$\mathbf{T} = \begin{pmatrix} \frac{1}{k_x} & \frac{1}{k_y} \cos\theta \\ 0 & \frac{1}{k_y} \sin\theta \end{pmatrix}^{-1} = \begin{pmatrix} k_x & -k_x \frac{\cos\theta}{\sin\theta} \\ 0 & \frac{k_y}{\sin\theta} \end{pmatrix} \tag{3.10}$$

We can combine $\mathbf{T}$ and the displacement Vector $(c_x, c_y)^T$ using homogeneous coordinates to get the intrinsic camera parameter matrix $\mathbf{K}$:

$$\mathbf{K} = \begin{pmatrix} T_{11} & T_{12} & -c_x \\ T_{21} & T_{22} & -c_y \\ 0 & 0 & 1 \end{pmatrix} \tag{3.11}$$

## 3.3. Complete Projection

Now we can put all Matrices we have thought about in one step and get the Projection matrix $\mathbf{P}$:

$$\rho\tilde{q} = \mathbf{P} \cdot \tilde{p} = \mathbf{K} \cdot \mathbf{P}_{\text{proj}} \cdot \mathbf{D} \cdot \tilde{p} \tag{3.12}$$

## 3.4. Calibration

We use a Calibration Pattern with Points where we know the coordinates:

$$x_i = \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix}, \begin{pmatrix} x_2 \\ y_2 \\ z_2 \end{pmatrix}, ..., \begin{pmatrix} x_n \\ y_n \\ z_n \end{pmatrix} \in \mathbb{R}^3 \tag{3.13}$$

we observe with our X-ray system a set of 2D-Points:

$$\begin{pmatrix} u_1 \\ v_1 \end{pmatrix}, \begin{pmatrix} u_2 \\ v_2 \end{pmatrix}, ..., \begin{pmatrix} u_n \\ v_n \end{pmatrix} \in \mathbb{R}^2 \tag{3.14}$$

the Projection Matrix we want to estimate have 12 entries, whereby we are in homogeneous coordinate so our solution is unique up to a scaling (we lose here one degree freedom). So we have eleven unknowns to estimate. In other words we need eleven equations to solve this, and each Point on the Calibration Pattern gives us two equations, which means we need at least 5.5 Points to solve this estimation Problem.

with our Projection Matrix $\mathbf{P}$ we know the mapping between these points, up to scaling $w_i$:

$$\forall i : \begin{pmatrix} w_i \cdot u_i \\ w_i \cdot v_i \\ w_i \end{pmatrix} \tilde{=} \mathbf{P} \begin{pmatrix} x_i \\ y_i \\ z_i \\ 1 \end{pmatrix} \Leftrightarrow \begin{pmatrix} r_1^T \\ r_2^T \\ r_3^T \end{pmatrix} \begin{pmatrix} x_i \\ y_i \\ z_i \\ 1 \end{pmatrix} = \begin{pmatrix} \vec{r}_1^T \vec{x}_i \\ \vec{r}_2^T \vec{x}_i \\ \vec{r}_3^T \vec{x}_i \end{pmatrix} \tilde{=} \begin{pmatrix} u_i \\ v_i \\ 1 \end{pmatrix} \tag{3.15}$$

**Hint:** there is a change from the $x$ component to the complete vector $\vec{x}_i$, take care of this !

This formulation, where we split up our Projection matrix into the rows $\vec{r}_1^T, \vec{r}_2^T$ and $\vec{r}_3^T$, leads to:

$$u_i = \frac{\vec{r}_1^T \vec{x}_i}{\vec{r}_3^T \vec{x}_i} \qquad v_i = \frac{\vec{r}_2^T \vec{x}_i}{\vec{r}_3^T \vec{x}_i} \tag{3.16}$$

to get the estimation of the rows, we can set up a least square estimator:

$$u_i - \frac{\vec{r}_1^T \vec{x}_i}{\vec{r}_3^T \vec{x}_i} \stackrel{!}{=} 0 \qquad v_i - \frac{\vec{r}_2^T \vec{x}_i}{\vec{r}_3^T \vec{x}_i} \stackrel{!}{=} 0 \tag{3.17}$$

because of the ratio, this is non-linear, but we can fix this by multiplying with the denominator:

$$u_i \cdot \vec{r}_3^T \vec{x}_i - \vec{r}_1^T \vec{x}_i = 0 \tag{3.18}$$

$$v_i \cdot \vec{r}_3^T \vec{x}_i - \vec{r}_2^T \vec{x}_i = 0 \tag{3.19}$$

and this is now linear in the components of the projection matrix and we can finally set up our least square estimator:

$$\hat{\mathbf{P}} = \arg\min_{\mathbf{P}} \sum_{i=1}^{N} \left[ u_i \cdot \vec{r}_3^T \vec{x}_i - \vec{r}_1^T \vec{x}_i \right]^2 + \left[ v_i \cdot \vec{r}_3^T \vec{x}_i - \vec{r}_2^T \vec{x}_i \right]^2 \tag{3.20}$$

$$\text{subject to: } \|\mathbf{P}\|_F = 1 \tag{3.21}$$

because $\mathbf{P}$ is unique up to scaling we incorporate the constraint that the Frobenius-norm[1] of $\mathbf{P}$ have to be equal to 1.

**Rule of thumb:** Always optimize differences in the image space resp. in space of observation.

This rule gets violated when we multiply with the denominator. But after multiplying with he denominator we have a linear optimization problem and get a closed form solution. Without multiplying with the denominator we have a non-linear opti-

[1] whichs means the sum of squares of the components of $\mathbf{P}$ have to be one

mization problem which is hard to solve, but we can optimize this with a numerical approach like Newton-Raphson Method and use the closed-form Solution as an Initialization, which leads to better results and due to the Initialization the method converges fast.
To make life easier, we can rewrite the linear part in matrix form, where the measurement matrix $\mathbf{M}$ will include the information on the 3-D calibration points and the measured 2-D points according the equations in 3.18. The equations in 3.18 looks then like:

$$\mathbf{M} \begin{pmatrix} p_{1,1} \\ p_{1,2} \\ \vdots \\ p_{3,3} \\ p_{3,4} \end{pmatrix} = 0 \tag{3.22}$$

with this, the calibration problem is reduced to the computation of the nullspace of measurement matrix $\mathbf{M}$, which can be done using SVD. And due to our knowledge about the calibration parameters and the constraint we know that the rank of matrix $\mathbf{M}$ is 11 and thereby $\mathbf{M}$ have a non-trivial nullspace !
The objective function for this looks like:

$$\|\mathbf{Mp}\|^2 \rightarrow \min, \qquad \text{subject to } \|\mathbf{p}\|^2 = 1 \tag{3.23}$$

which can be done with the Lagrange multiplier method:

$$\mathbf{p}^T\mathbf{M}^T\mathbf{Mp} - \lambda(\mathbf{p}^T\mathbf{p} - 1) \rightarrow \min \tag{3.24}$$

compute the derivative and the zero crossings:

$$2\mathbf{M}^T\mathbf{Mp} - 2\lambda\mathbf{p} = 0 \tag{3.25}$$

which is nothing else then:

$$\mathbf{M}^T\mathbf{Mp} = \lambda\mathbf{p} \tag{3.26}$$

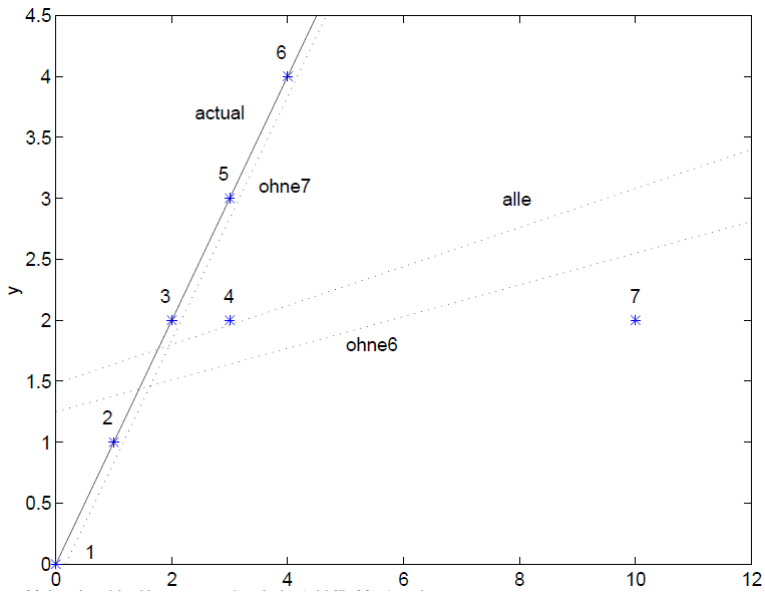
So the components of the projection matrix $\mathbf{P}$ result from the eigenvector belonging to the smallest eigenvalue. As mentioned above this linear estimate of $\mathbf{P}$ is an excellent initialization for the non-linear least square estimation of the projection matrix!

## 3.5. RANSAC

Problem in calibration are inaccuracies in observations and outliers, there are 2 types of outliers:

- badly localized points
- wrong correspondence

in figure 3.5 we can see the impact of those points.The dotted line „alle " shows very good that a outlier like Point 7 can mess up the whole calibration progress. The **RAN**dom **SA**mple **C**onsensus, short RANSAC, algorithm try to handle this




problems:

- draw samples uniformly and at random from the input data set
- cardinality of sample set: smallest size sufficient to estimate the model parameters compute the model parameters for each element the sample data
- evaluate the quality of the hypothetical models on the full data set
- cost function for the evaluation of the quality of the model
- inliers: data points which agree with the model within an error tolerance
- The hypothesis which gets the most support from the data set: best estimate.

In the case of our calibration this means, draw randomly the minimal amount of Points (6) we need to estimate **P** from the data set, do the estimation and then use the estimated Projection matrix to compute the difference between the projected

Points and the measured Points. With this we can identify outliers and at the end we can use all good Points to estimate **P** without Points which affects the estimate badly.

# 4. Reconstruction

## 4.1. Thomography

the basic idea of Thomography is to solve the Puzzel in figure 4.1

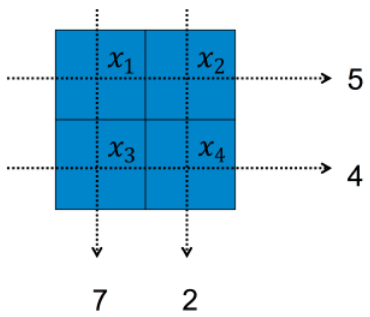


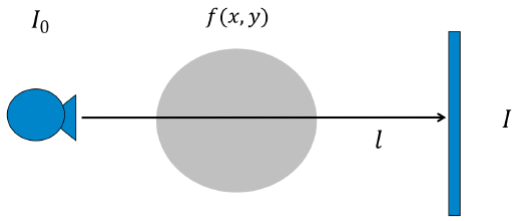

Abbildung 4.1.: basic Tomography Idea

$$\begin{aligned} x_1 \quad &+x_3 \quad &= 7 \qquad & x_1 \quad &= 3 \\ x_2 \quad &+x_4 \quad &= 2 \qquad & x_2 \quad &= 2 \\ x_1 \quad &+x_2 \quad &= 5 \qquad & x_3 \quad &= 4 \\ x_3 \quad &+x_4 \quad &= 4 \qquad & x_4 \quad &= 0 \end{aligned} \tag{4.1}$$

for real data this kind of method gets very fast very large, imagine a size of $512 \times 512 \times 512$ leads to 134 217 728 unknowns !

## 4.2. X-Ray attenuation Law - Beer's Law

The physical observation about X-Ray projection is that you start with an Intensity at the X-Ray source $I_0$ and the Object have a density function $f(x, y)$, which gives you a attenuation value in the space, and then the X-Ray hits the detector where we measure the Intensity $I$. The attenuation is characterized by the line Integral along




the line $l$, where you integrate the function of the object along the line $l$, which gives you the measured Intensity $I$: **Beer's Law** describes that in the way that we

$$
\begin{aligned}
I =& I_0 e^{-(\int f(x,y)dl)} \\
ln\frac{I}{I_0} =& -\int f(x,y)dl \qquad \stackrel{!}{=} p
\end{aligned}
$$

Abbildung 4.2.: Beer's Law

have an exponential decay along the line l from the original Value $I_0$ down to $I$. computing the function $f$ of the object out of many line Integrals is more or less the problem of solving a system of integral equations, which is the reconstruction process. Radon has shown that if we have a infinite number of X-Ray projections you can reconstruct it properly, in practice a finite number of Projections is enough for meaningful reconstructions.
If we consider only those points that sit on the line going through the origin (see fig: 4.3), we can describe these lines with this equation:

$$\vec{n}^T \vec{x} - d = 0 \tag{4.2}$$

where $\vec{n}$ is the normal vector of the line. This equation is fulfilled if a point lies on the line and returns the signed distance to the line if not. We can rewrite the line integral to a double integral over all x- and y-values and preserve the line integral using Dirac's delta function 1.3.2. If the points lie on the Line Dirac's delta function

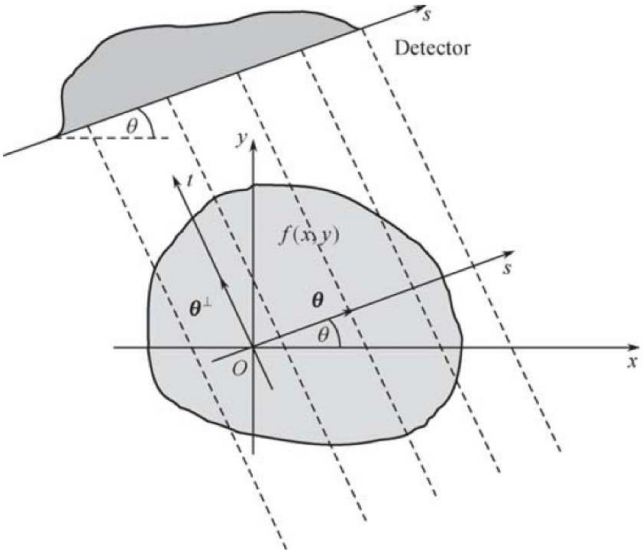



Abbildung 4.3.: parallel projection rays

Abbildung 4.4.: The x-axis represents $s$ and the y-axis represents the rotation-angle $\theta$

give use 1 and otherwise 0, which means the double integral collapses down to the line integral. We get this reformulated formula:

$$p(s,\theta) = \int f(x,y)dl = \int\int f(x,y)\delta(x \cdot \cos\theta + y \cdot \sin\theta - s)dxdy \tag{4.3}$$

the parameter $s$ is the offset if we move our straight line from the origin to another point (such that the line is still parallel to the line though the origin, like in figure 4.3) We can now follow a Detector-element over the acquisition process and build up a so called Sinogram, where on the x-axis we plot $s$ and on the y-axis the rotation-angle. A point in the Rotationcenter will be a straight line, a point which is not in the Rotationcenter will shows a sinusoid-curve (simple Example with 2 Points 4.4). As a

simple Example we reconstruct the Problem of 4.1, where $\mathbf{P}$ are the measurements and Matrix $\mathbf{A}$ represents the linear equations we set up 4.1:

$$\mathbf{P} = \mathbf{AX} \tag{4.4}$$

$$\mathbf{P} = \begin{pmatrix} 7 \\ 2 \\ 5 \\ 4 \end{pmatrix}, \qquad \mathbf{A} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}, \qquad \mathbf{X} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \tag{4.5}$$

To solve the reconstruction Problem we have to Inverse $\mathbf{A}$ (which does not work, but pseudo-inverse works):

$$\mathbf{A}^{-1}\mathbf{P} = \mathbf{X}$$

But a Problem in practise is the size of the Matrix. Imagine we have a Object with $512 \times 512 \times 512$ we want to measure and our detector captures Images of size $512 \times 512$ and we do 512 projections from different Angles, this leads to the following Size of the Matrix:

$$\mathbf{A} \in \mathbb{R}^{512^3 \times 512^2 \times 512}, \qquad 512^6 \cdot 4\text{Byte} = 65536\text{TB}$$

so measuring a small object with a low resolution still needs a 65536 Terabyte for Matrix $\mathbf{A}$!

## 4.3. Back-projection

A Idea of reconstruction is the Back-projection. Which means we smear back additive the measured values along the projection line: So we need a mathematical description

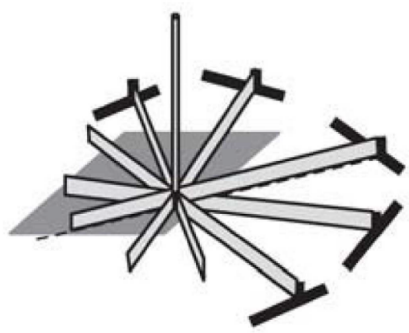


Abbildung 4.5.: smear back additive the measured Values

for Backprojection. Backprojection is the adjoint of projections, which is explicitly not the inverse of Projection. We described the Projection with the Matrix $\mathbf{A}$, so the adjoint of a Matrix is its transposed. So the Backprojection is a Matrix multiplication with the transposed projection matrix:

$$\text{Projection: } \mathbf{A}\vec{x} = \vec{p}, \qquad \Rightarrow \qquad \mathbf{A}^T\mathbf{A}\vec{x} = \underbrace{\mathbf{A}^T\vec{p}}_{\text{Backprojection}} \tag{4.6}$$

Applying Backprojection on our introduction Example 4.1 we get: where $\vec{p}$ is the

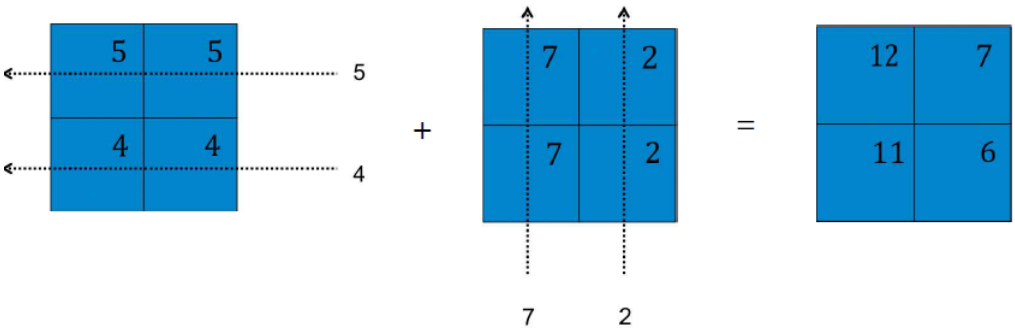



Abbildung 4.6.: smear back additive the measured Values

observed part (projections), $\vec{x}$ is function to be reconstructed and $\mathbf{A}$ is the system matrix which carries the Geometry (the Matrix is fixed by the Geometry-Setup) this are obviously not the correct Result, but the structure comes back up to scaling, if we look at the Values from the beginning we see that we have on the left side two higher values and on the right side to lower values and these value pairs are similar. This structure we can also see in 4.6.
The mathematical expression of this example is:

$$\mathbf{B} = \mathbf{A}^T\mathbf{P} \tag{4.7}$$

$$\mathbf{P} = \begin{pmatrix} 7 \\ 2 \\ 5 \\ 4 \end{pmatrix}, \qquad \mathbf{A}^T = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}, \qquad \mathbf{B} = \begin{pmatrix} 12 \\ 7 \\ 11 \\ 6 \end{pmatrix} \tag{4.8}$$

which is only a intermediate-Step to get the result $\vec{x}$.
For the continuous case in Integral notation we get:

$$b(x,y) = \int_0^{\pi} p(s,\theta) \underbrace{|_{s=x\cos\theta+y\sin\theta}}_{\text{normal vector of the given line}} d\theta \tag{4.9}$$

so you sum up - integrate - over all considered rotation angles for a given line.
If you smear back all the Projections we get something like this: For a good Re-

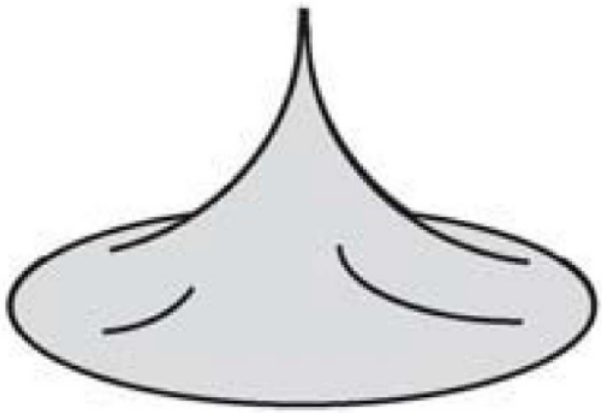


Abbildung 4.7.: smear back additive the measured Values

construction of the Point we need some „negative Wings“at the smearing back step such that only the Points gets reconstructed. This can be done with a filter step, but before introduce this filter which allows a good reconstruction we should look in the mathematical steps.

## 4.4. Central Slice Theorem

Beside beer's Law the Central Slice Theorem is the second core component for the function of CT and Reconstruction.
Basically the Central Slice Theorem says:

> The Fouriertransform of the Projection is equal to the 1-D Fouriertransform of the 2D-Object trough the Origin, parallel to the Projectionline (in our case the Detector)

This can be described in the (for the oral exam very important) Picture: **Picture Explanation:**
If you fouriertransform the Projection of the 2D-Object with the density Function

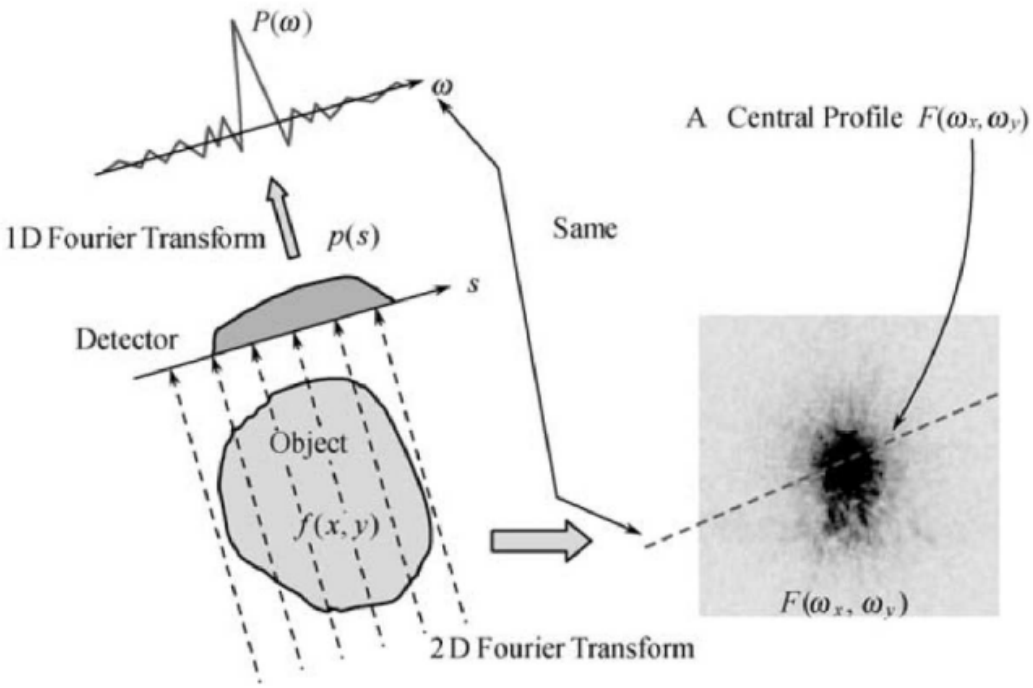



Abbildung 4.8.: Central Slice Theorem

$f(x, y)$, then you can find exactly that function on a line in the 2D-Frouriertransform of the Object which goes trough the Origin and is parallel to the Projection line. Of course the central slice theorem is a mathematical approach: Lets Fouriertransform the projection with the angle $\theta = 0$:

$$FT((s, 0)) = \int_{-\infty}^{\infty} p(s, 0)e^{2\pi i\omega s} ds$$

know look at how we get $p(s, 0)$: We project our object function $f(x, y)$ according to beer's law on the projectionline with the angle $\theta = 0$. So we just use the definition of the Projection: $p(s, 0) \int_{-\infty}^{\infty} f(x = s, y)dy$:

$$\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \underbrace{f(s, y)dy}_{p(s,0)} \quad e^{-2\pi i\omega s} \quad ds$$

rename s back to x (for easier reading) and bring $e^{-2\pi i\omega s}$ into the inner Integral (this is allowed because it does not depend on y):

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-2\pi i\omega x}dx\ dy$$

and now add to the e-function in the exponent the value $(+0 \cdot y)$:

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-2\pi i\omega x+0\cdot y}dx\ dy$$

and this is the just the 2D-Fouriertransform at the point $\omega = 0$ in frequency Space! This can be generalized and shown that its true for all Angle $\theta$.

## 4.5. Filtered Backprojection

Just start with the definition for the Projection:

$$P(\omega,\theta) = \int_{-\infty}^{\infty} p(s,\theta)e^{-2\pi i\omega s}ds$$

now take the definition for arbritrary Angle use the dirac's delta notation:

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\delta(x\cos\theta + y\sin\theta - s)dx\ dye^{-2\pi i\omega s}\ ds$$

now we can reorganize the terms:

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\int_{-\infty}^{\infty}\delta(x\cos\theta + y\sin\theta - s)e^{-2\pi i\omega s}\ ds\ dx\ dy$$

we can drop now the inner Integral because only for $x\cos\theta + y\sin\theta = s$ the dirac's delta function is not Zero:

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-2\pi i\omega(\underbrace{x\cos\theta + y\sin\theta}_{s})}\ dx\ dy$$

this looks like the 2D-Fouriertransform of the function $f(x, y)$, to make this more obvious we can write this more clear:

$$F(\omega\cos\theta, \omega\sin\theta) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-2\pi i((\underbrace{\omega\cos\theta}_{\zeta})\cdot x+(\underbrace{\omega\sin\theta}_{\eta})\cdot y)}\ dx\ dy$$

This is the 2D-Fouriertransform of $f(x, y)$ in **Polarcoordinates** ! IMAGE : EXPLAIN
The coordinate Transform needs to incorporate into the Integral:

$$f(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(\zeta,\eta)e^{2\pi i(\zeta x+\eta y)}\ \ d\zeta\ d\eta$$

So we need here to apply a coordinate transform:

$$\zeta = \omega\cos\theta$$
$$\eta = \omega\sin\theta$$

to do this we need to multiply with the determinant of the Jacobian: the first column is the derivative with respect to $\omega$ and the second column with respect to $\theta$:

$$\text{Jacobian} = \begin{pmatrix} \cos\theta & -\omega\sin\theta \\ \sin\theta & \omega\cos\theta \end{pmatrix} \Rightarrow \det\ \mathrm{J} = \omega(\underbrace{\cos^2\theta + \sin^2\theta}_{=1}) = \omega$$

now we can write our integral with $\omega\cos\theta$ and $\omega\sin\theta$ and multiply it with the Determinant of th Jacobian as it is required for the coordinate transform:

$$\int_{0}^{2\pi}\int_{-\infty}^{\infty} |\omega| \underbrace{F(\omega\cos\theta, \omega\sin\theta)}_{2D-\text{Ft in Polar}} e^{2\pi i\omega(x\cos\theta+y\sin\theta)}\ \ d\omega d\theta$$

**outer Integral** ($\theta$): the $\theta$ goes from 0 to $2\pi$ because we rotate around the whole object
**inner Integral** (Omega): the omega is just the scaling factor of our unit vector in terms of Polar coordinates, if you go from $-\infty$ to $\infty$ we count twice. We make the $\omega$ absolute because you always read only positive values: e.g. for the angle 15ř you just read the positive values of the vector, if you want the negative you just rotate to 195ř and read again only the positive values.

We know that the Fouriertransform of the function we look for in polarcoordina-

tes is the same as the fouriertransform of all the projections. So we can replace $F(\omega\cos\theta, \omega\sin\theta)$ with the Fouriertransform of the Projections:

$$\int_0^{2\pi}\int_{-\infty}^{\infty} |\omega| \cdot P(\omega\cos\theta, \omega\sin\theta) \cdot e^{2\pi i\omega(x\cos\theta + y\sin\theta)} \quad d\omega d\theta$$

know we see a product of the Fouriertransform with the absolute Value $|\omega|$ which is in the spatial domain a Convolution ! So the determinant of the Jacobian is a Filter !

**meaning of the Filter:** because we sample the Fouriertransform of the function we look for in polar coordinates we have a lot of samples around the origin and less samples farer away from the origin. So we have to weight the view outer samples higher than the inner samples (where we have a lot from). And if you look the function of the absolute Value it does exactly this.

> So our Filter weights the inner Samples with lower weights and the outer Samples with higher weights !

The outer Integral (from 0 to $2\pi$) is the Backprojection (smear it back) and the multiplication with $|\omega|$ is the Filter step. In practice the Filter needs some modifications. The absolute value function weights the sample in the origin with 0 because there we have theoretically an infinite number of Samples but in practice we have not. Also we cant integrate to $\infty$ so we need some point where we cut the absolute Value.

### 4.5.1. Reconstruction with Hilbert Backprojection

An alternative way for reconstruction is to use the Hilbert-transform. For this we have to rewrite the $|\omega|$ in the following way:

$$|\omega| = (2\pi i\omega) \cdot \left\{ \frac{1}{2\pi}[-i\ sign(\omega)] \right\}$$

the multiplication with $-\ sign(\omega)$ in frequency Domain is the Hilbert transform. Now we can describe the reconstruction as follows:

- Compute first derivative of the detector row (derivative w.r.t. s):

$$q_1(s,\theta) = \frac{\partial p(s,\theta)}{\partial s}$$

- Apply Hilbert Transform:

$$q_2(s,\theta) = \frac{1}{2\pi^2 s} * q_1(s,\theta)$$

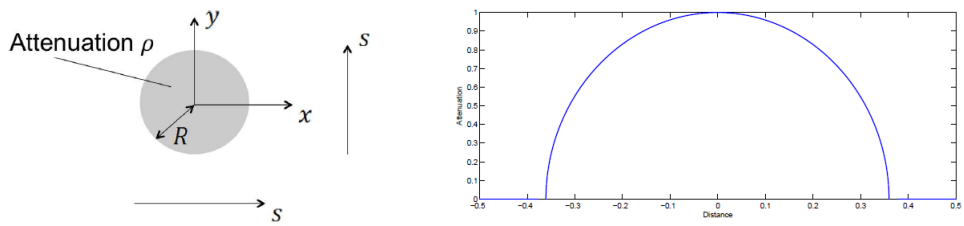
- Backproject $q_2(s,\theta)$:

$$f(x,y) = \int_0^{\pi} q_2(s,\theta)|_{s=x\cos\theta+y\sin\theta}\ d\theta$$

but take into consideration that we compute the first derivative. And the computation of derivatives from measured, and so noisy data, could be cause problems because the derivation can double the error in the worst case !

## 4.6. Practical aspects of FBP

Lets look at a very simple case, our object is a disk with Attenuation $\rho$ The Back-




(a) Object (b) Projection of the Object

projection could look like:

$$q(s,\theta) = h(s) * p(s,\theta) \qquad h(s) = \int_{-\infty}^{\infty} |\omega| e^{2\pi i\omega s} d\omega$$

$$f(x,y) = \int_0^{\pi} q(s,\theta)|_{s=x\cos\theta+y\sin\theta}\ d\theta$$

So the projections will be convolved with the Filterkernel $h(s)$. Here we get the problems shortly remarked in the FPB-section. A usable representation of the Filterkernel is to cut at $-\frac{1}{2}$ and $\frac{1}{2}$ and compute the spatial domain representation of this kernel:

$$h(s) = \int_{-\frac{1}{2}}^{\frac{1}{2}} |\omega| e^{2\pi i\omega s}\ d\omega \Rightarrow h(s) = \frac{1}{2}\frac{\sin \pi s}{\pi s} - \frac{1}{4}\left[\frac{\sin\frac{\pi s}{2}}{\frac{\pi s}{2}}\right]^2$$

and the discretisation of this is called the **Ram-Lack** convolver:

$$h(n) = \begin{cases} \frac{1}{4}, & n = 0 \\ 0, & n \text{ even} \\ -\frac{1}{n^2\pi^2}, & n \text{ off} \end{cases}$$

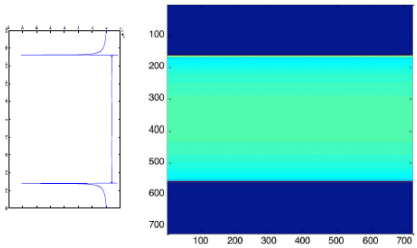
the discrete spatial form of the Kernel-filtering return a weight not 0 at the Origin, so it take into account that we have not an infinite number of Samples there(this number is only theoretical).
If we look at the computational complexity of the reconstruction:

- Precompute filter $h(s)$ in spatial domain: $\mathcal{O}(N)$
- Transform filter to frequency domain $H(\omega)$ via FFT: $\mathcal{O}(N \log N)$
- For each of $\#P$ projections:
  - Compute FFT of $p(s, \theta)$: $\mathcal{O}(N \log N)$
  - Apply filter $P(\omega, \theta) \cdot H(\omega)$: $\mathcal{O}(N)$
  - Compute filtered projection $q(s)$ via FFT: $\mathcal{O}(N \log N)$

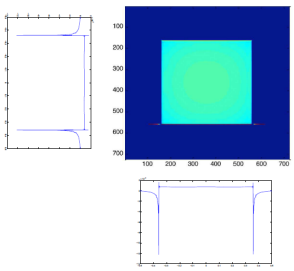
$\Rightarrow$ so we get a total complexity of:

$$\mathcal{O}(N + N \log N + \#P(N + 2N \log N)) = \mathcal{O}(\#PN \log N)$$

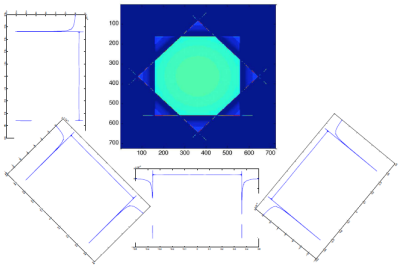
so the important aspect is here,**the complexitity** of the reconstruction **is linearly in the number of Projections** ($\#P$)



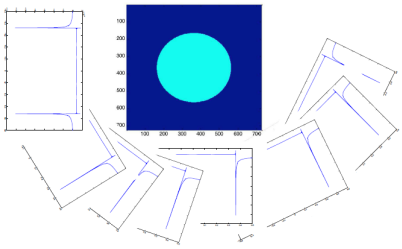
(c) Backprojection with one Projection



(d) Backprojection with two Projection



(e) Backprojection with four Projection



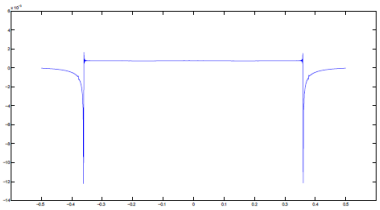
(f) Backprojection with all Projection

Abbildung 4.9.: Example Backprojection of one Slice from the disk-Object

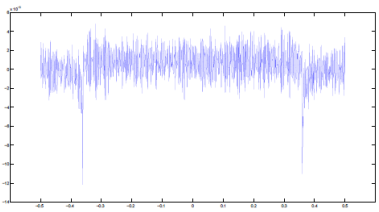
## Practical Problems with Noise

In the real world we use sensors and so we have to handle with Noise. Beside the analytical point of view the reconstruction Algorithm differ in the numerical handling of the noise data.

in this figure we can see, that the noisy Data are very dirty after Filtering. This



(a) Non-Noisy Projection convolved with the Ramp-Filter



(b) Noise Projection convolved with the Ramp-Filter

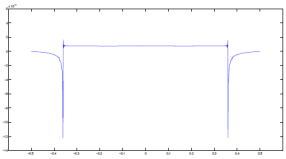
Abbildung 4.10.: Projection convolved with the Ramp-Filter

will definitively result into very bad reconstruction !

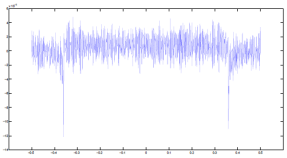
We can incorporate into the Filtering process a Windowing Function to smooth the Noise and reduce the bad impact. For this we can use the associative properties from Filters, so we multiply the Ramp-Filter with a Windowing-Filter and this combined

Filter we multiply with the Projection.
Possible Windowing-Filter:

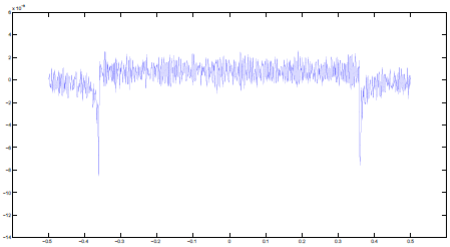
- Rectangular Window (rectangular in frequency domain)
- Cosine Window: $cos(\pi x)$
- Shepp-Logan Window: $\frac{\sin(\pi x)}{\pi x}$



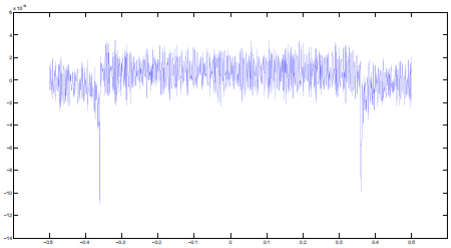
(a) Non-Noisy



(b) Noise Projection convolved with the Ramp-Filter



(c) Noise Projection convolved with the Shepp-Logan-Filter



(d) Noise Projection convolved with the Cosin-Filter

Abbildung 4.11.: Example Backprojection of one Slice from the disk-Object

Obvious the Windowing gives us better results but they are different. The Shepp-Logan-Filter reduces the Noise-Impact very good, but we can see that the two spikes into the negative direction are not that „long“(badly for sharp edges). The cosine-Filter does not reduce the noise as good as shepp-logan but the negative spikes are longer. From this example we can summarize that the Filter we use for reconstruction depends to the result we want to have.
Summarize about Noise:

- Is amplified by ramp Filter
- Has to be taken care of an appropriate manner
- is indirectly proportional to the applied dose (we have low-dose Images because of the patient health)
- Affects different reconstruction methods differently (for example hits hilbert-approach due to the differentiation)

## 4.7. Fan Beam Geometry

All considerations above are done on the parallel beam Geometry, so the X-Rays are parallel. And now we want to look at geometries where the x-rays are not parallel.
The Parallel beam algorithm cannot be applied directly anymore. We **do not** have

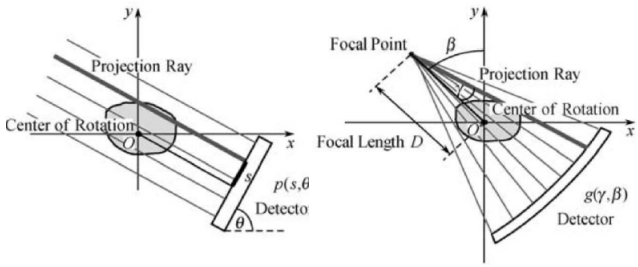



Abbildung 4.12.: left the considered parallel Beam geometry. Right the Fan Beam geometry we consider in this chapter

a central slice theorem anymore !
**Basic Idea:** is to use all acquired Images and reorganize the measured Values such that we get back the parallel beam geometry and with that the central slice theorem so we can reconstruct as we consider before.
in the Following some Examples how the Projection of the Object could change in Fan-Beam geometry setup: Before we come to the sketched Idea, we take a short

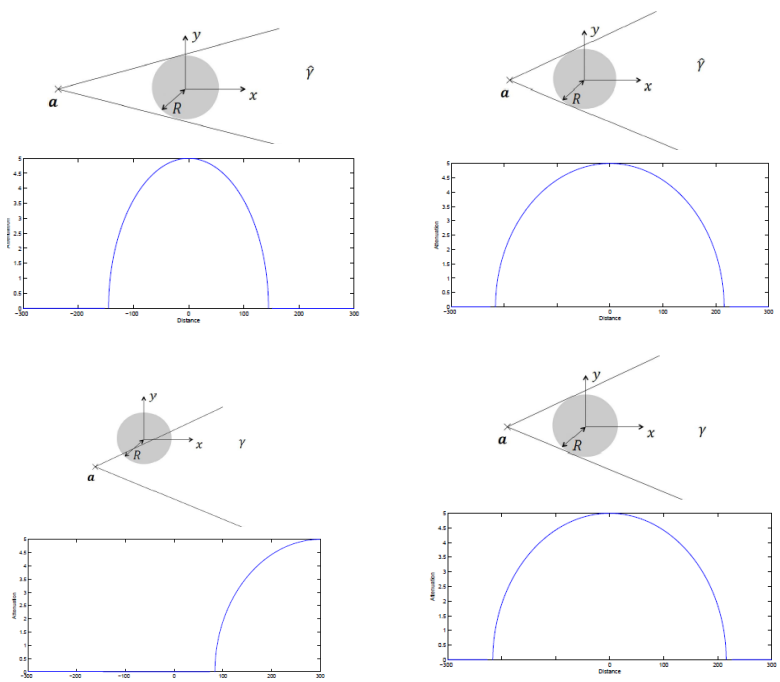
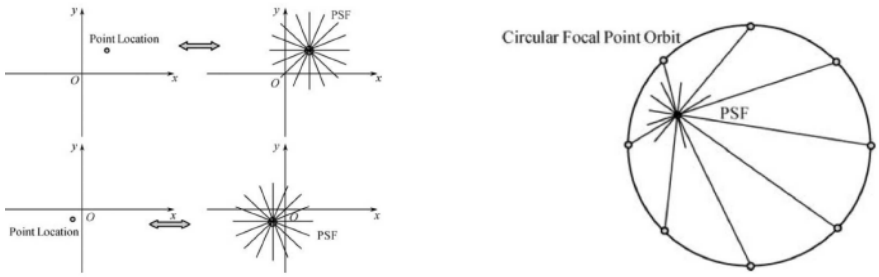



Abbildung 4.13.: Example Fan-Beam Projections of the Object

look on the Point Spread Function:

### Point-Spread-Function

Do a normal Backprojection (without Filter) of a Single point out of all Projections we have acquired. The Point-Spread-Function tells us how a single Point is „smeared“by the System in the Backprojection step. It tells us how is a single Point blurred by the acquisition Device.




(a) PSF - Parallel Beam (b) PSF - Fan Beam

Abbildung 4.14.: PSF for parallel and fan Beam geometry

- For Parallel Beam Geometry: Draw a line through the reconstructed point that is perpendicular to the detector and repeat for every detector position.

⇒ In this case, the point spread function is shift-invariant, i.e. every reconstfucted point shows the same pattern

- For Fan Beam Geometry: Draw a line trough the reconstructed point and the source position and repeat for every source position.

⇒ For a complete circle, the pattern is also shift-invariant

It can be shown that the full circle PSF is equivalent to the parallel beam PSF.

> This observation tells us that if you project and backproject an image, you will get the **same blurred** version of that image, **regardless** of the use of parallel-beam or fan-beam geometry !

If the original image is $f(x, y)$ and tghe backprojection of the projection data is $b(x, y)$, then the PSF can be shown to be $\frac{1}{r}$ where r is the distance of the currently considered point to the rotation axis: $r = \sqrt{x^2 + y^2}$. b and f are related by:

$$
\begin{aligned}
b(x, y) &= f(x, y) * * \frac{1}{r} \\
B(\omega_x, \omega_y) &= F(\omega_x, \omega_y) \cdot \frac{1}{\sqrt{\omega_x^2 + \omega_y^2}} \\
F(\omega_x, \omega_y) &= B(\omega_x, \omega_y) \cdot \sqrt{\omega_x^2 + \omega_y^2}
\end{aligned}
$$

The relation ship of b and f after the Fourier transform is shown in line 2, the Fourier transform of $r = \frac{1}{\sqrt{x^2+y^2}}$ is $\frac{1}{\sqrt{\omega_x^2+\omega_y^2}}$.

We know that we need to apply a 2D ramp filter to the backprojected Image b to get the original Image f. We can use the same approach for the fan-beam geometry. So apply the 2D ramp filter $|\omega| = \sqrt{\omega_x^2 + \omega_y^2}$ on both sides leads to the third Line. Know we just have to apply the inverse Fourier transform on $F(\omega_x, \omega_y)$ to get the original Image $f(x, y)$.
This is the **backproject-then-filter** algorithm !

## Parallel-Beam to Fan-Beam Algorithm Conversion

We want to have a filter-then-backproject (FBP) algorithm, so we need another approach than the above with the PSF.
Now we come back to the sketched basic Idea from the beginning of the Fan-Beam chapter.
We need to find equal rays in both geometries. For that we need a description for the rays.

- Parallel-Beam: we used the rotation Angle of the detector $\theta$ and the pixel index $s$ to describe the Ray.
- Fan-Beam: we can use $\beta$ which is the angle between the central ray which is perpendicular to the detector and the y-axis and then the angle $\gamma$ between the central Ray and the Ray we want to consider.

So we have our known Function $p(s, \theta)$ for parallel beam geometry and for fan-beam geometry we got $g(\gamma, \beta)$. For each fan-beam ray-sum $g(\gamma, \beta)$, we can find a parallel-beam ray-sum $p(s, \theta)$ that has the same orientation as the fan-beam ray with the relations seen in following figure: **Figure description:** They Rays are fan-beam

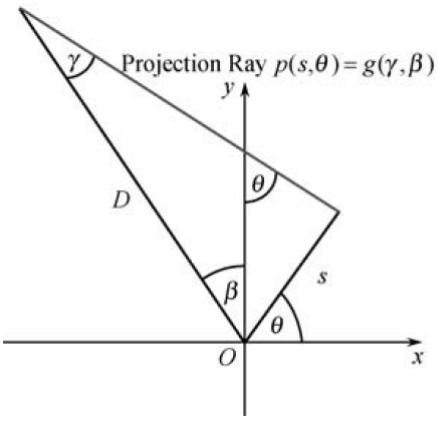



Abbildung 4.15.: A fan-beam ray can be represented using the parallel-beam, geometry parameters (called: Rebinning)

geometry Rays but the detector-line (where $\theta$ and $s$ are written) is used from the parallel-beam geometry. So we mixed both geometry to find the relation-ship !
With basic knowledge about the interior Angle in a triangle (180°) and basic trigonometry we can find the following representations:

$$\theta = \gamma + \beta$$
$$s = D \cdot \sin\gamma$$

where D is the focal length. Describe the big triangle with $\gamma, \beta$:

$$180 = \gamma + \beta + 180 - \theta \Rightarrow \theta = \gamma + \beta$$

Now we can assign $p(s,\theta) = g(\gamma,\beta)$. After rebinning the fan-beam data into the parallel-beam format we then use a parallel-beam image reconstruction algorithm to reconstruct the Image. A big disadvantage of this approach is that we need to interpolate in the rebinning step, and interpolation introduce errors because we don't know the real function and can only approximate it.
We can use the Idea from above and incorporate it directly into the parallel-beam integral notation. So we start with the FPB:

$$f(x,y) = c \cdot \int_{0}^{2\pi} \int_{-\infty}^{infty} p(s,\theta) \cdot h(x\cos\theta + y\sin\theta - s)\ ds\ d\theta$$

change this to polar coordinates with $x = r \cdot \cos\varphi$ and $y = r \cdot \sin\varphi$:

$$f(r,\varphi) = c \cdot \int_0^{2\pi} \int_{-\infty}^{infty} p(s,\theta) \cdot h(r \cdot \cos\varphi \cos\theta + r \cdot \sin\varphi \sin\theta - s)\ ds\ d\theta$$

simplify:

$$f(r,\varphi) = c \cdot \int_0^{2\pi} \int_{-\infty}^{\infty} p(s,\theta) \cdot h(r \cdot \cos(\theta - \varphi) - s)\ ds\ d\theta$$

we know from the consideration above that $p(s,\theta) = g(\beta,\gamma)$ only for the case:

$$\theta = \beta + \gamma \qquad \text{and} \qquad s = D \sin\gamma$$

so we can replace p with g, but we have to make sure that the mentioned case is fulfilled. That is guaranteed if we replace $\theta$ and $s$ with the formulas. If we do that, we change the variables over we integrate so we need again the determinant of the Jacobian:

$$f(r,\varphi) = c \cdot \int_0^{2\pi} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} g(\gamma,\beta) \cdot h(r \cos(\underbrace{\gamma + \beta}_{\theta} - \varphi) - \underbrace{D \sin\gamma}_{s}) \underbrace{D \cdot \cos\gamma}_{det(J)} d\gamma\ d\beta$$

The bad thing about this it is no convolution. And a convolution is much easier to compute than an numerical integration. To get a convolution we have to consider the point we want to reconstruct in another coordinate System. We can describe

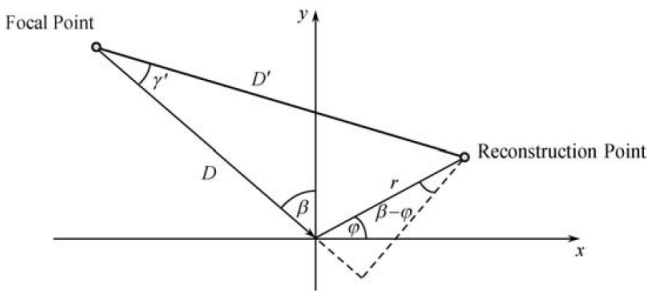



Abbildung 4.16.: The reconstruction point $(r,\varphi)$ defines the angle $\gamma'$ and distance $D'$

the point we want to reconstruct instead of $r, \varphi$ with $\gamma', D'$. Because we change the

inner integral, the $\beta$ is arbitrary but fixed: so rewrite $r \cdot \cos(\beta + \gamma - \varphi) - D \sin\gamma$ with the other coordinate system as shown in figure:

$$r \cdot \cos(\beta + \gamma - \varphi) - D \sin\gamma = D' \sin(\gamma' - \gamma)$$

$$\Longrightarrow \qquad f(r, \varphi) = c \cdot \int_0^{2\pi} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} g(\gamma, \beta) \cdot h(D' \sin(\gamma' - \gamma)) D \cdot \cos\gamma \; d\gamma \; d\beta$$

and this almost looks like a convolution, now we take a property of the ramp-filter:

$$h(s) = \int_{-\infty}^{\infty} |\omega| e^{2\pi i \omega s} d\omega \Rightarrow h(D' \sin(\gamma' - \gamma)) = \int_{-\infty}^{\infty} |\omega| e^{2\pi i \omega D' \sin(\gamma' - \gamma)} d\omega$$

this formula can be massaged and end the end we get the ramp-filter for fan-beam:

$$h_{fan} = \frac{D}{2} (\frac{\gamma}{\sin\gamma})^2 \cdot h(\gamma)$$

this we can put into our reconstruction algorithm:

$$f(r, \varphi) = \int_0^{2\pi} \frac{1}{D'^2} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \cos\gamma \; g(\gamma, \beta) h(\gamma' - \gamma) \; d\gamma \; d\beta$$

and this is the Filtered-Backprojection-Algorithm for the Fan-Beam geometry (for curved Detector, so for equiangular Detectors)!
The $\cos\gamma$ is called **cosine weighting**, and we can see, the more the ray diverge from the central ray the smaller the weight gets.
Because at this point its very confusing with all the coordinate systems we consider and we use to get these formula here are all used coordinate systems:

- $(\theta, s)$-coordinate system
- $(\beta, \gamma)$-coordinate system
- $(x, y)$-coordinate system
- $(r, \varphi)$-coordinate system
- $(D', \gamma')$-coordinate system

Another important point is, that the backprojection formula is different for the different detectors. The Filtered-Backprojection-Formular we dirved above is valid for the right Detector-type! In Summary here again the Algorithm:

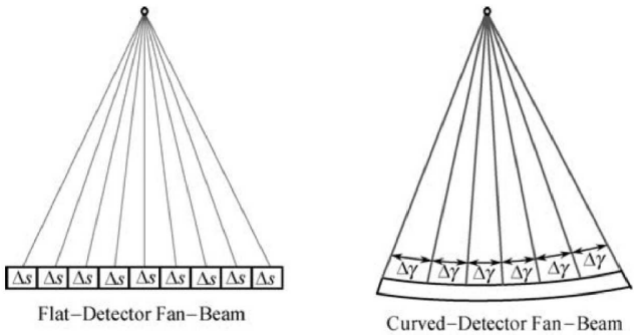



Abbildung 4.17.: Left is an Equally-spaced and right an Equiangular Detector

- Perform cosine weighting:

$$g_1(\gamma, \beta) = g(\gamma, \beta) \cos \gamma$$

- Apply fan beam filter:

$$g_2(\gamma, \beta) = g_1(\gamma, \beta) * h_{fan}(\gamma)$$
$$h_{fan}(\gamma) = \frac{D}{2}\Big(\frac{\gamma}{\sin \gamma}\Big) h(\gamma)$$

- Backproject with distance weight:

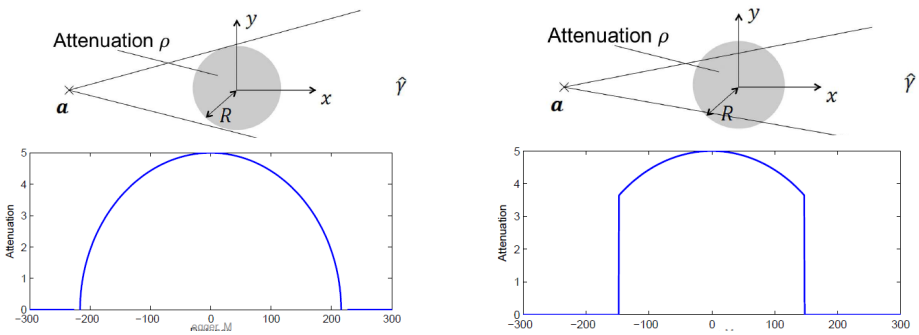
$$f(r, \varphi) = \int_0^{2\pi} \frac{1}{D'^2} g_2(\gamma', \beta)\ d\beta$$

### Short Scan - Redundancy of Rays

In the case of parallel beam geometry it is obvious that after an 180° degree rotation the rays gets the same, just the direction is switched. So with a full 360° degree measurement we measure twice.
In the case of Fan-Beam geometry this is different. There are cases where you have the ray two times and and some not, so you have to be aware that some measurement are measured twice and some not. There exists a standard weighting scheme called Parker weight.

## 4.8. Truncation

Truncation leads to reconstruction artefacts so we have to handle it.
On the left we see that the object is in the full fan, and we can measure the com-

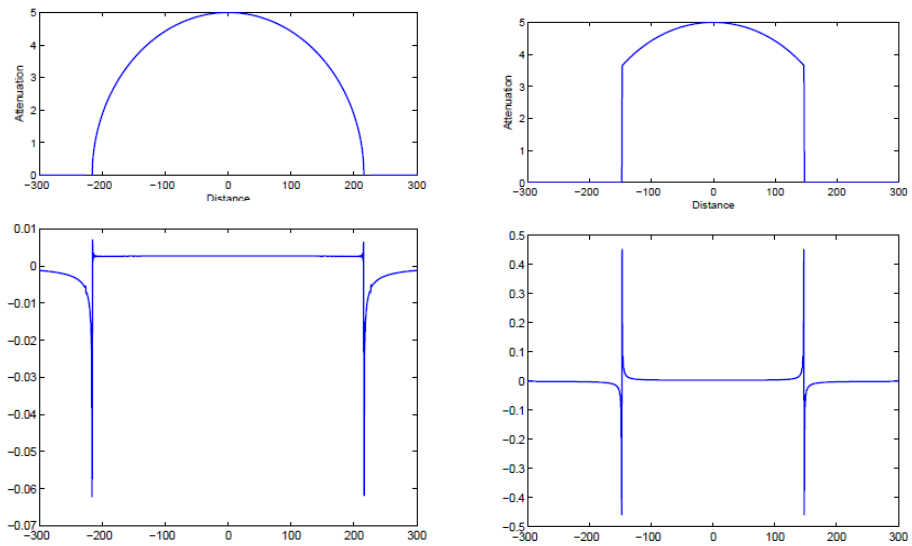



(a) No Truncation (b) with Truncation

Abbildung 4.18.: Truncation and the results in the Projection

plete projection of the object. On the right side we can see that the object is not completely in the fan and due to this we get an incomplete projection of the object. So this problem is a missing Data problem.
We can model the truncation as the complete projection multiplied with an rectangular window function, the result would look like the projection on the right side.
The main Problem is, that the ramp-filter is in fact a high-pass filtering and as you can imagine a high pass filtering on these huge steps (which are actually not real) will lead to problems. Lets take a look on the filtered Projections: As expected the huge steps which comes from the truncation leads to high filter response. If you compare the filter result with the left one (non Truncation) you can easily see that the filtered projections are completely different. After the filter step, the backprojection step is applied, so imagine you smear back the right filter response instead of the left: On the left side we can the a Water Cylinder reconstructed without truncation. On the Right side we see the same Water Cylinder but this time it was the truncation case. We can directly map the results to the filter response we have looked at. The very high valued edges on the right image are correlated to the high filter response due to the truncation, and afterwards like in the filter response the values drop fast to a low level, as we can observe in the right image.
This Intensity drop from outside to inside is called **„cupping“artifact**.
Truncation is not an academical Problem, there are reasons why we have truncation in real systems:

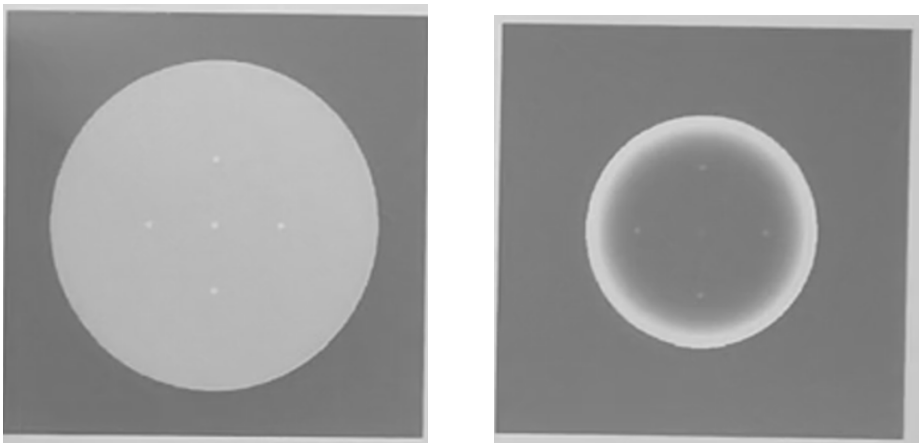
- Small Field of View: Detector too small to fit the whole anatomy
  - Typical for abdomen and thorax scans




(a) filtered Projection without Truncation

(b) filtered Projection with truncation

Abbildung 4.19.: Comparison of the filter result between projection with and without truncation



(a) No Truncation

(b) with Truncation (croped outside of FoV)

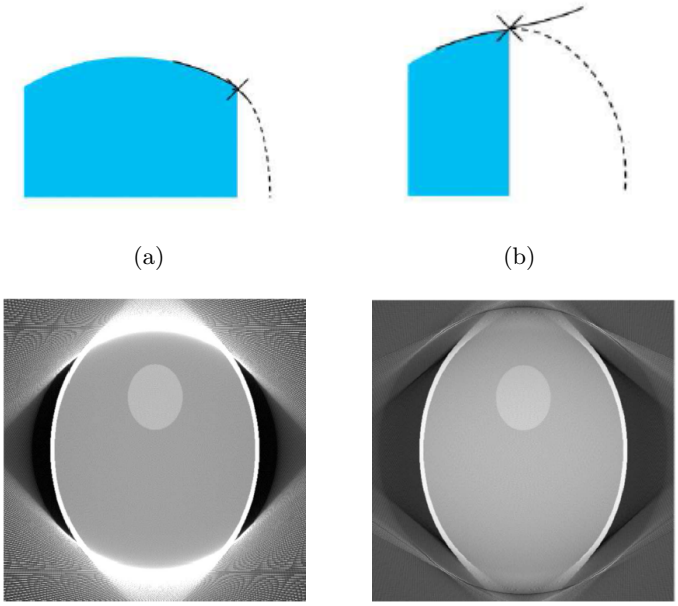
Abbildung 4.20.: Water Cylinder With Beads - Reconstruction

  - Medical instruments that extend the Field of View

- Collimation: Intentional truncation to reduce radiation dose
  - lead plates in x-ray source can be adjusted to focus on an ROI
  - lead absorbs most x-ray photons $\longrightarrow$ No information in projection

There are different ideas to solve the truncation problem, basically all solutions try to compensate the missing Data (all extrapolation methods make he assumption that the object is not infinite and will go to Zero at some point):

- Solution 1: Defect Pixel extrapolation
  - Model extrapolation as deconvolution: Use defect pixel interpolation algorithm from the beginning of the lecture
  - Unfortunately, the algorithm works not as well as expected
- Solution 2: Heuristic extrapolation
  - Use mirroring for extrapolation. In order to enforce a limited size of the object, a cosine-like weighting is added:



(a) (b)

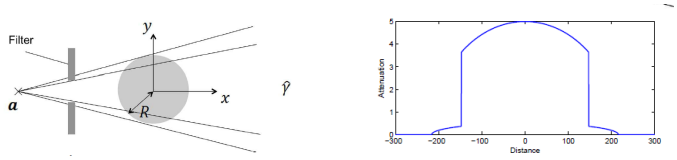
(c) truncated without correction (d) heuristic extrapolation

Abbildung 4.21.: Efficient correction for CT image artefacts caused by object extending

- Solution 3: Water Cylinder Assumption: Assume that the imaged object consists of water: Fit water cylinder model to observed data and use model to extrapolate
  - will work perfectly, if a water cylinder is imaged
  - yields good results for most objects (head, abdomen, etc..)

  - will yield suboptimal results if water cylinder assumption is violated. There exists other methods which makes other object assumptions out of water

- Solution 4: Use of Prior Knowledge: Do a full-scan with low-dose and a ROI scan (with truncation) with high dose
  - Use data from low-dose scan to complete the data from the second scan
  - Correction will be perfect, if the object did not change !
- Solution 5: Semi-transparent Filter




(a) (b)

Abbildung 4.22.: Semi-transparent Filter

  - use the low signal for extrapolation
  - Filter boundary must be located correctly (which may be influenced by the object)
  - Method has to be applied carefully in order not to introduce artificial high frequencies
  - yields perfect truncation correction

## Phatoms

Phantoms are used to evaluate reconstruction algorithms. Phantoms are used because x-rays are ionizing (so badly for patients) and we don't know the geometry of the patient but we know the geometry from the phantom exactly. So we know how the phantoms should look like and we can compare our reconstruction results with this.
We distinguish two kinds of phantoms:

Numerical / simulated phantoms: Originate from computer simulations, they are know exactly but have only limited realism

  - Shepp-Logan phantom (specified for 2D)
  - Forbild phantom (more complicated thatn shepp-logan)

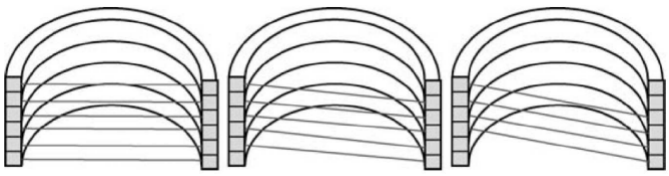
- X-Cat phantom (targeting the realism part)

Real phantoms: are designed with desired properties and are manufactured with a high accuracy.

## 4.9. 3D - Reconstruction

We have considered the 2D-Case of reconstruction.In reality detectors have multiple lines and instead of a fan-beam the x-ray source propagate the x-rays in a cone-beam. So we have to adjust our reconstruction approaches to the 3D-cases.
One Idea for the 3D-case is to do a slice-by-slice 2D reconstruction, and this is what we first discuss in the following. After that we discuss a another approach with the radon-transform. The assumption behind this idea is that the Projection-Rays are perpendicular to the rotation angle. Before we look what happens if this is not true we look at the sampling of the fourierspace

### 4.9.1. Parallel-Line Integral Data



In the left case the projection lines are perpendicular to the rotation axis and we will get complete Data and can do our slice-by-slice 2D reconstruction. The middle and the right case will lead to missing data, so we cannot reconstruct the function properly with the slice-by-slice 2D reconstruction approach.

#### 4.9.1.1. Central Slice Theorem for 3D

First of all we have to check if the central slice theorem for the 1D/2D-case can be generalized or be lifted up to the 2D/3D case, so we can still reconstruct.
and of course we can generalize the the central slice Theorem to higher dimensions (just look at the proof for 1D/2D and lift it up to higher dimensions). So in this case we want to sample the 3D-Fourier transform. For each measurement we get a plane, and at the end we get the sampled fourier transform of the object in cylindric-coordinates (like the Polar-coordinates for the 1D/2D-case): Like in the 1D/2D-case this holds for parallel beams and not for the cone-beam geometry.

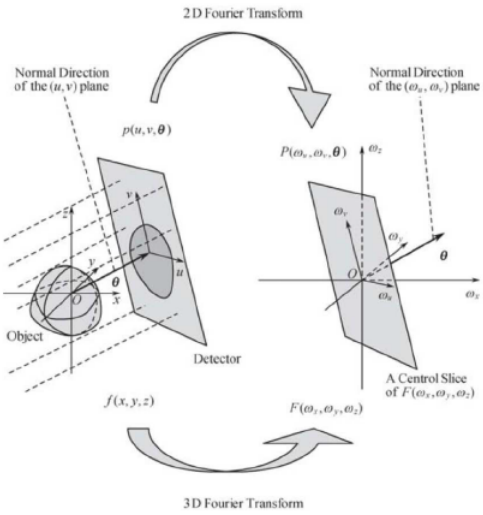
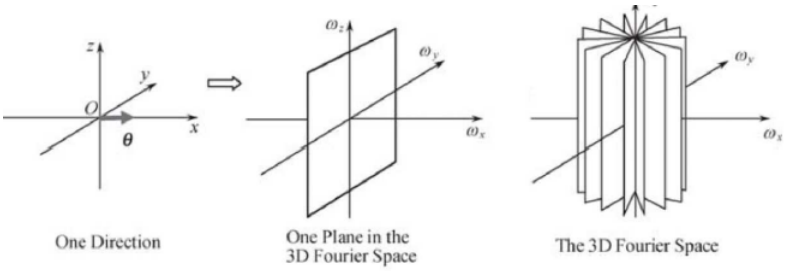
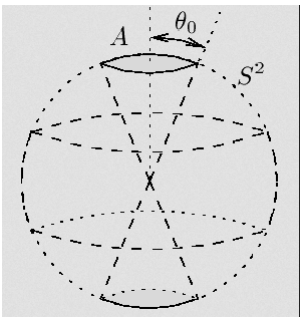



Abbildung 4.23.: Central-Slice-Theorem for the 3D-Case




#### 4.9.1.2. Orlov's condition

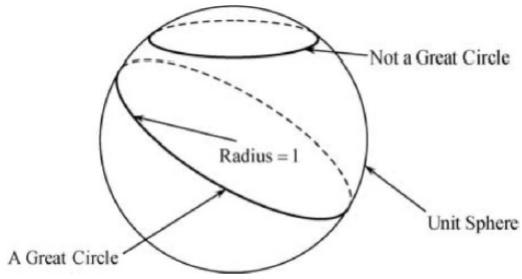
As we sketched above, what happens if the projection rays are not perpendicular to the rotation axis, how does the sampling of the Fourier space looks like? Lets consider that we have an angle which is not 90° between the projection lines and the rotation axis, then we will get missing Data (see figure 4.9.1 middle and left). We will miss two Cones (like shown in the figure) in the Fourier space: this will lead




to artefacts in the reconstruction.

Orlovs condition makes sure that we sample the whole Fourierspace and don't miss data:

> A Complete data set can be obtained, if every great circle intersects the trajectory




If we move our X-Ray source on a great circle on a sphere (the shown unit sphere) we can sample the Fourierspace complete. A few Examples: The first three examples

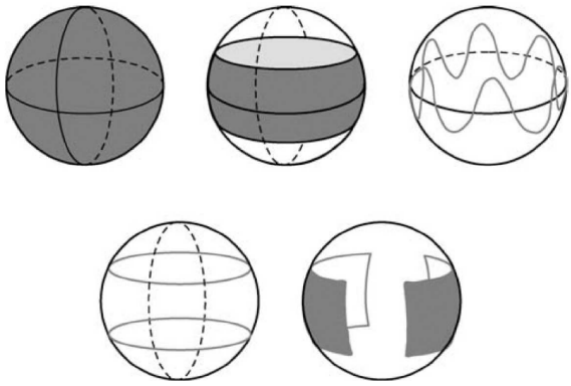


Abbildung 4.24.: Examples for trajectories

fulfil orlovs condition, all great circle you can draw on this sphere will intersects your trajectory. The two example below violate orlovs condition, we can draw great circles which will not intersect. So in the two examples below we cannot have a proper sampling of the Fourierspace !

#### 4.9.1.3. Parallel Line Backprojection-then-Filtering Algorithm

If we perform the Backprojection first then the following steps describe the algorithm:

- If the original image is $f(x, y, z)$ and the Backprojected image is $b(x, y, z)$, then:

$$b = f * * * h$$

where $h$ is the Point-Spread-Function and $***$ denotes the 3D convolution.

- if we have the trajectory like in Figure 4.24 in the left upper case is shown then the PSF is:
$$h(x,y,z) = \frac{1}{x^2+y^2+z^2} = \frac{1}{r^2}$$

- the Fourier transform looks then like:
$$B(\omega_x,\omega_y,\omega_z) = F(\omega_x,\omega_y,\omega_z)H(\omega_x,\omega_y,\omega_z)$$
$$H(\omega_x,\omega_y,\omega_z) = \frac{\pi}{\sqrt{\omega_x^2+\omega_y^2+\omega_z^2}}$$

- to get the original Function just devide through the PSF and then apply the inverse Fourier transform
$$F(\omega_x,\omega_y,\omega_z) = B(\omega_x,\omega_y,\omega_z)\frac{\sqrt{\omega_x^2+\omega_y^2+\omega_z^2}}{\pi}$$

- In general, 3D line-integral data are measured with heavy redundancy. therefore, the image reconstruction algorithm is not unique because you can always weight redundant data differently.

#### 4.9.1.4. Filtered-Backprojection

For the filtered Backprojection we can use the idea from the 1D/2D-case. The filtering with the filter-Kernel is described by:

$$g(\boldsymbol{\theta},u,v) = p(\boldsymbol{\theta},u,v) ** h(\boldsymbol{\theta},u,v)$$
$$G(\boldsymbol{\theta},\omega_u,\omega_v) = P(\boldsymbol{\theta},\omega_u,\omega_v)\cdot H(\boldsymbol{\theta},\omega_u,\omega_v)$$

$\boldsymbol{\theta}$ is a 3D-Angle and describes the plan and $u,v$ are the coordinates on this Plane, you can look up the situation in Figure 4.23. The Filter $H_{\boldsymbol{\theta}}(\omega_u,\omega_v)$ is often view-dependent, for the unit-sphere the filter is view-independent.

### 4.9.2. Parallel Plane-Integral Data

Above we described the approach of the slice-by-slice 2D reconstruction to reconstruct 3D Objects.
Let us consider know the reconstruction for 3D-objects completely new without fall back to the 2D-case.
In 3D, the parallel plane-integreal $p(s,\boldsymbol{\theta})$ of an object $f(x,y,z)$ is referred to as the

Radon transform (see Figure 4.25). The Radon transform in 3D is worthwhile to investigate because is has a simple and nice inversion and can be used to solve other related imaging problems. For this study we imagine a 1D detector that is able to

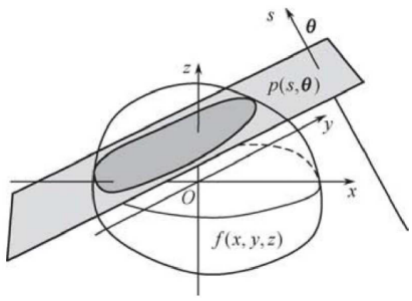
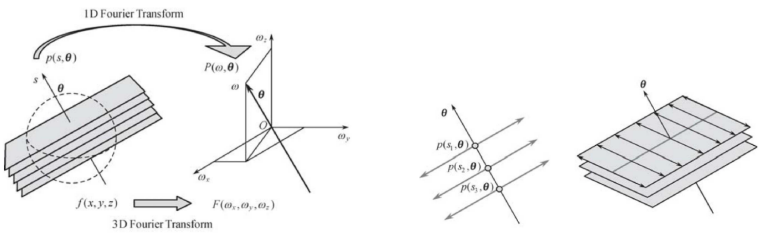



Abbildung 4.25.: In 3D, the plane integral of an object is the Radon transform

measure plane-integrals $p(s, \boldsymbol{\theta})$ [2] with the planes orthogonal to the detector. The detector is aling the direction $\boldsymbol{\theta}$ which is a 2D angle (inner plane rotation + outer plan rotation).

The **central slice theorem** for the radon transform in 3D states that the 1D Fourier trasnfor $P(\omega, \boldsymbol{\theta})$ of the projection $p(s, \boldsymbol{\theta})$ of a 3D function $f(x, y, z)$ is equal to a 1D profile through the origin of the 3D Fourier transform $F(\omega_x, \omega_y, \omega_z)$ of that function which is parallel to the detector (see figure 4.26) So nice, we have a central




(a) In 3D, the plane integral of an object is the Radon transform

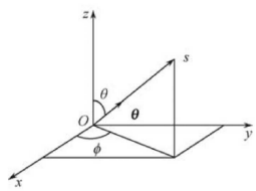
(b) $p(s, \boldsymbol{\theta})$

Abbildung 4.26.: In 3D, the plane integral of an object is the Radon transform

slice theorem for the 3D case ! But we do not have a 1D detector which can measure plane-integrals. But we can handle this problem by using a 2D-detector which can measures the line Integrals in the plane, and then integrate over these line-integrals to get the plane-integral.

**take care:** that the intersection line of the plane (we want to integrate) and the detector must not correspond with a pixel-row on the detector ! So we have to compute the intersection-line of the plane and the detector to know which measured line-integrals we have to integrate over.

[2] in the real case, the detector is 2D and we integrate the line Integrals such that we get the plan integral




The Backprojection is different, eachs point on the 1D $p(s, \boldsymbol{\theta})$ correspond to a plane, so a point has to be backprojected as a plane in 3D.
The inversion Formular of the Radon transform is given by the second order derivative of $p(s, \boldsymbol{\theta})$ with respect to s (this step is called filtering !) and then backproject the filtered data to the 3D image array:

$$f(x, y, z) = \frac{1}{8\pi^2} \int\int_{4\pi} \frac{\partial p(s, \boldsymbol{\theta})}{\partial s^2}|_{s=\boldsymbol{x}\cdot\boldsymbol{\theta}} \quad \sin\theta \; d\theta \; d\phi \qquad \text{with} \quad \boldsymbol{\theta} = \begin{pmatrix} \sin\theta\cos\phi \\ \sin\theta\sin\phi \\ \cos\theta \end{pmatrix}$$

the second order derivative is from numerical perspectives on noisy data not a good idea!
which gives us the **Backprojection-Then-Filterin Algorithm:**

- The backprojection of the image is obtained as

$$b(x, y, z) = \int\int_{2\pi} p(s, \boldsymbol{\theta})|_{s=\boldsymbol{x}\cdot\boldsymbol{\theta}} \quad \sin\theta \; d\theta \; d\phi$$

- For the 3d Radon case, the PSF is given as:

$$H(\omega_x, \omega_y, \omega_z) = \frac{1}{\omega_x^2 + \omega_y^2 + \omega_z^2}$$

- Hence the image can be obtained by:

$$F(\omega_x, \omega_y, \omega_z) = B(\omega_x, \omega_y, \omega_z) \cdot (\omega_x^2 + \omega_y^2 + \omega_z^2)$$

- in spatial domain this can be written as:

$$f(x, y, z) = \Delta b(x, y, z) = \frac{\partial^2 b(x, y, z)}{\partial x^2} + \frac{\partial^2 b(x, y, z)}{\partial y^2} + \frac{\partial^2 b(x, y, z)}{\partial z^2}$$

### 4.9.3. Cone Beam Geometry

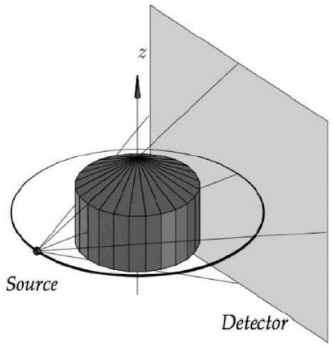



Abbildung 4.27.: Cone Beam Geometry

This is a very popular geometry. A flat pane oder multi-row detector can detect considerably more data at a time.
As in the 2D/1D case there is not directly a central slice theorem for cone beam geometry, we can think of the type of resampling like in fan-beam geometry with the plane-integrals. The Projection and backprojection can be described using projection matrices.

#### 4.9.3.1. Tuy's condition - complete Data

first lets look at Tuy's condition which describe how we have to capture images with the cone beam geometry to sample the fourierspace complete:

> Tuy's condition say: Every plane that intersects the object of interest must contain a cone beam focal point !

We can see, only a rotation with fan beam geometry around the object does **not** full fill tuy's condition, so we don't sample the Fourierspace properly. Only the slice in the plan with the focal point will reconstruct properly. A solution would be a

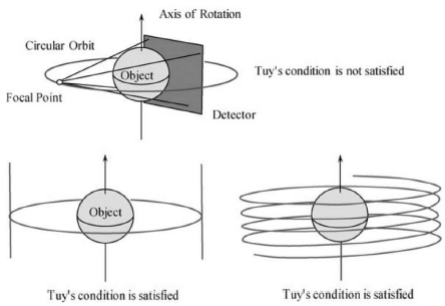



Abbildung 4.28.: Tuy's condition

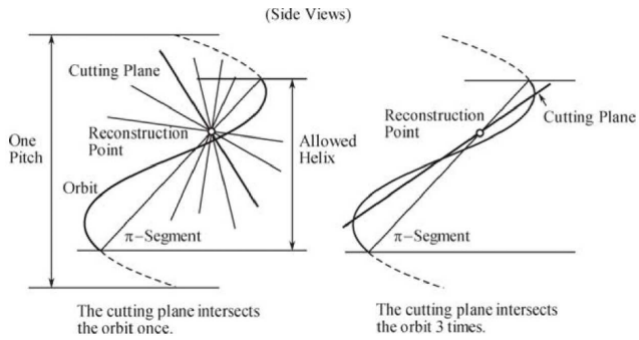



Abbildung 4.29.: The $\pi$-segment defines a section of helix. A cutting plane is a plane passing through the reconstruction point. The cutting plane either intersect the section helix once or three times

complete rotation and afterwards include a line shift (lower left picture) or use a helical trajectory.
Nevertheless also the cone-beam geometry have redundant data which we have to take in consideration in the reconstruction process with the proper weighting of the measurements. In the helix case there exsists point which are measured two or some times even three times:

#### 4.9.3.2. Feldkamps Algorithm

This is the algorithm which is very often used in standard CT-setup with 2D-Detectors. It is basically a filtered-backprojection with a proper weighting of the projection rays. The algorithm is based on a circular trajectory (which means we have incomplete data !). The algorithm it is based on the fan beam algorithm with appropriate cosine weights. For the central Ray is just the pure fan-beam reconstruction algorithm and for all others we have to extrapolate the idea of fan beam to another angle. The geometry for Feldkamps algorithm s shown: The short words the algorithm works like: We take the measurements and do the weighting. Convolve the signal with a filter (ramp-filter) and at the end we just integrate up over all the volume.

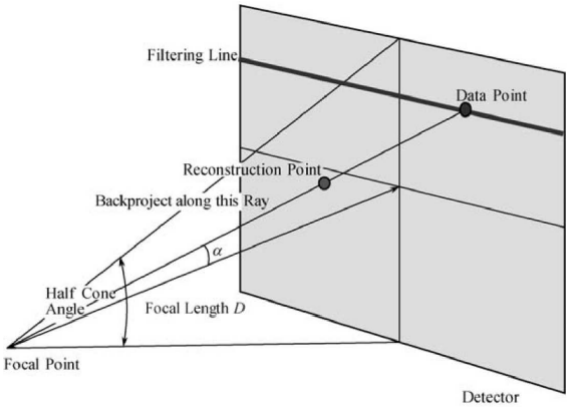


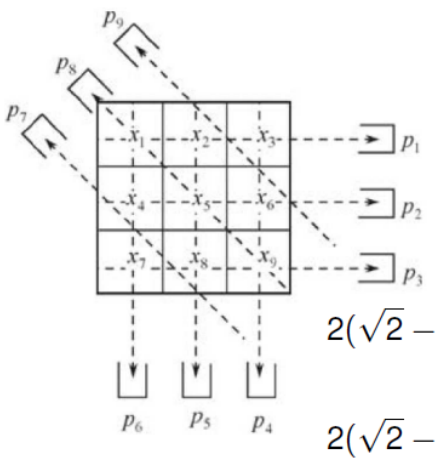

Abbildung 4.30.: Geometry for Feldkamps Algorithm

## 4.10. ART - Algebraic Reconstruction Technique

Since the computational power especially due to GPU's rises extremely there is a renaissance of the algebraic approach. First CT-scanner used this, but then the number of linear equations which need to be solved explode and the computational power was not enough to deal with this (short rethinking: for a Volume with $512 \times 512 \times 512$ we have $2^{27}$ unknowns). That's why the Backprojection was and is still the „working horse"of CT-Reconstruction.

### 4.10.1. Linear Equations

In the Figure we see, how the linear equations generally can build up, we use the length of the ray trough the Voxel as a weighting factor how much this ray contributes to the value we end up with: this system can be rewritten in Vector-Matrix-




$$
\begin{aligned}
x_1 + x_2 + x_3 &= p_1 \\
x_4 + x_5 + x_6 &= p_2 \\
x_7 + x_8 + x_9 &= p_3 \\
x_3 + x_6 + x_9 &= p_4 \\
x_2 + x_5 + x_8 &= p_5 \\
x_1 + x_4 + x_7 &= p_6 \\
2(\sqrt{2}-1)x_4 + (2-\sqrt{2})x_4 + 2(\sqrt{2}-1)x_8 &= p_7 \\
(\sqrt{2})x_1 + (\sqrt{2})x_5 + (\sqrt{2})x_9 &= p_8 \\
2(\sqrt{2}-1)x_2 + (2-\sqrt{2})x_3 + 2(\sqrt{2}-1)x_6 &= p_9
\end{aligned}
$$

Abbildung 4.31.: An example with 9 unknowns and 9 measurements

Notation:

$$\mathbf{A}\vec{x} = \vec{p} \qquad \Leftrightarrow \qquad \vec{x} = \mathbf{A}^{-1}\vec{p} \qquad \Leftrightarrow \qquad \vec{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\vec{p}$$
$$\vec{x} = (x_1, x_2, ..., x_p)^T \qquad \vec{p} = (p_1, p_2, ..., p_9)^T$$

where $\mathbf{A}$ is the system matrix with elements $a_[ij]$, each $a_[ij]$ describes the contribution of each voxel to each ray.
We can solve with the inverse of $\mathbf{A}$, but the problem is most of the time we have more unknowns than equations. And then the Matrix can't inverted, this could be solved if we compute the pseudo-inverse (which is the close form solution of the least square estimator of: $|\mathbf{A}\vec{x} - \vec{p}|_2^2 \to min$).
We look for a Solution that does not require the inversion of $\mathbf{A}$ or a product of $\mathbf{A}$ (obviously because of the high dimension of the Matrix).
**Simple Example:** there can be used a simple example to sketch the idea:
think about a non-trivial but very easy object we want to reconstruct: An object with 2 Voxels, we have two unknowns $x_1, x_2$ and two measurements $p_1, p_2$. Then we have to solve these two linear equations:

$$a_{11}\vec{x}_1 + a_{12}\vec{x}_2 = \vec{p}_1$$
$$a_{21}\vec{x}_1 + a_{22}\vec{x}_2 = \vec{p}_1$$

we can write this in the matrix notation:

$$\mathbf{A}\vec{x} = \vec{p}$$

Systemmatrix: $\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$ measurement: $\vec{p} = \begin{pmatrix} p_1 \\ p_2 \end{pmatrix}$ Volume: $\vec{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$

we can rewrite the Matrix $\mathbf{A}$ in terms of row-vectors $(a_1^T, a_2^T)$:

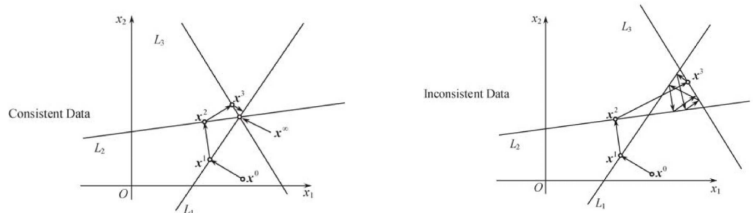
$$\begin{pmatrix} a_1^T \\ a_2^T \end{pmatrix} \vec{x} = \vec{p}$$

then we can rewrite our initial equations:

$$a_{11}\vec{x}_1 + a_{12}\vec{x}_2 = \vec{p}_1 \Longleftrightarrow a_1^T\vec{x} = \vec{p}_1$$
$$a_{21}\vec{x}_1 + a_{22}\vec{x}_2 = \vec{p}_1 \Longleftrightarrow a_2^T\vec{x} = \vec{p}_2$$

and this is interesting, because we can read the right hand side as the Hesse-normal form of a linear manifold where $a_1^T$ is the normal vector of the line and $p_1$ is the

distance of this line to the origin !
We know looking for a $\vec{x}$ that fulfil both equation, and this is the intersection point of both lines!
We can find the intersection of both lines with an simple iterative scheme: Start with a point $x^0$ (e.g. we get this from a backprojection) and then compute the orthogonal projection of this point to one of the lines (this is then called $x^1$). Then project the point $x^1$ orthogonally to the other line ($x^2$). Repeat this till you find the intersection point(see left Figure 4.32). The more Orthogonal the lines are the faster the scheme




(a) Example with two Voxels

(b) Example with two Voxels and incosistent Data

Abbildung 4.32.: Example of the iterative scheme to find the intersection

converge !
Lets take a look at the iteration scheme in mathematical notation. We have a point $\vec{p}$ and want to compute $\vec{p}\,'$ which is the orthogonally projection on the straight line $\vec{n}^T\vec{x} = d$ :

$$
\begin{aligned}
(I) \quad \vec{p}\,' - \vec{p} &= \lambda\vec{n} && \text{(orthogonality criterion)}\\
(II) \quad \vec{n}^T\vec{p}\,' &= d && (\vec{p}\,' \text{ is the projection so it fulfil the line equation})\\
\vec{n}^T(\lambda\vec{n} + \vec{p}) &= d && \text{(put I into II and solve to } \lambda)\\
&= \lambda\vec{n}^T\vec{n} + \vec{n}^T\vec{p}\\
&= \lambda\,\|\vec{n}\|_2^2 + \vec{n}^T\vec{p} && (\vec{n}^T\vec{n} = \|\vec{n}\|_2^2)\\
\lambda &= \frac{d - \vec{n}^T\vec{p}}{\|\vec{n}\|_2^2}\\
& && \text{(now put } \lambda \text{ into (I) and solve to } \vec{p}\,')\\
\vec{p}\,' - \vec{p} &= \frac{d - \vec{n}^T\vec{p}}{\|\vec{n}\|_2^2}\vec{n}\\
\vec{p}\,' &= \vec{p} + \frac{d - \vec{n}^T\vec{p}}{\|\vec{n}\|_2^2}\vec{n}
\end{aligned}
$$

and this is the iterative scheme, we can lift it up to our Matrix notation, which is then the originally method published by Kaczmarz. For each pixel $p_i$ and each row $\mathbf{A}_i$ of $\mathbf{A}$ perform the following update:

$$\vec{x}^{k+1} = \vec{x}^k + \frac{p_i - \mathbf{A}_i \vec{x}^k}{\mathbf{A}_i \mathbf{A}_i^T} \mathbf{A}_i^T$$

there are some nice facts:

- is has been shown that the iterative scheme converges to the solution, if there exists a unique solution
- the angle between hyper planes influences the rate of convergence to the solution
- if hyper planes are orthogonal to each other, it is obvious that the method converges rapidly
- Orthogonalization methods applied in advance to iterations will improve convergence. Or alternative: careful selection of sequence of projections !
- Overdetermined systems and noise: no unique solution and oscillation
- a main drawback of ART is the slow convergence !

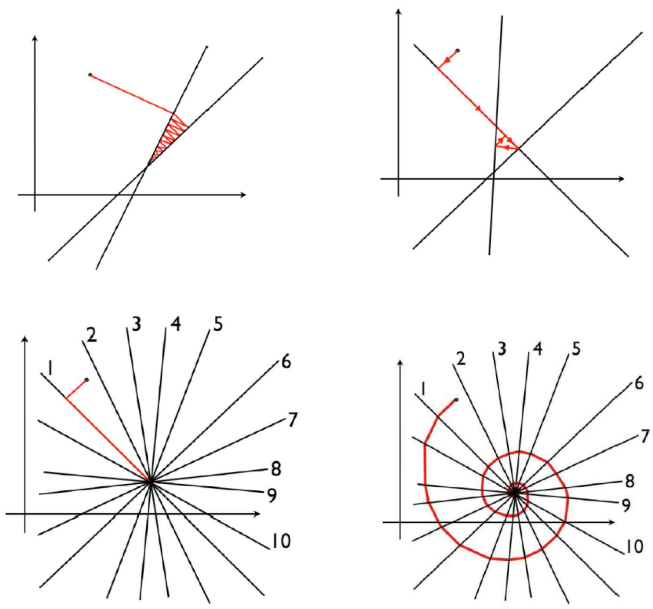



Abbildung 4.33.: Example convergence of the iterative scheme

there exist different method which improve the convergence speed:

**SART:** The idea is to compute parallel all different Projections of the point to the different lines and then compute the mean between all projected points and start with again. This convergence also, but faster.

**Ordered Subsets:** Employs knowledge on the acquisition sequence, select subsets of equations that are most likely orthogonal to each other

**Towards more Realism:** instead of taking the length from the ray through the Voxel as a weight try to find more realistic approximations to build up the system matrix. Include scattering, noise, etc.

### 4.10.2. Gradient Descent Algorithm

Formulate the reconstruction problem as an optimization Problem, find an optimum via peak condition. This enables the use of various methods that are common in optimization.
We can write the objective function as:

$$\chi(\vec{x}) = \|\mathbf{A}\vec{x} - \vec{p}\|_2^2 = (\mathbf{A}\vec{x} - \vec{p})^T(\mathbf{A}\vec{x} - \vec{p})$$

for this we can compute either the close form solution with the pseudo inverse:

$$\vec{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\vec{p}$$

or we reformulate the optimization problem as an iterative procedure:

$$\vec{x}^{k+1} = \vec{x}^k - \lambda\Delta$$

so in each step we want to go one step towards the minimum, i.e. opposite gradient direction $-\lambda\Delta$. At the end this looks like:

$$\vec{x}^{k+1} = \vec{x}^k - \lambda(\underbrace{2\mathbf{A}^T(\mathbf{A}\vec{x} - \vec{p})}_{\text{gradient}}) \Longleftrightarrow \vec{x}^{k+1} = \vec{x}^k + \lambda(\mathbf{A}^T(\vec{p} - \mathbf{A}\vec{x}))$$

### 4.10.3. Maximum-Likelihood Expectation-Maximization Methods

Formulate the objective function as likelihood function. Find the optimum of the likelihood function, i.e. the most likely solution. This can be solved with the Expectation Maximization Algorithm.
A very basic example to show how you can reformulate a optimization problem into the language of statistics:

- if we want to compute $ax - b = 0$ we can setup the minimization problem: $\sum_i \|ax_i - b\|^2 \to \min$ for various $x$ and minimize this with respect to a and b.
- if you look at the term: $(ax_i - b)(ax_i - b)$ you can see that you can put the inverse of a covariance matrix between these two and then we have the exponent of a Gaussian:

  $$e^{-(ax_i-b)\Sigma^{-1}(ax_i-b)}$$

  and know we can maximize the value of the Gaussian PDF with respect to a and b:

  $$\prod_i e^{-(ax_i-b)\Sigma^{-1}(ax_i-b)} \to \max$$

and this is the idea we want to apply to the reconstruction Problem.
Lets consider for the following an emission tomography problem. The probability density function for a random variable r to emit a certain amount of energy follows the Poisson distribution:

$$P(r|\lambda) = e^{-\lambda}\frac{\lambda^r}{r!}$$

the expected value of this random variable is $\lambda$ the observed value at each detector bin is:

$$p_i = \sum_j c_{ij}$$

where $\lambda_{ij} = E(c_{ij}) = a_{ij}x_j$
**parenthesis** the Expectation definition is: $E[x] = \int x \cdot p(x)dx$
So a detector pixel $p$ is a sum of random variables.
To summarize we can say we maximize the function for the given observations !

### 4.10.4. Regularized Reconstruction

Again the problem what we consider is $\mathbf{A}\vec{x} = \vec{p} \Rightarrow \|\mathbf{A}\vec{x} - \vec{p}\|_n^2 \to \min$, where $n$ is some norm.
But we can say that we want to introduce a regularizer into the optimization problem:

$$\|\mathbf{A}\vec{x} - \vec{p}\|_d + \lambda \cdot R(\vec{x}) \to \min$$

for example $R(\vec{x})$ could be smoothness $\|\nabla\vec{x}\| \to$ small, so we optimize our problem with respect the smoothness which means the difference between the neighbours

should be small.

> We introduce additional information into the reconstruction precess to enforce a certain solution

with this we can incorporate prior knowledge about the Volume we want to reconstruct.
**TV-Norm** The Total Variation Norm is popular combination of a sparsifying transform with $L_1$ norm, the sparsifying transform is the gradient image. But it can be shown that reconstruction with TV-regularization is identical to FBP reconstruction with subsequent non-linear edge preserving filtering !

## 4.11. Bilateral Filter

# 5. Exercise1

## 1 SVD

### 1a

Create a Matrix: $\mathbf{A} = \begin{bmatrix} 11 & 10 & 14 \\ 12 & 11 & -13 \\ 14 & 13 & -66 \end{bmatrix}$ Check the determinant of this matrix. Compute the inverse matrix of **A** without using MATLAB command **inv**:

```
A = [11 10 14 ;
     12 11 -13;
     14 13 -66];

 [U S V] = svd(A);%make the SVDecomposition

 Ainverse = V * S^(-1) * U' %compute the pseudo-inverse . If the matrix is invertible
                            %then the pseudo-invers will be the inverse
```

Compare the result to **inv(A)** → it's the same
How do we get the condition number and what does the condition number express?
The condition Number can be computed with SVD: divide the largest number of **S** by the smallest. (Hint: the first divided by the last element on the diagonal of**S**).

```
 kappaA = S(1,1) / S(length(S),length(S))
```

- a condition number near to 1 means the matrix is well-conditioned.
- a condition number which far away from 1 means the matrix is.

### 1b

With the command **eigshow** you can see how the Matrix affect the unit-ball (for 2D). Because you use the uni-vectors for this you can see in a way the geometrical representation of the Matrix.

### 1c

If you set the threshold $\epsilon = 10^{-3}$, you get a rank deficiency. How can you get the null space and the rank of the Matrix $\mathbf{A}$? The null space determine all vectors which fulfil the following equation (the zero-vector is the trivial case): $\mathbf{A}\vec{x} = 0$ You can use SVD to determine the null space. Take the collumn-vector of $\mathbf{V}$ which is related to the zero-entries on the diagonal of $\mathbf{S}$. In our example case, the last diagonal entry will be zero after you applied the Threshold. Which means the last collumn of $\mathbf{V}$ is the null space for Matrix $\mathbf{A}$.

```
kernelA = V(:,3)
```

The rank of a Matrix can also be determine with SVD. Just count the non-zero entries in the diagonal of $\mathbf{S}$:

```
%The rank of the matrix are all non-zero entries on the diagonal of S
rankA = nnz(S)
%The range are the related collumns in U to the non-zero entries in S
rangeA = U(:,1:2)
```

The range of the Matrix $\mathbf{A}$ is then the collumn-vectors of $\mathbf{U}$ which are related to the non-zero entries of $\mathbf{S}$.

## 1.1

Show that a variation of elements of $\vec{b}$ by 0.1% implies a change in $\vec{x}$ by 241%.

```
b = [1.001;0.999;1.001];
x = Ainverse * b
%Console:
x =
   -0.6830
    0.8430
    0.0060

%changing b by 0.1%
b = [1.002001;0.999;1.001];
x = Ainverse * b
%Console:
x =
   -1.2406
    1.4536
    0.0080
```

# A. Anhang