DMIP Summary

Untertitel

Art der Arbeit

im Fachgebiet Fachgebiet

vorgelegt von: Christopher Syben

Studienbereich: Studienbereich

Abstract

Abstract

Inhaltsverzeichnis

Abkürzungsverzeichnis

Abk. Abkürzung

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

bla (Bäni 2005)

1.1. Motivation

1.2. Ziel der Arbeit

1.3. Aufbau der Arbeit

2. Math

2.1. Matricies

Nullspace

The nullspace of a matrix is:

$$\mathbf{A}\vec{x} = \vec{0} \tag{2.1}$$

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.

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ \\ a_{21} & a_{22} \end{bmatrix} = (\vec{a_1}, \vec{a_2}) \tag{2.2}$$

pseudo-inverse

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

2.2. 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)

2.3. Fourier Transform

2.3.1. Symmetry Property of Fourier Transform

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

$$F(\xi) = \bar{F}(N - \xi) \tag{2.3}$$

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

2.3.2. Dirac's δ-function

The δ-function is defined as:

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

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

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

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

2.4. Statistics

2.4.1. 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:

$$KL(p, q) = \sum_{i=1}^{n} p(x_i) \log \frac{p(x_i)}{q(x_i)} \tag{2.4}$$

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 $p \to q$

2.4.2. 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.

2.4.3. 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**

2.4.4. Marginalization - Hidden Random Variables

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

$$\int\int\int p(X, Y, Z) dX dY dZ = 1 \tag{2.5}$$

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

$$\int \int p(X, Y, Z) dY dZ = p(X) \tag{2.6}$$

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

2.5. 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:

$$\begin{pmatrix} x \\ y \end{pmatrix} \rightarrow w \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \qquad \text{where } w \in \mathbb{R} \tag{2.7}$$

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

2.5.1. Rotation and Translation

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

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \mathbf{R} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + t \tag{2.8}$$

if you use homogeneous coordinates you can incorporate the translation:

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} R_{11} & R_{12} & R_{13} & t_x \\ R_{21} & R_{22} & R_{23} & t_y \\ R_{31} & R_{32} & R_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{2.9}$$

3. PreProcessing

3.1. Disortion & Interpolation

3.2. Detectors

The Image quality is defined by three major components.

- Noise
- spatial Resolution
- contrast resolution

3.2.1. DQE - Measurement

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

3.3. Defect Pixel Interpolation

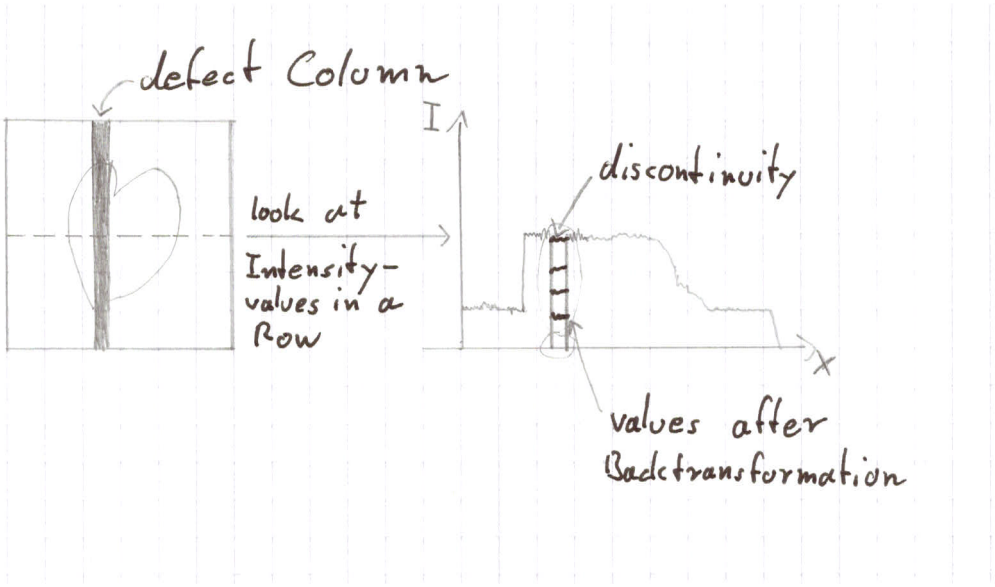
In Interpolation you hit the sampling points exactly, not like regression where you fit a line with the smallest error. Or like extrapolation, where you try to find a sampling point outside your interval of sampling Points.

3.3.1. 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

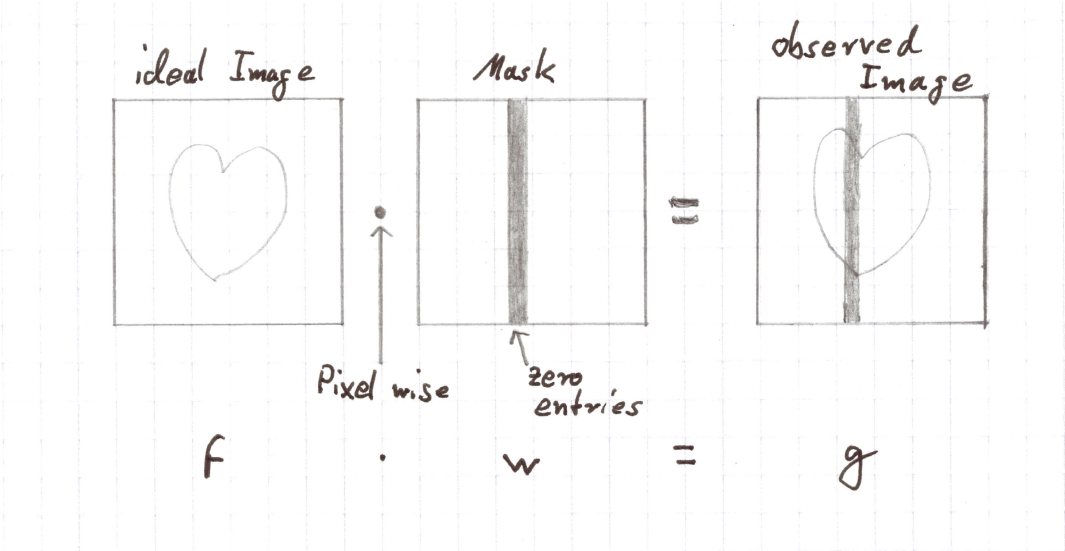


the discontinuity leads to a Signal which have high frequencies and is not band-limited any more ! So the Fourier transform run into the problem that the Fourier transform will not achieve zero values for a certain threshold ξ.
So take the Fourier transform of the signal with the defect and cut off all 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 δ-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 δ-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:

$$\Delta_\epsilon = \frac{1}{N}\sum_{n=0}^{N-1}(g(s) - w(n)\cdot f(n))^2 \tag{3.1}$$

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 Δ_ϵ using (8)	
	IF	error Δ_ϵ above a threshold
	THEN Compute error spectrum $G^i(k)$, increment i	
UNTIL error are below a threshold		
Compute inverse FT of $\hat{F}^i(k)$		

3.4. 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

3.4.1. 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

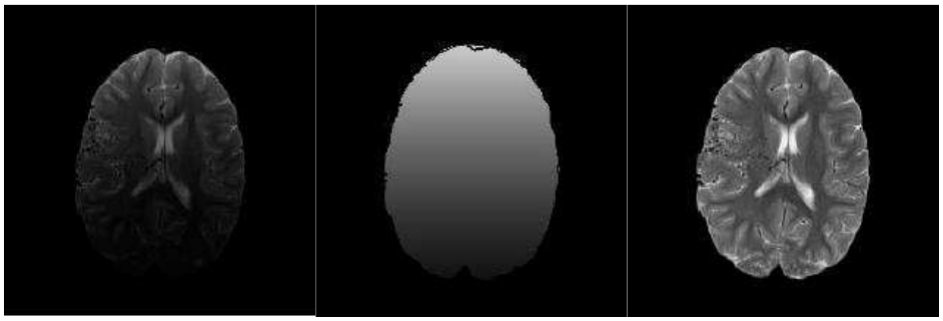


Abbildung 3.1.: Original MR Image(left),gain field (middle), restored image (right)

- **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:

$$g_{ij} = f_{ij} \cdot b_{ij} + n_{ij} \tag{3.2}$$

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:

$$H_{k,l} = 1 - \beta \cdot e^{-\frac{k^2+l^2}{2\sigma^2}} \tag{3.3}$$

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

$$f = \mathrm{FT}^{-1}\Big\{G_{k,l} \cdot H_{k,l}\Big\} \tag{3.4}$$

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):

$$[h_{i,j}] = \mathrm{LPF}([\log g_{i,j}]) \tag{3.5}$$

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

$$[\log f_{i,j}] = [\log g_{i,j}] - [h_{i,j}] + \mu \tag{3.6}$$

  where μ 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 3.1 that not true:

$$f_{i,j} = g_{i,j} \cdot \frac{\mu}{\mu_{i,j}} \tag{3.7}$$

where μ 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:

$$J(x_1, x_2, ..., x_n) = \sum_{i=1}^{N_c} \sum_{k=1}^{n} a_{i,k}^d ||x_k - c_i||^2 \tag{3.8}$$

- 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:

$$\text{minimize } \sum_{i=1}^{N_c} \sum_{k=1}^{n} a_{i,k}^d ||x_k - c_i||^2 \tag{3.9}$$

$$\text{subject to } \sum_{i=1}^{N_c} a_{i,k} = 1 \tag{3.10}$$

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: 2.4.3):

$$J(x_1, x_2, ..., x_n) = \underbrace{\sum_{i=1}^{N_c}\sum_{k=1}^{n} a_{i,k}^d ||x_k - c_i||^2}_{\text{data Term}} + \underbrace{\sum_{i=1}^{N_c}\sum_{k=1}^{n} \frac{\lambda}{\#\mathcal{N}_k} a_{i,k}^d \sum_{x_r \in \mathcal{N}_k} ||x_r - c_i||^2}_{\begin{array}{c}\text{data term whith prior;}\\ \text{incorporates neighbourhood}\end{array}} \quad (3.11)$$

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 β:

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

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

with this we try to compute the bias field β_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:

$$J_R = \sum_{i=1}^{N_c}\sum_{k=1}^{n}\left(a_{i,k}^d D_{i,k} + \frac{\lambda}{\#\mathcal{N}_k} a_{i,k}^d E_{i,k}\right) + \sum_{k=1}^{n} \eta_k (1 - \sum_{j=1}^{N_c} a_{j,k}) \quad (3.12)$$

$$\text{where } D_{i,k} = \|y_k - \beta_k - c_i\|^2 \quad (3.13)$$

$$E_{i,k} = \sum_{(y_r - \beta_r) \in (N)_k} \|y_r - \beta_r - c_i\|^2 \quad (3.14)$$

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

$$\hat{a}_{i,k} = \frac{1}{\sum_{j=1}^{N_c} \left(\frac{\#\mathcal{N}_k D_{i,k} + \lambda E_{i,k}}{\#\mathcal{N}_k D_{j,k} + \lambda E_{j,k}} \right)^{\frac{1}{d-1}}} \tag{3.15}$$

..leads to Cluster Prototype Update:

$$\hat{c}_i = \frac{\sum_{k=1}^{n} a_{i,k}^d \left((y_k - \beta_k) + \frac{\lambda}{\#\mathcal{N}_k} \sum_{y_r \in \mathcal{N}_k} (y_r - \beta_r) \right)}{(1+\lambda) \sum_{k=1}^{n} a_{i,k}^d} \tag{3.16}$$

and to the Bias Field Estimator:

$$\hat{\beta}_k = y_k - \frac{\sum_{i=1}^{N_c} a_{i,k}^d \cdot c_i}{\sum_{i=1}^{N_c} a_{i,k}^d} \tag{3.17}$$

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 2.4.4: 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 Γ are normally distributed:

$$p(x_{i,j}|\Gamma;\beta_{i,j}) = \mathcal{N}(x_{i,j};\mu_\Gamma + \beta_{i,j},\Sigma_\Gamma) \tag{3.18}$$

$$= \frac{1}{\sqrt{|2\pi\Sigma_\Gamma|}} e^{-\frac{1}{2}(x_{i,j}-\mu_\Gamma-\beta_{i,j})^T\Sigma_\Gamma^{-1}(x_{i,j}-\mu_\Gamma-\beta_{i,j})} \tag{3.19}$$

  where

  - $x_{i,j}$: observed log intensity
  - Σ_Γ: covariance matrix of tissue class Γ
  - $\beta_{i,j}$: bias at point (i,j)
  - μ_Σ mean log intensity of tissue class Σ

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

$$p(\Gamma), \text{for} \qquad \Gamma = 1, 2, ..., N \tag{3.20}$$

- prior density of bias field:

$$p(\beta_{i,j} = \mathcal{N}(\beta_{i,j};0,\Sigma_\beta)) \tag{3.21}$$

  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 Γ by marginalization:

$$p(x_{i,j};\beta_{i,j}) = \sum_{\Gamma=1}^{N} p(\Gamma)p(x_{i,j}|\Gamma;\beta_{i,j}) \tag{3.22}$$

  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 $\mathbf{x} = [x_{i,j}]$:

$$\hat{\beta} = \arg\max_{\beta} p(\beta|\mathbf{x}) = \arg\max_{\beta}(\log p(\beta) + \log p(\mathbf{x}|\beta)) \tag{3.23}$$

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 !):

$$p(\mathbf{x}|\beta) = \prod_{i,j} p(x_{i,j}|\beta_{i,j}) = \prod_{i,j} \sum_{\Gamma=1}^{N} p(\Gamma)p(x_{i,j}|\Gamma, \beta_{i,j}) \tag{3.24}$$

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 β
- covariance Σ_Γ
- priors $p(\Gamma)$
- means μ_Γ and covariances σ_Γ

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:

$$\hat{\Gamma} = \arg\max_{\Gamma} p(\Gamma|\vec{x}_{i,j}; \beta_{i,j}) \tag{3.25}$$

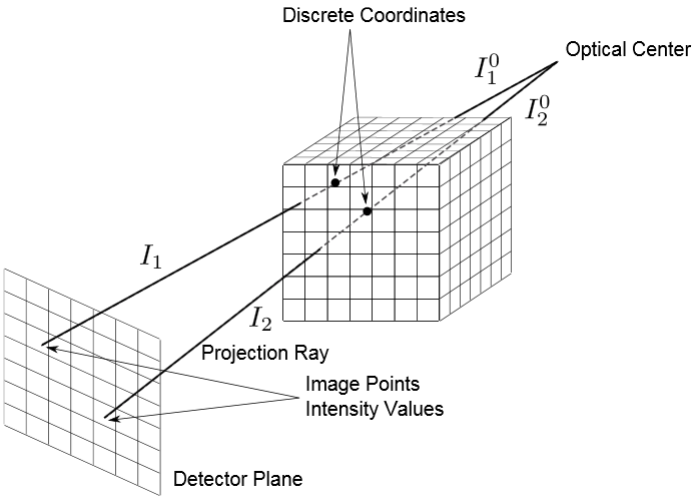
$$= \arg\max_{\Gamma}(\log p(\Gamma) + \log p(x_{i,j}|\Gamma; \beta_{i,j})) \tag{3.26}$$

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

$$\beta_{i,j} = \sum_{k=0}^{M} \theta_k \phi_k(i,j) \tag{3.27}$$

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

4. Projection and Homogeneous Coordinates



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

4.0.2. Projections

There existing different Projection Models:

- Orthographic projection

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \rightarrow \begin{pmatrix} x \\ y \end{pmatrix} \tag{4.1}$$

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

- Weak perspective projection

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \rightarrow \begin{pmatrix} k \cdot x \\ k \cdot y \end{pmatrix} \tag{4.2}$$

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

- Perspective projection

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \rightarrow \begin{pmatrix} f \cdot x/z \\ f \cdot y/f \end{pmatrix} \tag{4.3}$$

  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:

$$\tilde{\mathbf{P}} = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \qquad \tilde{\mathbf{P}}' = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tilde{\mathbf{P}} \tag{4.4}$$

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

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} x \\ y \\ y \end{pmatrix} \Rightarrow \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{4.5}$$

- Perspective Projection in homogeneous coordinates is defined by:

$$\tilde{\mathbf{P}} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} f \cdot \frac{x}{z} \\ f \cdot \frac{y}{z} \end{pmatrix} \rightarrow \begin{pmatrix} fx \\ fy \\ z \end{pmatrix} \qquad \tilde{\mathbf{P}}' = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tilde{\mathbf{P}} \tag{4.6}$$

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

$$\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} x \\ y \\ z \end{pmatrix} \Rightarrow \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{4.7}$$

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

4.1. 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 (2.5.1). Then it is just a matrix multiplication from the left side:

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \mathbf{D} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{4.8}$$

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

4.2. 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

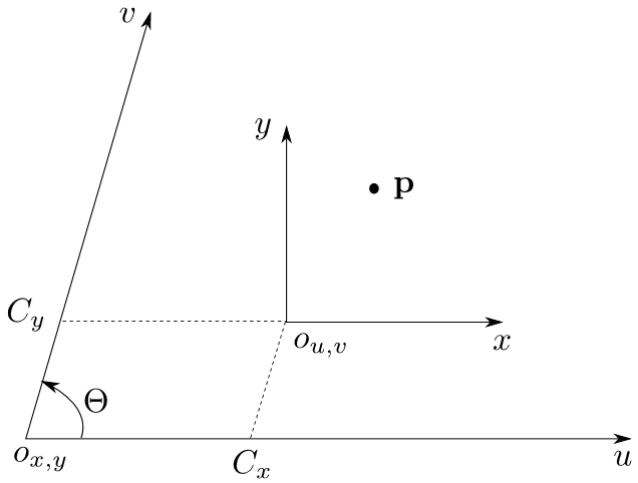


Abbildung 4.1.: $(u; v)$ detector and $(x; y)$ image coordinate system

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

$$\vec{x} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \curvearrowright \begin{pmatrix} \frac{1}{k_x} \\ 0 \end{pmatrix} \qquad \vec{y} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \curvearrowright \begin{pmatrix} \frac{1}{k_y} \cos\theta \\ \frac{1}{k_y} \sin\theta \end{pmatrix} \tag{4.9}$$

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

$$\mathbf{T} = \begin{pmatrix} \frac{1}{k_x} & \frac{1}{k_y} \cos\theta \\ 0 & \frac{1}{k_y} \sin\theta \end{pmatrix}^{-1} = \begin{pmatrix} k_x & -k_x \frac{\cos\theta}{\sin\theta} \\ 0 & \frac{k_y}{\sin\theta} \end{pmatrix} \tag{4.10}$$

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

$$\mathbf{K} = \begin{pmatrix} T_{11} & T_{12} & -c_x \\ T_{21} & T_{22} & -c_y \\ 0 & 0 & 1 \end{pmatrix} \tag{4.11}$$

4.3. Complete Projection

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

$$\rho\tilde{q} = \mathbf{P} \cdot \tilde{p} = \mathbf{K} \cdot \mathbf{P}_{\text{proj}} \cdot \mathbf{D} \cdot \tilde{p} \tag{4.12}$$

4.4. Calibration

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

$$x_i = \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix}, \begin{pmatrix} x_2 \\ y_2 \\ z_2 \end{pmatrix}, ..., \begin{pmatrix} x_n \\ y_n \\ z_n \end{pmatrix} \in \mathbb{R}^3 \tag{4.13}$$

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

$$\begin{pmatrix} u_1 \\ v_1 \end{pmatrix}, \begin{pmatrix} u_2 \\ v_2 \end{pmatrix}, ..., \begin{pmatrix} u_n \\ v_n \end{pmatrix} \in \mathbb{R}^2 \tag{4.14}$$

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:

$$\forall i : \begin{pmatrix} w_i \cdot u_i \\ w_i \cdot v_i \\ w_i \end{pmatrix} \hat{=} \mathbf{P} \begin{pmatrix} x_i \\ y_i \\ z_i \\ 1 \end{pmatrix} \Leftrightarrow \begin{pmatrix} r_1^T \\ r_2^T \\ r_3^T \end{pmatrix} \begin{pmatrix} x_i \\ y_i \\ z_i \\ 1 \end{pmatrix} = \begin{pmatrix} \vec{r}_1^T \vec{x}_i \\ \vec{r}_2^T \vec{x}_i \\ \vec{r}_3^T \vec{x}_i \end{pmatrix} \hat{=} \begin{pmatrix} u_i \\ v_i \\ 1 \end{pmatrix} \tag{4.15}$$

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:

$$u_i = \frac{\vec{r}_1^T \vec{x}_i}{\vec{r}_3^T \vec{x}_i} \qquad v_i = \frac{\vec{r}_2^T \vec{x}_i}{\vec{r}_3^T \vec{x}_i} \tag{4.16}$$

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

$$u_i - \frac{\vec{r}_1^T \vec{x}_i}{\vec{r}_3^T \vec{x}_i} \stackrel{!}{=} 0 \qquad v_i - \frac{\vec{r}_2^T \vec{x}_i}{\vec{r}_3^T \vec{x}_i} \stackrel{!}{=} 0 \tag{4.17}$$

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

$$u_i \cdot \vec{r}_3^T \vec{x}_i - \vec{r}_1^T \vec{x}_i = 0 \tag{4.18}$$

$$v_i \cdot \vec{r}_3^T \vec{x}_i - \vec{r}_2^T \vec{x}_i = 0 \tag{4.19}$$

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

$$\hat{\mathbf{P}} = \arg\min_{\mathbf{P}} \sum_{i=1}^{N} \left[u_i \cdot \vec{r}_3^T \vec{x}_i - \vec{r}_1^T \vec{x}_i \right]^2 + \left[v_i \cdot \vec{r}_3^T \vec{x}_i - \vec{r}_2^T \vec{x}_i \right]^2 \tag{4.20}$$

$$\text{subject to: } \|\mathbf{P}\|_F = 1 \tag{4.21}$$

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 4.18. The equations in 4.18 looks then like:

$$\mathbf{M} \begin{pmatrix} p_{1,1} \\ p_{1,2} \\ \vdots \\ p_{3,3} \\ p_{3,4} \end{pmatrix} = 0 \tag{4.22}$$

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:

$$\|\mathbf{Mp}\|^2 \rightarrow \min, \qquad \text{subject to } \|\mathbf{p}\|^2 = 1 \tag{4.23}$$

which can be done with the Lagrange multiplier method:

$$\mathbf{p}^T\mathbf{M}^T\mathbf{Mp} - \lambda(\mathbf{p}^T\mathbf{p} - 1) \rightarrow \min \tag{4.24}$$

compute the derivative and the zero crossings:

$$2\mathbf{M}^T\mathbf{Mp} - 2\lambda\mathbf{p} = 0 \tag{4.25}$$

which is nothing else then:

$$\mathbf{M}^T\mathbf{Mp} = \lambda\mathbf{p} \tag{4.26}$$

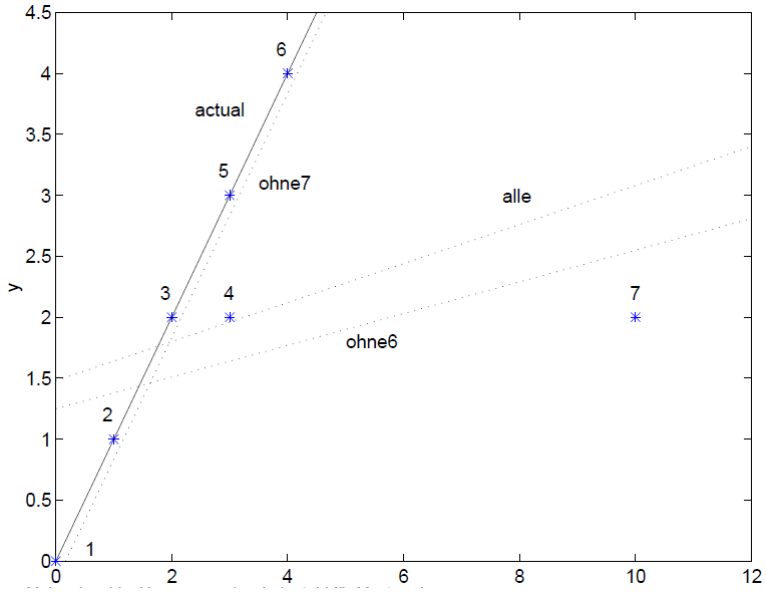
So the components of the projection matrix $\mathbf{P}$ result from the eigenvector belonging to the smallest eigenvalue. As mentioned above this linear estimate of $\mathbf{P}$ is an excellent initialization for the non-linear least square estimation of the projection matrix!

4.5. RANSAC

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

- badly localized points
- wrong correspondence

in figure 4.5 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.

5. Reconstruction

5.1. Thomography

the basic idea of Thomography is to solve the Puzzel in figure 5.1

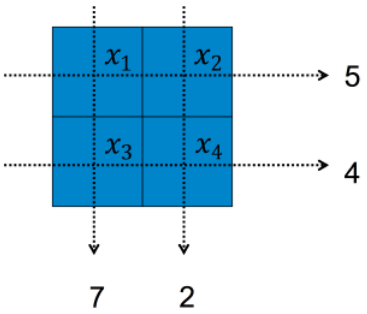


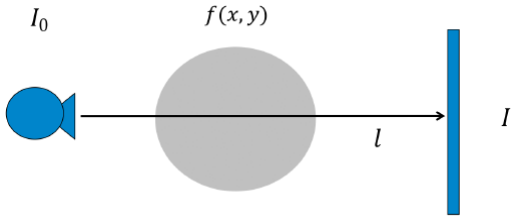
Abbildung 5.1.: basic Tomography Idea

$$\begin{aligned} x_1 + x_3 &= 7 & x_1 &= 3 \\ x_2 + x_4 &= 2 & x_2 &= 2 \\ x_1 + x_2 &= 5 & x_3 &= 4 \\ x_3 + x_4 &= 4 & x_4 &= 0 \end{aligned} \tag{5.1}$$

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 !

5.1.1. X-Ray attenuation 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

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

Abbildung 5.2.: 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: 5.3), we can describe these lines with this equation:

$$\vec{n}^T\vec{x} - d = 0 \tag{5.2}$$

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 2.3.2. If the points lie on the Line Dirac's delta funktion

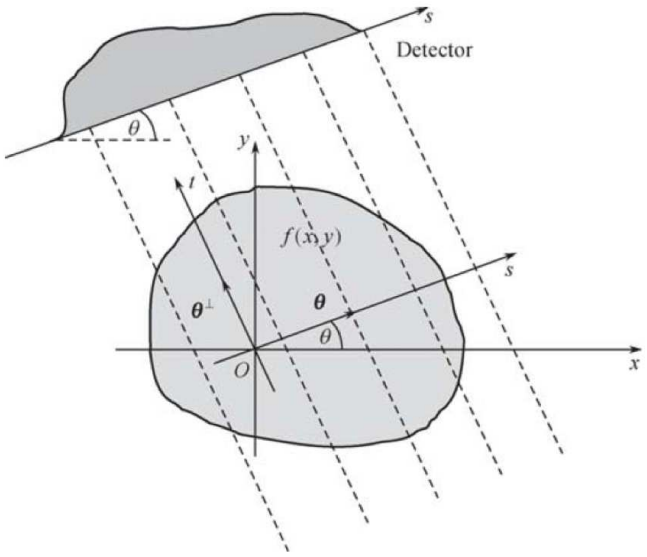


Abbildung 5.3.: parallel projection rays

Abbildung 5.4.: The x-axis represents s and the y-axis represents the rotation-angle θ

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

$$p = \int f(x,y)dl = \int\int f(x,y)\delta(x \cdot \cos\theta + y \cdot \sin\theta - s)dxdy \tag{5.3}$$

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 5.3) We can know 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 5.4). As a

simple Example we reconstruct the Problem of 5.1, where $\mathbf{P}$ are the measurements and Matrix $\mathbf{A}$ represents the linear equations we set up 5.1:

$$\mathbf{P} = \mathbf{AX} \tag{5.4}$$

$$\mathbf{P} = \begin{pmatrix} 7 \\ 2 \\ 5 \\ 4 \end{pmatrix}, \qquad \mathbf{A} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}, \qquad \mathbf{X} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \tag{5.5}$$

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 512×512 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 $\mathbf{A}$!

5.1.2. Backprojection

Another idea of reconstruction is the Backprojection. Which means we smear back additive the measured values along the projection line: in mathematical formulation

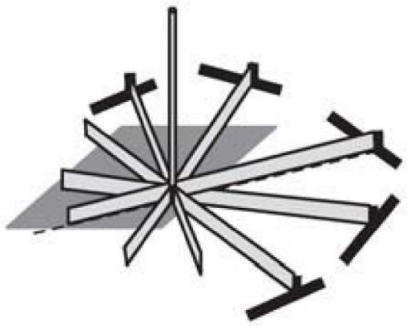


Abbildung 5.5.: smear back additive the measured Values

this means:

$$\text{Projection: } \mathbf{A}\vec{x} = \vec{p}, \qquad \Rightarrow \qquad \mathbf{A}^T\mathbf{A}\vec{x} = \underbrace{\mathbf{A}^T\vec{p}}_{\text{Backprojection}} \tag{5.6}$$

Applying Backprojection on our introduction Example 5.1 we get: this are obviously

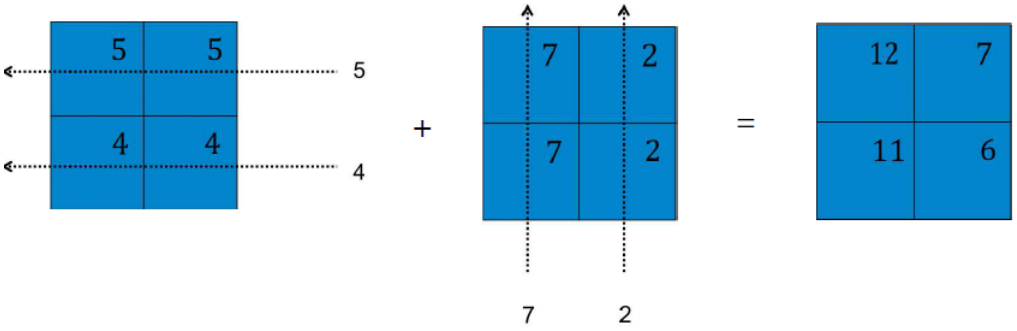


Abbildung 5.6.: smear back additive the measured Values

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 5.6.
The mathematical expression of this example is:

$$\mathbf{B} = \mathbf{A}^T\mathbf{P} \tag{5.7}$$

$$\mathbf{P} = \begin{pmatrix} 7 \\ 2 \\ 5 \\ 4 \end{pmatrix}, \qquad \mathbf{A}^T = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}, \qquad \mathbf{B} = \begin{pmatrix} 12 \\ 7 \\ 11 \\ 6 \end{pmatrix} \tag{5.8}$$

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

$$b(x,y) = \int_0^{\pi} p(s,\theta) \underbrace{|_{s=x\cos\theta+y\sin\theta}}_{\text{normal vector of the given line}} d\theta \tag{5.9}$$

so you sum up - integrate - over all considered rotation angles for a given line.

6. 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 **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 **V** which is related to the zero-entries on the diagonal of **S**. In our example case, the last diagonal entry will be zero after you applied the Threshold. Which means the last collumn of **V** is the null space for Matrix **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 **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 **A** is then the collumn-vectors of **U** which are related to the non-zero entries of **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