Universiteit Utrecht

Faculteit bètawetenschappen

bachelorscriptie

Schaduwtomografie

Auteur:
Hannah Gathu
12 januari 2018

Begeleider:
Tristan van Leeuwen

Inhoudsopgave

1 Inleiding

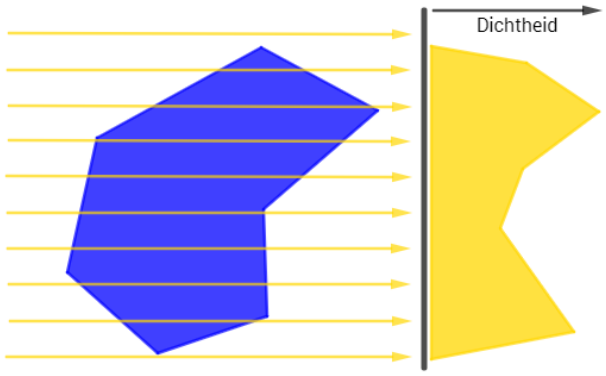
Tomografie is het maken van een twee-dimensionale doorsnede van een driedimensionaal object, zonder het object binnen te dringen.
Tomografie wordt vooral in de geneeskunde veel toegepast, om een doorsnede van het lichaam of een deel daarvan te reconstrueren[1]. Een van de manieren waarop dit gebeurt is met behulp van röntgenstraling. Deze vorm van tomografie wordt röntgentomografie genoemd. Hierbij wordt straling door een lichaamsdeel gezonden en wordt de intensiteit voor het betreden en na het verlaten van het lichaam gemeten. Omdat de weefsels in het lichaam de straling in verschillende mate doorlaten, kunnen we op basis hiervan een reconstructie van een doorsnede van het lichaamsdeel maken.
Een minder bekende vorm van tomografie is akoestische tomografie[2]. Deze vorm van tomografie wordt gebruikt bij de stabiliteitsanalyse van een boom. Dit werkt omdat geluidsgolven zich langzamer verplaatsen door hout van slechtere kwaliteit. Op basis van de snelheden van de verschillende geluidsgolven wordt met behulp van tomografie een doorsnede van de boom gemaakt.
In deze scriptie gaan we onderzoek doen naar een nieuwe soort tomografie, namelijk schaduwtomografie. Het idee hiervan is dat we aan de hand van de schaduw van een object bij licht vanuit verschillende invalshoeken, de doorsneden van een object gaan reconstrueren.
Hiermee zouden we uiteindelijk een driedimensionale reconstructie van het object kunnen maken. Zoals we in de figuur op de voorkant kunnen zien, is dit nog niet zo makkelijk: een schaduw kan verraderlijk zijn.
We beginnen in hoofdstuk **2** met het formuleren van het probleem. In sectie **2.1** demonstreren we hoe de gebruikelijke vorm van tomografie werkt en in sectie **2.2** leggen we uit wat schaduwtomografie is en aan welke eisen onze oplossing moet voldoen.
In hoofdstuk **3** definiëren we een barrièrefunctie, die ons helpt bij het concretiseren van onze oplossing.
Vervolgens beschrijven we in hoofdstuk **4** een methode om onze oplossing te berekenen, de gradient descent methode. In sectie **4.3** vertellen we hoe we deze implementeren in python.
Daarna bespreken we in hoofdstuk **5** de resultaten en tot slot trekken we in hoofdstuk **6** onze conclusie.

2 Tomografie

2.1 Tomografie bij een transparant voorwerp

De tomografie die onder andere in de geneeskunde veel wordt gebruikt is gebaseerd op het feit dat bepaalde delen van een voorwerp meer straling doorlaten dan andere[1].
Door straling door een transparant voorwerp te zenden en te meten hoeveel hiervan het object verlaat, kunnen we dan een 2-D reconstructie van een doorsnede van het voorwerp maken. Hierbij wordt steeds vanuit een bepaalde hoek een straal door het voorwerp gezonden. Nadat de straal het voorwerp heeft verlaten, wordt gemeten hoeveel de intensiteit van de straal is afgenomen. Wanneer de intensiteit onveranderd is gebleven, betekent dit dat er geen voorwerp in de lijn van de straal lag. Wanneer de intensiteit wel is afgenomen zegt de mate waarin dit is gebeurd iets over de dichtheid van het voorwerp over die lijn.
In figuur 1 is een voorbeeld te zien waarbij er over dezelfde hoek vanuit verschillende beginpunten stralen door een voorwerp worden gestuurd. Vervolgens wordt aan de hand van de intensiteit van de lichtstralen de dichtheid van het voorwerp over die lijnen bepaald, dit is

in het geel weergeven.



Figuur 1: Door straling door een voorwerp te zenden en te detecteren kunnen we de dichtheid van het voorwerp over die straal achterhalen.

2.1.1 Het model

Zoals eerder gezegd is stralingstomografie gebaseerd op het feit dat wanneer een lichtbundel met een bepaalde intensiteit door een absorberend materiaal wordt gezonden, deze bij het verlaten van het materiaal een lagere intensiteit heeft.
Bij tomografie met behulp van straling wordt de wet van Lambert-Beer[3] gebruikt. Deze wet beschrijft de afzwakking van een straal ℓ als gevolg van het voorwerp waar de straal doorheen wordt gezonden. Deze wet luidt als volgt:

$$I = I_0 \exp\left[-\int_\ell \mu(x)dx \right]. \tag{2.1}$$

Hierbij is I_0 de intensiteit van de straal direct na het verlaten van de bron en I de gemeten intensiteit (na het eventueel verlaten van het voorwerp). $\mu(x)$ is een dichtheidsfunctie van een lijn in het voorwerp. Des te groter μ is, des te meer de straling gedempt wordt. Het doel is nu om $\mu(x)$ te construeren op basis van metingen langs verschillende stralen (zie figuur 1) Dit probleem is niet lineair in μ en daarom moeilijk op te lossen. Om hier een lineair probleem van te maken introduceren we de variabele p:

$$p = \ln(I/I_0) = \int_\ell \mu(x)dx. \tag{2.2}$$

Om een stelsel vergelijkingen te krijgen dat we kunnen oplossen moeten we dit model discretiseren.

2.1.2 Het gediscretiseerde model

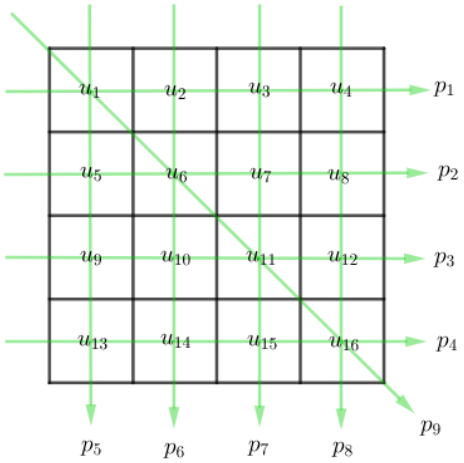
We zouden een doorsnede van een voorwerp kunnen weergeven in een matrix, waarbij de 'hokjes' pixels zijn (zie figuur 2).
Stel dat we metingen over l lijnen hebben en onze matrix uit k pixels bestaat.
Nu is p_i de totale dichtheid van een voorwerp gemeten over lijn i en geldt:

$$p_i = \sum_j w_{ij} u_j.$$

Waarbij w_{ij} het gewicht van de j^e pixel in de i^e straal is. Wanneer we alle l vergelijkingen bij elkaar nemen, krijgen we het volgende stelsel vergelijkingen:

$$\mathbf{p} = W\mathbf{u}.$$

Hierin is $\mathbf{p}$ een l-dimensionale vector en $\mathbf{u}$ een k-dimensionale vector. W is een $l \times k$ matrix met op plek ij het gewicht van pixel i voor lijn j. W bestaat uit nullen en enen: $w_{ij} = 1$ als straal j door de i^e pixel gaat en $w_{ij} = 0$ anders.



Figuur 2: Een voorbeeld van een gediscretiseerde doorsneden die bestaat uit 16 pixels, waarbij 9 metingen zijn gedaan.

2.1.3 Het vinden van een oplossing

Wanneer we het probleem formuleren als in sectie 2.1.1 willen we bij de reconstructie een oplossing vinden die zo goed mogelijk voldoet aan de metingen. Het liefst vinden we dus een vector $\mathbf{u}$ zodat $\mathbf{p} = W\mathbf{u}$.
Als W inverteerbaar is kunnen we de oplossing heel makkelijk vinden door

$$\mathbf{u} = W^{-1}\mathbf{p} \tag{2.3}$$

te nemen. Dan krijgen we namelijk

$$W\mathbf{u} = WW^{-1}\mathbf{p} = I\mathbf{p} = \mathbf{p}, \tag{2.4}$$

waarbij I de identeitsmatrix is.
De matrix W hoeft echter niet inverteerbaar te zijn. Als dit inderdaad niet het geval is, kunnen er de volgende problemen optreden:

- Het stelsel kan **inconsistent** zijn. Dit betekent dat er geen oplossingen zijn. Dit gebeurt als $\mathbf{p}$ niet in de kolomruimte van W zit. Dan bestaat er simpelweg geen u waarvoor $\mathbf{p} = W\mathbf{u}$. De oorzaak hiervan is dat er meetfouten hebben plaatsgevonden en er meer vergelijkingen dan onbekenden (meer metingen dan pixels) zijn.
  Om nu toch een oplossing te kunnen vinden, kunnen we in plaats van een $\mathbf{u}$ zoeken

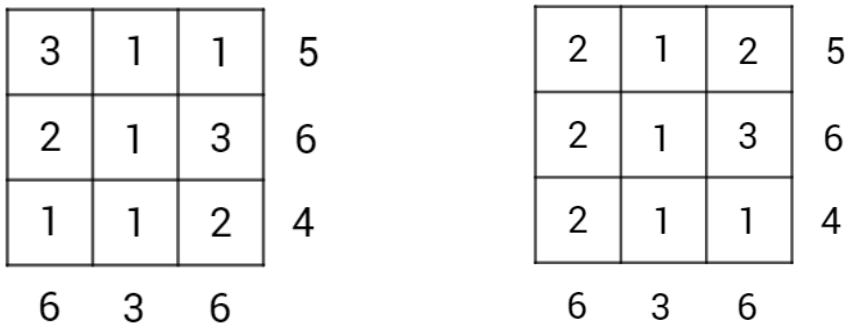
waarvoor $\mathbf{p} = W\mathbf{u}$, een $\mathbf{u}$ zoeken waarvoor het verschil tussen $\mathbf{p}$ en $W\mathbf{u}$ zo klein mogelijk is. Onze oplossing wordt dan gegeven door:

$$\mathbf{u} = \arg\min_{\mathbf{x}} \|\mathbf{p} - W\mathbf{x}\|_2^2, \tag{2.5}$$

waarbij arg min de waarde van $\mathbf{x}$ geeft waarvoor de functie minimaal is.

- Het stelsel kan nu echter **onderbepaald** zijn. Dit gebeurt als W een nulruimte heeft, oftewel als er een vector $\mathbf{z}$ is waarvoor $W\mathbf{z} = 0$.
  Dan kunnen we namelijk willekeurig veel veelvouden van $\mathbf{z}$ bij onze oplossing optellen en zijn er dus oneindig veel geldige oplossingen.
  Een voorbeeld van een onderbepaald stelsel is te zien in figuur 3. In deze figuur zie je twee verschillende oplossingen waarvoor $\|\mathbf{p} - W\mathbf{u}\|_2^2 = 0$.
  Hierbij moeten we in ons achterhoofd houden dat dit een voorbeeld met slechts negen pixels is, terwijl het er in de praktijk vaak meer zullen zijn. Bovendien zouden we de onderbepaaldheid van dit stelsel kunnen voorkomen door een meting over een van de diagonalen toe te voegen. Als dit echter geen optie is, kunnen we ervoor kiezen om uit deze oplossingen de oplossing met de kleinste norm $\|\mathbf{u}\|_2^2$ te nemen. De oplossing wordt dan gegeven door:

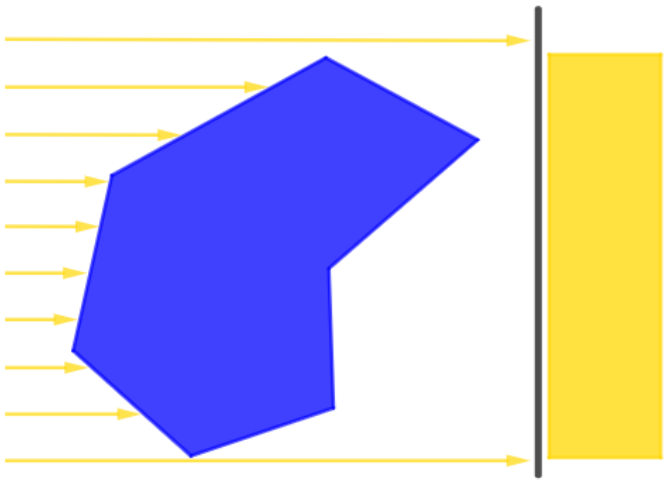
$$\mathbf{u} = \arg\min_{\mathbf{x}} \|\mathbf{p} - W\mathbf{x}\|_2^2 + \lambda\|\mathbf{x}\|_2^2. \tag{2.6}$$



Figuur 3: Dit figuur laat twee oplossingen van een onderbepaald stelsel zien, een stelsel met oneindig veel oplossingen.

2.2 Schaduwtomografie

In sectie 2.1 hebben we het probleem van reconstructie van een 2-D doorsnede van een transparant voorwerp en de oplossing hiervoor behandeld.
Bij schaduwtomografie hebben we te maken met een iets ander probleem. We kunnen alleen nog meten of er wel of geen voorwerp in de lijn van een straal ligt, maar niet welke dichtheid het voorwerp over die lijn heeft. We zijn dus informatie verloren. Je zou kunnen zeggen dat we nu te maken hebben met een intransparant voorwerp. Een schematische weergave van een schaduwmeting is te zien in figuur 4.



Figuur 4: Het voorwerp is nu intransparant geworden: òf al het licht wordt doorgelaten, òf er wordt geen licht doorgelaten. De dichtheid van het voorwerp is nu constant.

Wanneer we de lichtintensiteit bij verlaten van de bron als oneindig stellen, betekent dit dat de intensiteit van de straal na het verlaten van het voorwerp danwel nul, danwel oneindig is. Oftewel: $p_i \in \{0, \infty\}$.
De manier waarop we de oplossing in sectie 2.1 berekenden werkt nu niet meer, omdat de som van de pixels u_j over een straal nu ook kleiner dan de gemeten dichtheid p_i mag zijn. We moeten nu dus nieuwe eisen stellen.

2.2.1 Nieuwe eisen

We gaan nu de eisen opstellen waar de oplossing van ons nieuwe probleem aan moet voldoen. We merken de volgende dingen op:

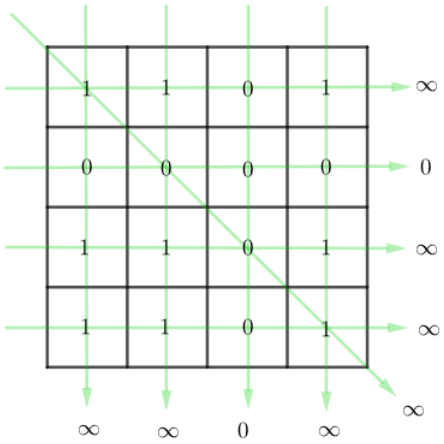
- Als $p_i = 0$ gaat straal i niet door het voorwerp heen. We weten dan dat alle pixels die door i doorkruist werden waarde nul moeten hebben, dus dat $\sum_j w_{ij} u_j = p_i = 0$. Om dit te bereiken stellen we de eis $\sum_j w_{ij} u_j \leq p_i$.
- Als $p_i = \infty$ betekent dit dat straal i door het voorwerp gaat. We willen nu dat minstens één van de pixels waar straal i doorheen gaat de waarde 1 heeft, dus dat $\sum_j w_{ij} u_j > 0$. We proberen dit te bereiken door u_j zo groot mogelijk te maken.

- Om te voorkomen dat u_j nu heel groot wordt, stellen we de eis dat $0 \leq u_j \leq 1$. Dit gaat ons later ook helpen als we de gevonden pixelwaarden naar nullen enen moeten schalen.

Dit komt neer op de volgende eisen voor onze oplossing **u**:

- $0 \leq u_j \leq 1$
- $\sum_j w_{ij} u_j \leq p_i$
- u_j maximaal

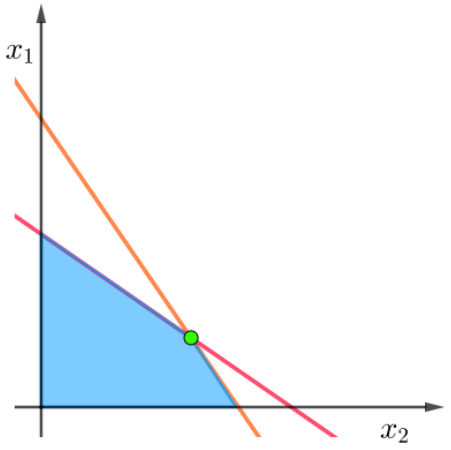
Een voorbeeld van een stelsel met een oplossing die aan al onze eisen voldoet is te zien in figuur 5.



Figuur 5: Een stelsel met hierin de oplossing die aan onze eisen voldoet.

2.2.2 Het bestaan van een unieke oplossing

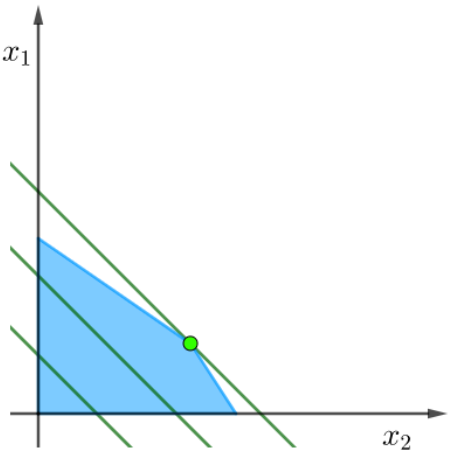
De eisen die we in sectie 2.2.1 hebben opgesteld vormen een lineair programma[4]. In figuur 6 is een voorbeeld van een lineair programma getekend waarbij $\mathbf{x}$ een (2×1) vector is. Dit is voor ons probleem geen realistisch voorbeeld, omdat we nooit een doorsnede zullen reconstueren die slechts uit twee pixels bestaat. Het voorbeeld is echter wel geschikt om inzicht te geven in het bestaan van een unieke oplossing.



Figuur 6: Een voorbeeld van een tweedimensionaal lineair programma. Het toegestane gebied voor $\mathbf{x}$ is in het blauw weergegeven.

1. **Kunnen we een oplossing vinden die aan onze eisen voldoet?**
   Een lineair programma heeft een oplossing mits de ongelijkheden waar de oplossing $\mathbf{x}$ aan moet voldoen elkaar niet tegenspreken. Stel bijvoorbeeld dat we een lineair programma hebben met de eisen $x \geq 1$ en $x \leq 0$, dan kunnen we nooit een oplossing vinden die aan alle eisen voldoet. In dit geval zou het blauwe gebied in figuur 6 leeg zijn en zou het dus niet mogelijk zijn om x te maximaliseren binnen het toegestane gebied.
   Bij ons lineair programma kan dit niet optreden, omdat onze matrix W alleen maar uit positieve getallen bestaat en we alleen maar kleiner dan eisen hebben. Voor positieve x kunnen de eisen $x \leq a$ en $x \leq b$ elkaar immers nooit tegenspreken. we kunnen dus altijd een oplossing vinden die aan al onze eisen voldoet.

2. **Is de oplossing uniek?**
   In figuur 7 is te zien dat de oplossing van het lineaire programma wordt bepaald door lijnen te tekenen waar de som van de variabelen constant is. Deze lijnen noemen we de somlijnen. Het snijpunt van een van deze lijnen met de rand van het toegestane gebied is het maximum.
   Deze methode zou oneindig veel oplossingen kunnen opleveren, als de somlijnen evenwijdig lopen aan de rand van het toegestane gebied. Oftewel, alseen van de ongelijkheden de som van alle waarden x_j afschat. Met onze eisen zou dit probleem alleen kunnen optreden als we twee pixels en één meting, dus als we zorgen dat $k \geq 3$, is er altijd een unieke oplossing.



Figuur 7: Door de lijnen te tekenen waar de som van de variabelen constant is kunnen we het maximum vinden. Dit maximumis in het lichtgroen weergegeven

3 Barrière functie

Stel dat we de (continue) functie $f(x)$ willen minimaliseren maar ook de eis $x \geq a$ hebben. Dus we hebben het volgende probleem:

minimaliseer $f(x)$
met $x \geq a$.

Dit probleem is lastig op te lossen, omdat we een eis op x hebben die niet in de minimalisatiefunctie voorkomt. We willen de te minimaliseren functie nu zo aanpassen dat we zeker weten dat het minimum hiervan voldoet aan de eis $x \geq a$. In het ideale geval zouden we een functie $c(x)$ bij $f(x)$ willen optellen die oneindig is voor $x < a$ en nul anders. Dan zouden we het minimalisatieprobleem als volgt kunnen herformuleren:

minimaliseer $f(x) + c(x)$
met

$$c(x) = \begin{cases} \infty & \text{als } \ x < a \\ 0 & \text{anders.} \end{cases}$$

We weten nu zeker dat het minimum zal voldoen aan de eis $x \geq a$, maar de te minimaliseren functie is wel discontinu geworden. Dit is een probleem, omdat we nu niet meer de afgeleide van de functie kunnen gebruiken om het minimum te vinden.

Om dit probleem op te lossen introduceren we de barrièrefunctie. Een barrière functie $g(x)$ is een continue functie die naar oneindig gaat als x tot b nadert. Belangrijk hierbij is dat $g(x)$ vrij plotseling naar oneindig gaat, zodat de functie op de rand van het gebied een hele grote waarde heeft, maar bij de rand vandaan niet. Op deze manier krijgen we hetzelfde effect als toen we de functie $c(x)$ gebruikten, maar is de te minimaliseren functie wèl differentieerbaar.
We kunnen met behulp van onze barrièrefunctie een nieuw probleem introduceren dat lijkt op het op te lossen probleem:

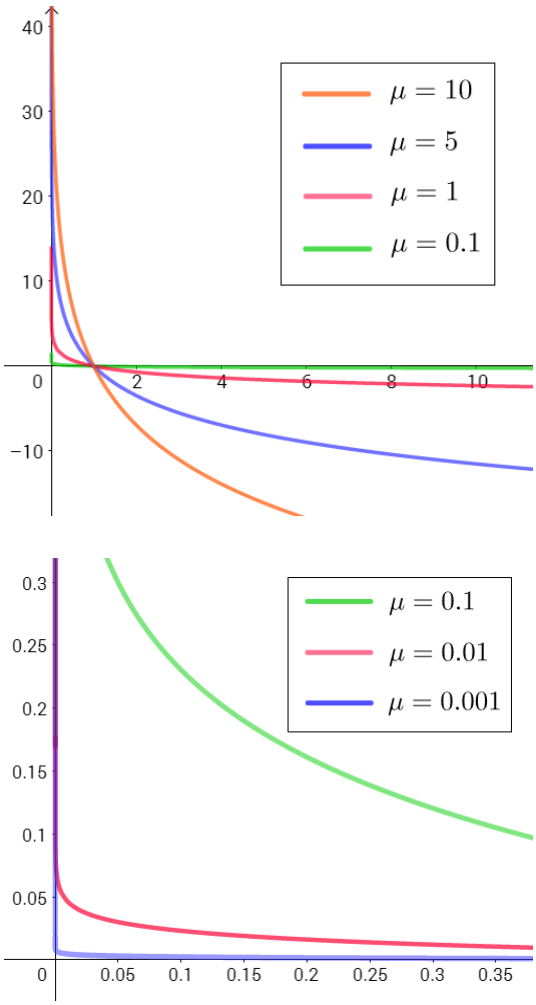
minimaliseer $f(x) + \mu g(x)$
waarbij $\mu > 0$.

3.1 Logaritmische barrièrefunctie

Een simpele en effectieve barrièrefunctie is de logaritmische functie. De functie $\log(x)$ gaat naar oneindig als x tot nul nadert, en als je de functie met een hele kleine constante vermenigvuldigt, gaat dit heel plotseling en ontstaat er een zogenaamde barrière. Dit maakt de functie zeer geschikt om als barrièrefunctie te fungeren.
In figuur 8 is de functie $-\mu \log(x)$ weergegeven voor verschillende waarden van μ. We zien inderdaad dat hoe kleiner de waarde van μ wordt, hoe scherper de hoek wordt waarmee de grafiek omhoog schiet bij het (in dit geval vanaf boven) naderen van de grens van het toegestane gebied voor x.
In het voorbeeld hiernaast schiet de functie omhoog bij het naderen van $x = 0$. Deze functie zou geschikt zijn als we de eis hebben dat x groter dan nul is. Stel echter dat we zoals in de voorgaande sectie de eis hebben dat x groter of gelijk aan een constante a is, dan moeten we de functie iets aanpassen. We kunnen dat we aan het begin van sectie 3 hebben geformuleerd nu als volgt herformuleren:

minimaliseer $f(x) + \mu\, g(x)$
met $g(x) = -\log(x - a)$.

Nu zal de functie als x vanaf onder tot a nadert naar oneindig gaan. Als we μ klein genoeg kiezen krijgen we op deze manier het minimum van $f(x)$ waarvoor $x \geq a$.



Figuur 8: De grafiek van $-\mu \ln(x)$ voor verschillende μ-waarden. We zien dat de functie zich naarmate $-\mu$ kleiner wordt steeds meer als een barrière gaat gedragen.

3.2 Het formuleren van een oplossing

We gaan de barrièremethode nu gebruiken om een oplossing voor ons probleem te formuleren.
In Sectie 2.2.1 hebben we de eisen opgesteld waar de oplossing van ons probleem aan moet voldoen. De eerste twee eisen bevatten beiden afschattingen, dus hier kunnen we een barrièrefunctie voor gebruiken. Het zou mooi zijn als we uiteindelijk een functie hebben die we kunnen minimaliseren met behulp van een gradiëntmethode. Als we alleen de laatste

eis, u_j zo groot mogelijk, meenemen in de te minimaliseren functie krijgen we het volgende:

$$\mathbf{u} = \arg\min_{\mathbf{x}} \ -\lambda \sum_j x_j$$
Met $0 \leq u_j \leq 1$ en $\sum_j w_{ij} u_j \leq p_i$.

We kunnen nu allereerst de logaritmische barrièrefunctie gebruiken om de eis $0 \leq u_j \leq 1$ in de functie te verwerken. Zo krijgen we:

$$\mathbf{u} = \arg\min_{\mathbf{x}} \ -\lambda \sum_j x_j - \mu_1 \sum_j \log(x_j) - \mu_2 \sum_j \log(1 - x_j)$$
met $\sum_j w_{ij} u_j \leq p_i$.

Vervolgens kunnen we de eis dat $\sum_j w_{ij} u_j \leq p_i$ in de functie verwerken. De oplossing wordt nu:

$$\mathbf{u} = \arg\min_{\mathbf{x}} \ -\lambda \sum_j x_j - \mu_1 \sum_j \log(x_j) - \mu_2 \sum_j \log(1 - x_j) - \mu_3 \sum_i \log(p_i - W_i \mathbf{x}). \quad (3.1)$$

Waarbij W_i de i^e rij van de matrix W is.
We hebben nu de oplossing van ons probleem geformuleerd. Om deze oplossing te berekenen gaan we de gradient descent methode gebruiken.

4 Gradient descent methode

De gradient descent methode is een methode om problemen van de vorm

$$\min_{x \in \mathbb{R}} f(x)$$

op te lossen. Om met behulp van deze methode een lokaal minimum van een functie te vinden, nemen we stappen die evenredig zijn met de gradiënt van de functie op het huidige punt.

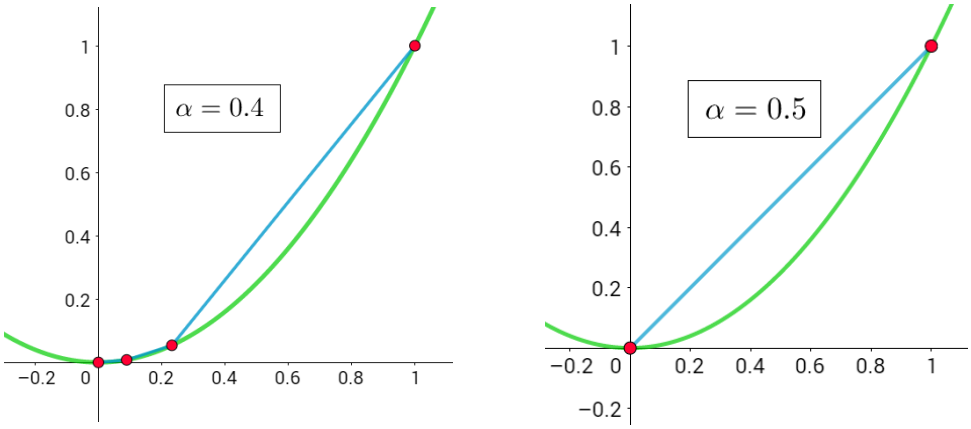
4.1 Gradient descent methode in één variabele

De gradiënt descent methode is gebaseerd op het feit dat een functie $f(x)$ het snelste afneemt wanneer men vanaf x in de richting van de negatieve gradiënt van f in x beweegt.
Dit is te vergelijken met wanneer je op een heuvel staat en zo snel mogelijk naar de vallei wil lopen. Dan loop je natuurlijk heuvel af en zo stijl mogelijk naar beneden, dus als het ware in de richting van de negatieve gradiënt van de heuvel.
De recursievergelijking voor de gradient descent method ziet er als volgt uit:

$$x_{k+1} = x_k - \alpha f'(x_k). \quad (4.1)$$

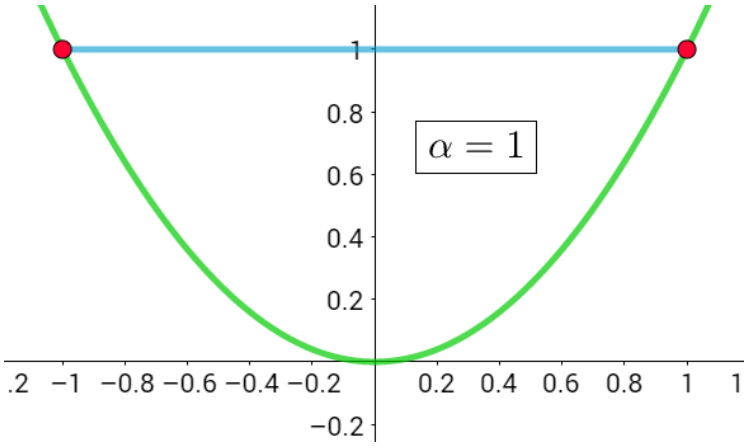
Deze vergelijking convergeert naar een lokaal minimum mits α klein genoeg is. In figuur 9 is voor de functie $f(x) = x^2$ voor twee verschillende waarden van α het verloop van de gradient descent methode weergegeven. Bij beide hebben we $x_0 = 1$ genomen.

We zien dat er voor $\alpha = 0.5$ slechts één iteratie nodig is. Dit is de optimale waarde van α voor deze functie. Voor $\alpha = 0.4$ convergeert de methode ook, maar minder snel dan voor $\alpha = 0.5$.



Figuur 9: De functie $f(x) = x^2$ en de bijbehorende iteraties voor $\alpha = 0.4$(links) en $\alpha = 0.5$(rechts)

Wanneer we een te grote waarde voor α nemen, convergeert de methode helemaal niet. Een voorbeeld hiervan is te zien in figuur 10. In dit voorbeeld blijft x steeds heen en weer schommelen tussen de waarden min één en één.



Figuur 10: Als we α te groot nemen convergeert de methode niet naar het minimum.

Om te voorkomen dat onze methode zoals in figuur 10 niet convergeert, moeten we een grenswaarde voor α bepalen. Uit vergelijking 4.1 volgt dat:

$$f(x_{k+1}) = f(x_k - \alpha f'(x_k)). \tag{4.2}$$

Om de rechterkant van deze vergelijking bruikbaarder te maken, gebruiken we de stelling[6] van Taylor:

Stelling (Taylor). *Stel $f : \mathbb{R} \longrightarrow \mathbb{R}$ is tweemaal differentieerbaar in x, dan:*

$$f(x+a) = f(x) + af'(x) + \int_x^{x+a} (t-x)f''(t)dt.$$

Omdat $(t-x)$ tussen x en $x+a$ altijd positief is, kunnen we de tussenwaardestelling voor integralen[5] gebruiken:

Stelling (Tussenwaardestelling voor integralen). *Laat $f : [a,b] \longrightarrow \mathbb{R}$ continu en g een integreerbare functie die niet van teken veranderd op het interval $[a,b]$, dan bestaat er een c in (a,b) zodat:*

$$\int_a^b f(x)g(x)dx = f(c)\int_a^b g(x)dx.$$

Uit deze stelling en vergelijking 4.2 volgt nu dat:

$$f(x+a) = f(x) + af'(x) + f''(c)\int_x^{x+a} (t-x)dt. \tag{4.3}$$

Als we aannemen dat $|f''(c)| \leq L$ voor c in $(x, x+a)$, krijgen we:

$$\begin{aligned} f(x+a) &\leq f(x) + af'(x) + L\int_x^{x+a} (t-x)\ dt \\ &= f(x) + af'(x) + L\ \frac{1}{2}t^2 - xt\Big|_x^{x+a} \\ &= f(x) + af'(x) + L\frac{1}{2}t^2. \end{aligned}$$

Als we nu $x = x_k$ en $a = -\alpha f'(x_k)$ nemen, krijgen we:

$$\begin{aligned} f(x_{k+1}) &= f(x_k - \alpha f'(x_k)) \\ &\leq f(x_k) - \alpha f'(x_k) + \frac{1}{2}L[\alpha f'(x_k)]^2 \\ &= f(x_k) - [f'(x_k)]^2(\alpha - \frac{L}{2}\alpha^2). \end{aligned}$$

We willen dat $f(x_k) < f(x_{k+1})$, zodat we dichter bij het minimum komen. Dus we willen dat $[f'(x_k)]^2(\alpha - \frac{L}{2}\alpha^2)$ maximaal is. Dit gebeurt als $\alpha - \frac{L}{2}\alpha^2$ maximaal is. Zo vinden we dat de methode maximaal convergeert voor:

$$\alpha = \frac{1}{L}.$$

4.2 Gradient descent methode in meerdere variabelen

We kunnen de gradiëntmethode ook toepassen op een functie van meerdere variabelen. Dan ziet de recursievergelijking er als volgt uit:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha\nabla f'(\mathbf{x}_k).$$

We gaan nu opnieuw de waarde van alfa bepalen waarvoor deze methode het snelst naar een geschikte oplossing convergeert. We willen dus een alfa vinden zodat $f(\mathbf{x}_{k+1})$ kleiner is dan $f(\mathbf{x}_k)$. Hiervoor gebruiken we de stelling[6] van Taylor voor een functie in meerdere variabelen:

Stelling (**Taylors stelling in meerdere variabelen**). *Laat $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ continu en f een tweemaal differentieerbare functie. Dan is er een $\xi \in (x, x + \Delta x)$ zodat*

$$f(\mathbf{x} + \Delta\mathbf{x}) = f(x) + \nabla f(x)^\intercal \Delta\mathbf{x} + ... + \frac{f^{(r-1)}(a)}{(r-1)!}\Delta\mathbf{x}^{(r-1)} + \frac{f^{(r)}(\xi)}{r!}\Delta\mathbf{x}^r.$$

Wanneer we deze stelling gebruiken om $f(\mathbf{x_{k+1}})$ om te schrijven krijgen we:

$$\begin{aligned} f(\mathbf{x}_{k+1}) &= f(\mathbf{x}_k) - \alpha\nabla f(\mathbf{x}_k)^\intercal \nabla f(\mathbf{x}_k) + \frac{1}{2}\alpha^2 \nabla f(\mathbf{x}_k)^\intercal \nabla^2 f(\xi)\nabla f(\mathbf{x}_k) \\ &= f(\mathbf{x}_k) - \alpha\|\nabla f(\mathbf{x}_k)\|_2^2 + \frac{\alpha^2}{2}\nabla f(\mathbf{x}_k)^\intercal \nabla^2 f(\xi)\nabla f(\mathbf{x}_k). \end{aligned}$$

We willen dat $f(\mathbf{x}_{k+1})$ kleiner is dan $f(\mathbf{x}_k)$, dus dat

$$-\alpha\|\nabla f(\mathbf{x}_k)\|_2^2 + \frac{\alpha^2}{2}\nabla f(\mathbf{x}_k)^\intercal \nabla^2 f(\xi)\nabla f(\mathbf{x}_k) < 0.$$

Wanneer we dit verder uitwerken komen we uit op het volgende:

$$\alpha < \frac{2}{\left(\dfrac{\nabla f(\mathbf{x}_k)^\intercal \nabla^2 f(\xi)\nabla f(\mathbf{x}_k)}{\|\nabla f(\mathbf{x}_k)\|_2^2}\right)}.$$

Er geldt dat

$$\frac{\nabla f(\mathbf{x}_k)^\intercal \nabla^2 f(\xi)\nabla f(\mathbf{x}_k)}{\|\nabla f(\mathbf{x}_k)\|_2^2} < \max_v \max_x \frac{\mathbf{v}^\intercal \nabla^2 f(x)\mathbf{v}}{\|\mathbf{v}\|_2^2}.$$

Dus de methode convergeert het snelst voor:

$$\alpha = \frac{2}{L} \text{ met } L = \max_v \max_x \frac{\mathbf{v}^\intercal \nabla^2 f(x)\mathbf{v}}{\|\mathbf{v}\|_2^2}.$$

Hierbij valt ons op dat L gelijk is aan de spectrale radius, de grootste eigenwaarde, van de Hessiaan van $f(x)$. Er geldt nu dat

$$\alpha = \frac{2}{L} \text{ met } L = \rho(\nabla^2 f(\mathbf{x})),$$

waarbij $\rho(\nabla^2 f(\mathbf{x}))$ de spectrale radius van de Hessiaan is. Het berekenen van de spectrale radius van de Hessiaan is moeilijk, omdat we hiervoor de eigenwaarden van de Hessiaan nodig hebben. Daarom gebruiken we de volgende stelling[5]:

Stelling (**Spectrale radius en norm**). *Als $A \in \mathbb{R}^{nxn}$ en $\|A\|_p = \sup_{x \neq 0} \dfrac{\|A\mathbf{x}\|_p}{\|\mathbf{x}\|_p}$, dan:*

$$\rho(A) \leq \|A\|_p.$$

Omdat de norm $\|\nabla^2 f(\mathbf{x})\|_p$ niet makkelijk te berekenen is gebruiken we de volgende stelling[5]:

Stelling (Spectrale radius en norm). *Als* $\|A\|_p = \sup_{x \neq 0} \frac{\|A\mathbf{x}\|_p}{\|\mathbf{x}\|_p}$*, dan geldt:*

$$\|A\|_\infty = \max_{1 \leq i \leq m} \sum_{j=1}^{n} |a_{ij}|.$$

Dus

$$\alpha = \frac{2}{M} \tag{4.4}$$

waarbij M gelijk is aan het maximum van de rij-sommen van de Hessiaan, is de optimale waarde voor α.

4.3 Gradiënt en hessiaan

Om met behulp van de gradient descent methode een oplossing voor ons probleem te berekenen, gaan we deze in python implementeren. Voordat we dit kunnen doen moeten we eerst de gradiënt van de functie in vergelijking 3.1 bepalen. Deze functie luidt als volgt:

$$f(\mathbf{x}) = -\lambda \sum_j x_j - \mu_1 \sum_j \log(x_j) - \mu_2 \sum_j \log(1 - x_j) - \mu_3 \sum_i \log(p_i - W_i \mathbf{x}). \tag{4.5}$$

We vinden de volgende partiële afgeleiden:

$$\frac{\partial f}{\partial x_j} = -\lambda - \frac{\mu_1}{x_j} + \frac{\mu_2}{1 - x_j} - \mu_3 \sum_i W_{ij} \frac{1}{p_i - (\mathbf{w_i x})}. \tag{4.6}$$

De gradiënt van $f(\mathbf{x})$ wordt nu:

$$\nabla f(\mathbf{x}) = \begin{pmatrix} -\lambda - \frac{\mu_1}{x_1} + \frac{\mu_2}{1-x_1} - \mu_3 \sum_i W_{i1} \frac{1}{p_i - (\mathbf{w_i x})} \\ \vdots \\ -\lambda - \frac{\mu_1}{x_k} + \frac{\mu_2}{1-x_k} - \mu_3 \sum_i W_{ik} \frac{1}{p_i - (\mathbf{w_i x})}. \end{pmatrix} \tag{4.7}$$

Om de optimale waarde van α te kunnen implementeren moeten we ook de Hessiaan van $f(\mathbf{x})$ bepalen. We vinden de volgende tweede-orde partiële afgeleiden:

$$\frac{\partial^2 f}{\partial x_a \partial x_b} = -\mu_3 \sum_i \frac{W_{ia} W_{ib}}{(p_i - \mathbf{W_i x})^2}. \tag{4.8}$$

De hessiaan van $f(\mathbf{x})$ wordt nu:

$$-\mu_3 \; W^\intercal \begin{pmatrix} \frac{1}{(p_1 - \mathbf{W_1 x})^2} & & \\ & \ddots & \\ & & \frac{1}{(p_l - \mathbf{W_l x})^2}. \end{pmatrix} W. \tag{4.9}$$

4.4 Implementatie

We hebben onze methode in python geïmplementeerd. De pythoncode is terug te zien in de appendix.
We hebben bij het implementern in de gradiëntmethode voor x_0 een vector met overal de waarde 0.5 genomen. Hierdoor hebben we als α klein genoeg is slechts één iteratie nodig om het minimum van $f(\mathbf{x})$ te vinden. In figuur 11 is voor een voorbeeld van vier pixels het verloop van de gradiëntmethode te zien voor willekeurige α en voor α bepaald door middel van vergelijking 4.5. We zien dat de gradiëntmethode in figuur 11b naar verwachting sneller convergeert dan die in figuur 11a.
Omdat we de pixelwaarden in oplossing uiteindelijk schalen naar 0 of 1, maakt het echter niet uit of we de linker of rechter manier gebruiken. We hebben bij beiden manieren aan één iteratie genoeg. Daarom hebben we ervoor gekozen om α in onze implementatie niet op de manier van sectie 4.2 te berekenen.
We hebben de parameters op basis van de resultaten steeds iets verfijnd. De parameters die we uiteindelijk hebben gebruikt om de doorsneden te reconstrueren zijn terug te vinden in de bijlage.

```
[[ 0.5        0.5        0.5        0.5       ]
 [ 0.499      0.499      0.50400001 0.50400001]
 [ 0.49799038 0.49799038 0.50799842 0.50799842]
 [ 0.49697101 0.49697101 0.51199523 0.51199523]
 [ 0.49594174 0.49594174 0.51599044 0.51599044]
 [ 0.49490245 0.49490245 0.51998405 0.51998405]
 [ 0.49385299 0.49385299 0.52397605 0.52397605]
 [ 0.49279322 0.49279322 0.52796645 0.52796645]
 [ 0.49172298 0.49172298 0.53195524 0.53195524]
 [ 0.49064212 0.49064212 0.53594241 0.53594241]
 [ 0.48955051 0.48955051 0.53992797 0.53992797]]
oplossing: [ 0.  0.  1.  1.]
```

(a) Het verloop van $\mathbf{x}$ voor $\alpha = 0.5$

```
[[ 0.5         0.5         0.5         0.5       ]
 [ 0.3         0.3         1.300002    1.300002  ]
 [-0.00514286 -0.00514286  1.60277187  1.60277187]
 [ 0.00502073  0.00502073  1.60285892  1.60285892]
 [-0.00474025 -0.00474025  1.60294189  1.60294189]
 [ 0.00462165  0.00462165  1.60301584  1.60301584]
 [-0.00436929 -0.00436929  1.60308614  1.60308614]
 [ 0.00425485  0.00425485  1.60314897  1.60314897]
 [-0.00402745 -0.00402745  1.60320856  1.60320856]
 [ 0.00391761  0.00391761  1.60326194  1.60326194]
 [-0.00371241 -0.00371241  1.60331246  1.60331246]]
oplossing: [ 0.  0.  1.  1.]
```

(b) Het verloop van $\mathbf{x}$ voor α berekend middels vergelijking 4.6

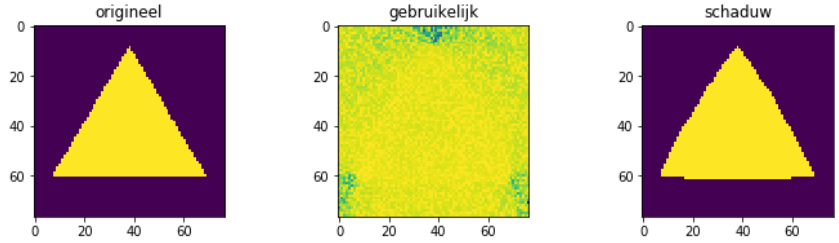
Figuur 11: Het verloop van de gradiëntmethode waarbij α willekeurig wordt gekozen(links) en waarbij α wordt berekend met vergelijking 4.5(rechts).

5 Resultaten

In dit hoofdstuk zullen we de resultaten van reconstructies door middel van onze methode bespreken.
Het resultaat dat we hebben verkregen bij het reconstrueren van de doorsnede van een bol is te zien in figuur 12. Het eerste plaatje geeft de originele doorsnede weer, het tweede plaatje geeft de reconstructie van deze doorsnede op gebruikelijke manier (Zoals beschreven in sectie 2.1.3) weer. Het meest rechter plaatje geeft de reconstructie met onze methode weer. We zien dat de gebruikelijke methode de doorsnede niet goed reconstrueert. De contouren van de cirkel zijn heel vaag terug te zien, maar verder toont deze reconstructie geen gelijkenissen met het origineel. We zien dat de reconstructie door middel van onze methode (rechts) bijna exact hetzelfde is als het origineel, een zeer goed resultaat dus. Onze methode is dus geschikt voor het reconstrueren van een bol.

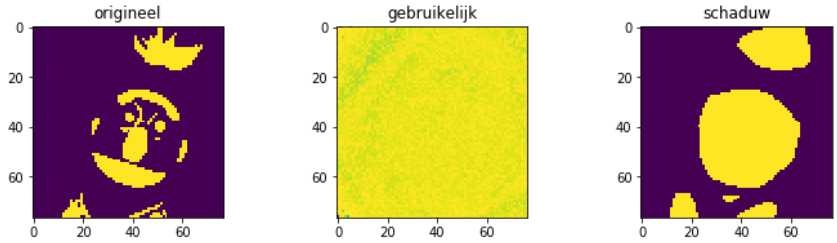
Figuur 12: Door straling door een voorwerp te zenden en te detecteren kunnen we de dichtheid van het voorwerp over die straal achterhalen.

In figuur 13 is het resultaat voor het reconstrueren van een driehoek te zien. We zien dat deze reconstructie minder goed is gelukt dan de reconstructie van de cirkel, maar de driehoek is zeker terug te herkennen in onze reconstructie.
De gebogen zijden in de reconstructie zijn waarschijnlijk te wijten aan de matrix W. We zouden dus nog kunnen kijken naar een effectievere manier om de matrix W op te bouwen, waarbij elke meting alleen de middens van de pixels doorkruist. We denken dat dit het resultaat van het reconstrueren van een prisma zou kunnen verbeteren.
Onze methode is dus redelijk geschikt om een driehoekig prisma te reconstrueren en met verbeteringen aan W zou de reconstructie wellicht nog beter kunnen worden.



Figuur 13: Door straling door een voorwerp te zenden en te detecteren kunnen we de dichtheid van het voorwerp over die straal achterhalen.

Het resultaat bij het reconstrueren van Bert van Bert en Ernie is in figuur 14 weergegeven. Het resultaat is in lijn met onze verwachting, omdat bepaalde delen van het voorwerp het licht blokkeren. wanneer we bijvoorbeeld een meting doen over de lijn die Berts oren en neus doorkruist, krijgen we alleen de informatie dat er ergens op die lijn een voorwerp is, maar komen we niets te weten over de lege ruimte tussen zijn oren en neus. Vandaar dat zijn hele gezicht in onze reconstructie één vlak is geworden. Dit is een gebrek van schaduwtomografie dat niet te verhelpen is.



Figuur 14: Door straling door een voorwerp te zenden en te detecteren kunnen we de dichtheid van het voorwerp over die straal achterhalen.

6 Conclusie

In deze scriptie hebben we een methode ontwikkeld om aan de hand van schaduwen de doorsneden van een voorwerp te reconstrueren, welke we schaduwtomografie noemen.
De resultaten zoals besproken in hoofdstuk 5, vertellen ons dat schaduwtomografie vooral geschikt is voor het reconstrueren van massieve voorwerpen.
We moeten helaas vaststellen dat bepaalde tekortkomingen van onze methode niet te verhelpen zijn. We kunnen niet voorkomen dat we bij het meten informatie over het voorwerp verliezen, bijvoorbeeld over holtes in het voorwerp.
We kunnen concluderen dat de ontwikkelde methode goed werkt om de informatie die in de schaduwen van een voorwerp verborgen zit te gebruiken om een zo goed mogelijke reconstructie van een voorwerp te maken.

Referenties

[1] Cantatore, A. en Müller, P. (2011): *Introduction to computed tomography*, Kgs.Lyngby: DTU Mechanical Engineering.

[2] Tomografie. (2016, juli 21) *Wikipedia*. Opgehaald 18:13, juli 21, 2016 van //nl.wikipedia.org/w/index.php?title=Tomografie&oldid=47117856.

[3] John Hardesty and Bassam Attili: *Spectrophotometry and the Beer-Lambert Law: An Important Analytical Technique in Chemistry*. Collin college, 2010

[4] Thomas S. Ferguson: *Linear programming: A Concise Introduction*, UCLA Department of Mathematics

[5] Uri M. Ascher and Chen Greif: *A First Course in Numerical Methods*, , 2011

[6] Serger Lang: *Calculus of several variables*, Yale University, 2016

Appendix

January 12, 2018

In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy import misc
from scipy import sparse
from scipy import optimize
import math
from scipy.optimize import linprog
```

In [3]:
```
def getRow(i,j,n,theta,offset,viz=False):

    # grid
    xg = np.linspace(-1,1,n+1)
    yg = np.linspace(-1,1,n+1)

    # compute coordinates of intersection with ray
    ax = offset[j]*np.sin(np.deg2rad(theta[i]))
    bx = np.cos(np.deg2rad(theta[i]))

    ay = -offset[j]*np.cos(np.deg2rad(theta[i]))
    by = np.sin(np.deg2rad(theta[i]))

    ti = []
    if abs(bx) > 1e-10:
        ti = np.append(ti, (xg - ax)/bx)
    if abs(by) > 1e-10:
        ti = np.append(ti, (yg - ay)/by)

    ti = ti[abs(ax + ti*bx)<=1]
    ti = ti[abs(ay + ti*by)<=1]
    ti.sort()

    xi = ax + ti*bx
    yi = ay + ti*by

    # compute lengths
    l = np.sqrt((xi[1:] - xi[:-1])**2 + (yi[1:] - yi[:-1])**2)
```

```
            # assign to cells and fill matrix
            ix = np.round(((xi[1:] + xi[:-1])/2)*n/2) + int(n/2)
            iy = np.round(((yi[1:] + yi[:-1])/2)*n/2) + int(n/2)

            col = [int(_) for _ in ix + iy*n]
            row = j + p*i

            if viz:
                # visualize
                plt.plot((2*ix + 1)/n - 1,(2*iy + 1)/n - 1,'s',ms=20) # plot centers of intersec
                plt.plot(ax + bx*ti, ay + by*ti,'-o') # plot ray and intersections
                plt.xticks(xg)
                plt.yticks(yg)
                plt.xlim([-1,1])
                plt.ylim([-1,1])
                plt.grid(lw=2)

            return (row,col,l)
```

In [4]:
```
#gradiënt van f:
def g(x,W,p,l1,l2,l3,l4):
    return -l1*x0**0 - l2/x + l3/(1-x) - l4*(W.T@(1/(p - W@x)))
```

In [24]:
```
#2e partiële afgeleide van f  (df/dx_adx_b):
def Hab(x,a,b,W,p,l4):
    return -l4*sum((W[i][a]*W[i][b]/(p[i]-W[i]@x)**2)for i in range(len(p)))
```

In [23]:
```
# norm van hessiaan:
def normH(x,W,p,l4):
    rijsommen=[(sum((abs(Hab(x,i,j,W,p,l4)))
                    for j in range(len(x))))for i in range(len(x))]
    return max(rijsommen)
```

In [5]:
```
#gradientmethode om minimum van f te vinden:
def gradientMethod(W, p, l1, l2, l3, l4, x0, atol, maxit):
    x= np.zeros((maxit+1,len(x0)))
    x[0]=x0
    alpha=0.1
    for k in range(maxit):
        x[k+1]= x[k]-alpha*(g(x[k],W,p,l1,l2,l3,l4))
        if np.linalg.norm(x[k+1]-x[k])<=atol:
            break
    sol=x[k+1]

    #alle xi in de oplossing x op 0 of 1 zetten:
    for o in range(len(sol)):
        if sol[o]>=0.5:
            sol[o]=1
        elif sol[o]<0.5:
```

```
                sol[o]=0
        return sol
```

In [16]:
```
# Reconstructie:

#Het inlezen van één van de drie plaatjes:
#u = misc.imread('bert.png',flatten=True)
u = misc.imread('cirkel.jpg',flatten=True)
#u=misc.imread('driehoek.png',flatten=True)
u = u[1:78,1:78]
u = np.round(u/255)
print(u.shape)
n = u.shape[0]
m = n+2
p = n+2
theta = np.linspace(0,180,m);
offset = np.linspace(-1+1/p,1-1/p,p)

for j in range(77):
    for k in range(77):
        if u[j][k]==0:
            u[j][k]=1
        elif u[j][k]==1:
            u[j][k]=0

# empty matrix
rows = np.array(0)
cols = np.array(0)
coeffs = np.array(0)

# fill matrix
for i in range(m):
    for j in range(p):
        (row,col,l) = getRow(i,j,n,theta,offset)
        nc = len(col)
        if nc > 0:
            rows = np.append(rows,np.array(nc*[row]))
            cols = np.append(cols,col)
            coeffs = np.append(coeffs,l)

# create sparse matrix
A = sparse.coo_matrix((coeffs,(rows,cols)),shape=(m*p,n*n))

# create sinogram
d = A@u.reshape(n*n)
p = np.copy(d)

for i in range(len(p)):
```