Randherkenning en de Voorwaartse Euler Methode

Steven Raaijmakers, Daan Meijers

January 2016

1 Inleiding

1.1 Randherkenning

Een digitale foto kunnen wij zien als een veld van pixels met bepaalde waardes die aangeven welke kleur deze pixel moet zijn. We kunnen dit vervolgens als een functie met twee variabelen zien die de kleurwaarde als uitkomst heeft. Met behulp van de gradient, die wij berekenen door de numerieke partiele afgeleides van onze functie te berekenen, kunnen wij inschatten waar er zich randen bevinden in onze afbeelding. Dit kan makkelijker gemaakt worden met behulp van filters die randen beter herkenbaar maken. De filters die wij gebruikt hebben zijn die van Prewitt, Sobel, Laplace en Gauss. Het doel van onze experimenten is om te bepalen met welk filter wij de beste randherkinneng krijgen. De eerste beelden die wij hebben gegenereerd laten ons vermoeden dat het filter van Gauss het meest effectief is.

Een ander experiment is het bepalen van het beste beeld bij het filter van Gauss. Een vereiste voor dit filter is namelijk een standaardafwijking die we zelf kunnen bepalen. Deze standaardafwijking geeft aan hoezeer de randen worden verzacht. In dit experiment proberen we verschillende waardes uit voor deze variabele.

1.2 Lijnelementenveld Voorwaartse Euler Methode

Wanneer een differentiaalvergelijking bekend is, is het mogelijk een voorbeeld te krijgen van de daadwerkelijke oplossing hiervan (de integraal van de vergelijking). Aan de hand van de differentiaalvergelijking is het namelijk mogelijk de helling uit te rekeken voor punten (x, y). Bij een lijnelementenveld worden er in een grafiek, verschillende punten - dan wel niet willekeurig - gekozen die in elk punt de helling beschrijven. Zo ontstaat een veld van lijnen met bepaalde hellingen die mogelijke oplossingen zouden kunnen zijn van de differentiaalvergelijking.

Via de Voorwaartse Euler methode is het mogelijk om voor een differentiaalvergelijking ϕ een lijn op te stellen op basis van een startwaarde en een Δt.

Hiermee wordt een mogelijke oplossing gevonden van de differentiaal vergelijking.

2 Implementatie

2.1 Randherkenning

Om randen te vinden moeten we eigenlijk zoeken naar een gradient met een grote waarde. Dit wil zeggen dat het verschil in grijswaardes tussen bepaalde pixels erg groot is. Dit kunnen we vinden in de verticale en horizontale richting afzonderlijk door de partiele afgeleide in x of y te nemen met behulp van convolutie. Aangezien we in twee dimensies werken zullen we hiervoor een matrix gebruiken. De partiele afgeleide in de verticale richting ziet er dus als volgt uit:

$$f_x(i,j) = \frac{1}{2} f * \begin{pmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \tag{1}$$

2.1.1 Prewitt

Voor een gradient met minder ruis kunnen we ook de diagonalen buren van de betreffende pixel betrekken bij de convolutie. Dit is het verschil tussen de normale partiele afgeleide en wat het Prewitt-filter doet. De convoluties van het Prewitt-filter zien er dan ook als volgt uit:

$$Prewitt_x(f) = f * \begin{pmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{pmatrix}$$

$$Prewitt_y(f) = f * \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix} \tag{2}$$

In onze code passen we het Prewitt-filter in de verticale en horizontale richting toe. Vervolgens kunnen we vectoren tekenen door het resultaat samen te voegen met behulp van de pyplot functie quiver. Wat we nu krijgen zijn vectoren die loodrecht op randen staan. Oftewel pixels met een erg grote gradient. Deze vectoren tekenen we elke 16 pixels om niet het hele plaatje te vullen met pijlen.

2.1.2 Sobel

Het Sobel-filter is vrij vergelijkbaar met het Prewitt-filter. Het verschil zit in het feit dat het Sobel-filter naaste buren minder zwaar meerekent. Het idee van Sobel is dat de gradienten die we in horizontale en verticale richting krijgen samenvoegen met behulp van de volgende formule:

$$G = \sqrt{(I * S_x)^2 + (I * S_y)^2} \tag{3}$$

Hier is G de lengte van de gradient, I de 3x3 matrix van de relevante pixels en Sx en Sy de Sobel-filters die er als volgt uit zien:

$$S_x = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix}$$

$$S_y = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix} \tag{4}$$

In onze code voeren we exact deze formule uit en printen we het plaatje dat we vervolgens krijgen. Dit resulteert in een plaatje dat er uitziet als een negatief waarbij de randen duidelijk zichtbaar zijn.

2.1.3 Gauss

Nog een andere manier die gebruikt word bij randherkenning is de Gauss-convolutie. Het idee is dat we in eerste instantie het plaatje vervagen waardoor het verschil in pixelwaardes onderling vrij miniem is. Vervolgens trekken we de pixelwaardes van het vervaagd beeld van die van het originele beeld. Hierdoor houden we enkel de extreme waardes over die onze randen representeren. Zoals al eerder genoemd moeten we bij deze methode de standaardafwijking zelf bepalen. Deze moeten we zodanig kiezen dat in het resultaat wel randen te zien zijn maar ook weer niet teveel details. We kunnen het plaatje vervagen door convolutie met een Gauss-functie die er als volgt uit ziet:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{5}$$

Hier is sigma de standaardafwijking. In onze code hebben we de functie gaussianfilter van scipy gebruikt. Als input geven we het originele beeld en de standaardafwijking. We krijgen dan het vervaagde beeld terug in hetzelfde formaat als het beeld dat we hebben meegegeven. Hierdoor kunnen wij dus makkelijk het verschil bepalen. Verder hebben we voor het bepalen van de meeste effectieve standaardafwijking per plaatje de waardes 1 t/m 10 geprobeerd als standaardafwijking.

2.1.4 Laplace

Tenslotte hebben we nog de Laplace-operator. De Laplace-operator van een functie is gelijk aan de som van de tweede afgeleide in de horizontale en verticale richting. Deze som geeft duidelijk aan waar de gradient extreem is. Deze operator is gelijk aan de som van de volgende twee functies:

$$f_{xx}(i,j) \approx f(i+1,j) - 2f(i,j) + f(i-1,j)$$

$$f_{yy}(i,j) \approx f(i,j+1) - 2f(i,j) + f(i,j-1) \tag{6}$$

Dit is gelijk aan de convolutie van het beeld en de volgende matrix:

$$M = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix} \tag{7}$$

In de code voeren we ook hier enkel exact deze convolutie uit. Het resultaat is een beeld waar alle pixel vrijwel dezelfde kleur hebben met uitzondering van de randen die wit zijn.

2.2 Lijnelementenveld

Wanneer een differentiaalformule ϕ bekend is, kunnen we voor een bepaald interval nabootsen hoe de oplossingen ervan eruit moeten zien. Dit doen we doordat we de helling van een punt (x, y) kunnen berekeken door $\phi(x, y)$

We kozen verschillende punten, en voor al deze punten berekenden we de helling en tekenen de helling als een lijn van lengte l met het midden op de (x, y) coördinaat. Vervolgens kunnen we de schaal berekenen (we rekenen met vectoren) door de middel van de Pythagoras formule: $a^2 + b^2 = c^2$. In ons geval levert dat:

$$c = \sqrt{1^2 + \phi(x,y)^2} \tag{8}$$

Omdat het punt een lijn moet zijn waarvan het midden op de (x, y) coördinaten moet liggen passen we het volgende toe: we nemen een horizontale vector en een verticale vector:

$$h = \frac{1}{c}, v = \frac{\phi(x,y)}{c} \tag{9}$$

We kunnen nu de begincoördinaten van de lijn in (x, y) berekenen met:

$$x, y = ((x - l * h), (y - l * h)) \tag{10}$$

en de eind coördinaten met:

$$x, y = ((x + l * h), (y + l * h))$$

2.3 Euler Methode

De Voorwaartse Euler methode is een numerieke manier om differentiaal-vergelijking ϕ op te lossen volgens:

$$y_{n+1} = y_n + \phi(x_n, y_n) * \Delta x \tag{11}$$

Hier berekenen we de coördinaten wanneer (x_n, y_n) bekend is. $x_n = x_{n+1} - x_n$. In het begin is er nog geen y bekend, en dus ben je vrij om y_0, de beginwaarde, zelf te kiezen.

Met deze methode is het dus mogelijk aan de hand van ϕ en een y_0 een oplossing te vinden van ϕ.

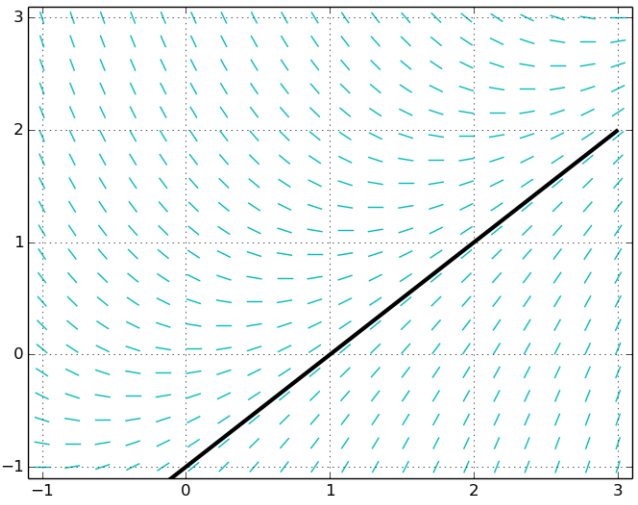
2.3.1 Methode

Om de kwaliteit van deze Euler methode te testen, hebben we gekozen om de volgende differentiaal vergelijking te nemen:

$$\phi(t, y) = \frac{dy}{dt} = t - y \tag{11}$$

We weten namelijk dat elke oplossingskromme hiervan de volgende lijn zal benaderen:

$$y(t) = t - 1 \tag{11}$$



We pasten verschillende parameters van de Euler methode aan, om zo te zien wat de afbreekfout was met $y(t)$.

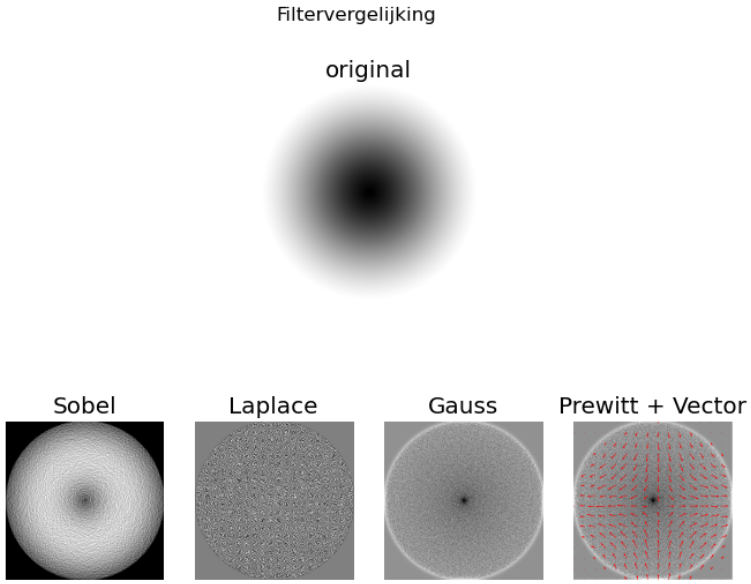
3 Meetresultaten

3.1 Randherkenning

De volgende afbeeldingen zijn enkele resultaten van de experimenten die we hebben gedaan. In totaal hebben we vijf afbeeldingen van verschillende aard gedaan. Naast de afbeeldingen die hier te zien zijn is er ook nog de foto van een boom gebruikt, maar gezien het feit deze vrij gedetailleerd is zijn er geen mooie resultaten uit gekomen. Bij deze afbeeldingen hebben wij elke methode er op toegepast. Daarnaast hebben we gekeken naar wat het effect is bij het gebruiken van verschillende waardes als standaardafwijking bij de methode van Gauss.

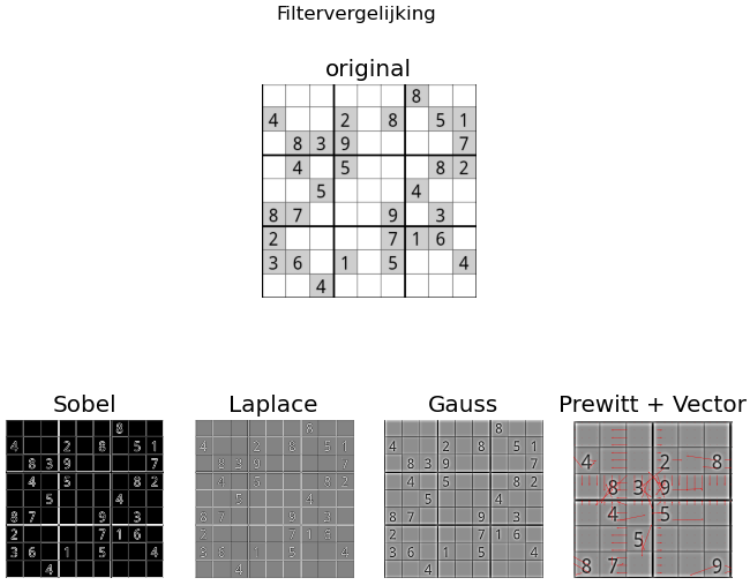
3.1.1 Filtervergelijking

De volgende twee plaatjes zijn voorbeelden van verschillende filter over bepaalde plaatjes. We hebben deze plaatjes gebruikt omdat het resultaat bij de cirkel vrij opmerkelijk is en omdat de sudoku een zeer duidelijk resultaat heeft.



Opmerkelijk bij het resultaat van dit plaatjes is dat de filters een heel duidelijk beeld geven van de rand van de cirkel terwijl dat voor het blote oog in eerste instantie misschien niet zo duidelijk zal zijn. Verder is opmerkelijk dat bij dit plaatje het verschil tussen de methode van Gauss en die van Prewitt vrij

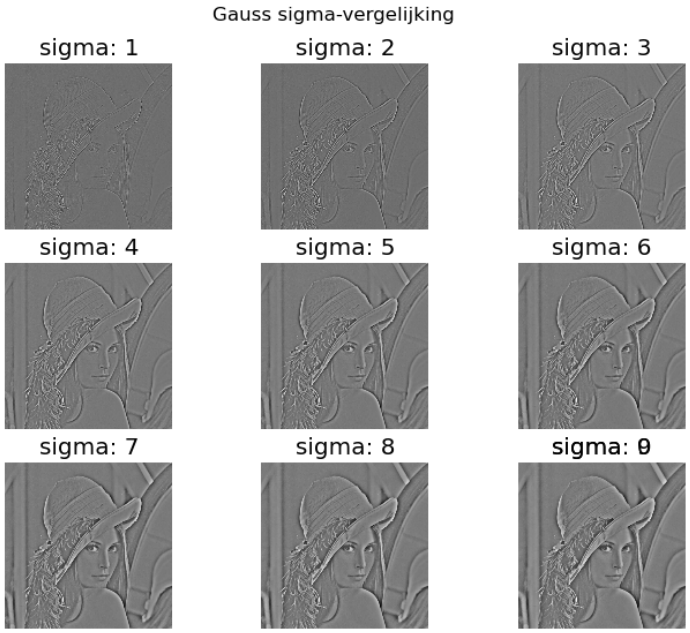
minimaal is. Het duidelijkste beeld wordt hier gegeven door de methode van Sobel, het negatief en de duidelijk scheiding van zwart en wit zorgt ervoor dat er duidelijk een rand kan worden waargenomen.



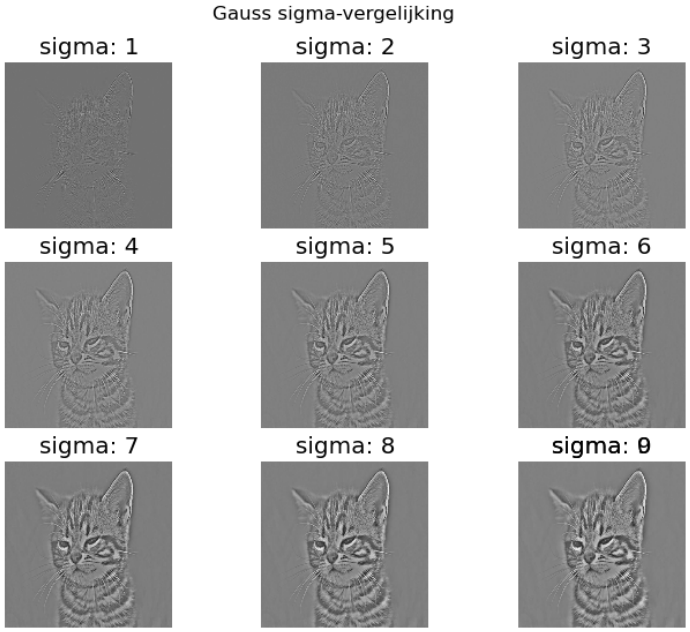
Bij dit plaatje lijken de methodes van Gauss en Prewitt minder op elkaar. Bij de methode van Gauss ontstaan er namelijk nieuwe opvallende lijnen op vrij merkwaardige posities. Verder komen de getallen bij de methode van Laplace er niet goed doorheen. De methode van Sobel geeft wederom het beste resultaat. Ten slotte is de methode van Prewitt ook vrij effectief, zoals aan de vectoren te zien is zijn de meeste randen goed gedetecteerd.

3.1.2 Standaardafwijking Gauss-methode

De volgende twee plaatjes zijn voorbeelden van de experimenten die we hebben gedaan met de Gauss-methode. Door de standaardafwijking te veranderen krijgen we meer of minder duidelijke randen. Door hier mee te spelen krijgen we verschillende beelden te zien.



Zoals bij deze afbeelding goed te zien is hangt de hoeveelheid detail sterk af van de standaardafwijking. Bij de eerste afbeelding is er zeer weinig detail maar de randen zijn ook niet erg duidelijk. Wanneer we naar de standaardafwijking van waarde drie kijken zien we dat we hier wel een vrij goede balans hebben gevonden tussen zo min mogelijk detail doch zo veel mogelijk randen. Bij een standaardafwijking van waarde negen is te zien dat er weer dikke lijnen komen en verschil in kleur buitenom randen.

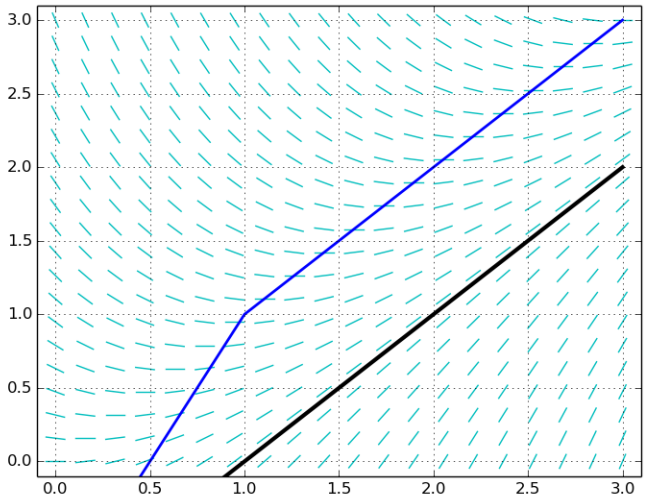


Deze afbeelding hebben we gekozen omdat deze zeer veel lijnen bevat. Hierdoor is het verschil in standaardafwijkingswaarden zeer goed zichtbaar. Wederom zijn de plaatjes met waarde een en twee net te wazig om duidelijk randen te onderscheiden, terwijl de hogere waardes juist teveel weergeven. Weer lijkt een standaardafwijking met een waarde rond de drie ideaal te zijn.

3.2 Lijnelementenveld en Eulermethode

We varieërden y_0, de en Δt.

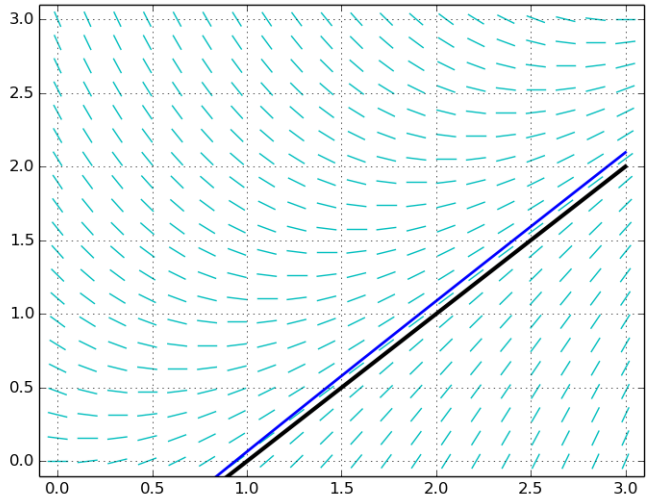
We testen allereerst de Euler methode met op $y_0 = -1$ en $\Delta x = 1$. We weten namelijk dat de lijn die we willen benaderen op tijdstip 0 op -1 begint. Dit levert het volgende resultaat:



Met $y_{benaderd} = y_b$ met $\Delta x = 1$
Met afbreekfouten:

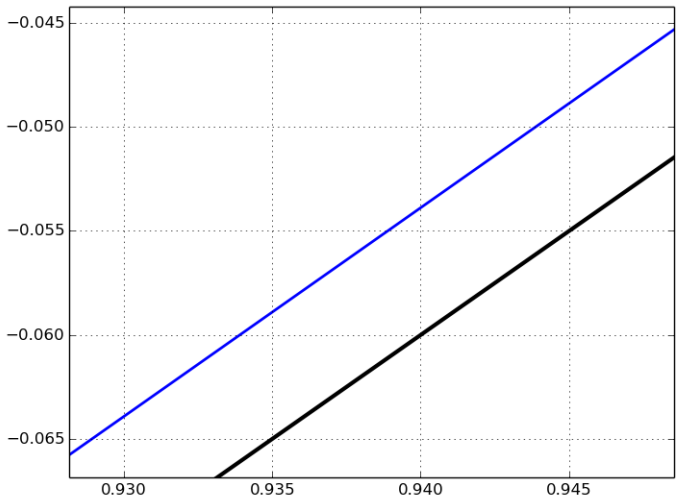
x	0	1	2	3
y	-1	0	1	2
y_b	-1	1	2	3
fout	0	1	1	1

Als we vervolgens $\Delta t = 0.1$ kiezen krijgen we het volgende resultaat:



x	0	1	2	3
yb	-1	0.06513216	1.08784233	2.09576088
fout	0	0.06513216	0.08784233	0.09576088

Met $\Delta t = 0.01$ krijgen we een figuur waar de lijnen nagenoeg op elkaar liggen. Als we inzoomen zien we echter dat de lijnen niet precies op elkaar liggen, en er dus afbreekfouten zijn:



x	0	1	2	3
yb	-1	0.00633968	1.0086602	2.00950959
fout	0	0.00633968	0.0086602	0.00950959

De fout lijkt een lineair verband te hebben met Δt, dus nemen we tot slot een Δt van 0.001. De afbeelding is niets zeggend, de tabel wel:

x	0	1	2	3
yb	-1	0.00063230	1.00086480	2.00095029
fout	0	0.00063230	0.00086480	0.00095029

4 Discussie en Conclusie

4.1 Randherkenning

4.1.1 Filtervergelijkingen

Wij begonnen aan onze experimenten met de veronderstelling dat de methode van Gauss de meest effectieve manier van filteren voor randherkenning is. De eerste plaatjes die wij hadden gegenereerd zorgden voor deze veronderstelling, maar dit was voordat we goed naar de andere methodes hadden gekeken. Uit onze experimenten is gebleken dat de filters wel goed functioneren alleen krijgt die van Sobel onze voorkeur. De methode van Laplace zorgt voor vrij vage beelden waar de randen niet altijd even goed zichtbaar zijn. De methode van Gauss werkt over het algemeen prima alleen kan het resultaat soms onvoorspelbaar zijn. Bij de sudoku kregen we bijvoorbeeld een vreemde mix tussen witte en zwarte lijnen. Het resultaat van Prewitt is over het algemeen stabiel maar doet onder aan dat van Sobel in duidelijkheid en kwaliteit.

4.1.2 Standaardafwijking Gauss-methode

Uit de experimenten is duidelijk geworden dat de standaardafwijking goed gekozen moet worden voor het ideale beeld. Kleinere standaardafwijkingen zorgen voor een beeld waar te weinig uit afgelezen kan worden, dit komt doordat het beeld niet zo sterk vervaagd word. Wanneer je het vervaagde beeld vervolgens van het originele beeld aftrekt verdwijnen er dus ook veel pieken in grijswaardes. Bij een te grote standaardafwijkingen word het beeld juist teveel vervaagd. In feite zou je het originele beeld behouden als je het verschil berekent met een oneindig grote standaardafwijking. De ideale standaardafwijking ligt volgens onze experimenten rond de drie. Natuurlijk kan het per afbeelding of per gewenst resultaat verschillend zijn.

4.2 Lijnelementenveld Eulermethode

We zien dat wanneer we Δt met factor 10 verhogen, de afbreekfout met factor 10 verkleint. Er is dus sprake van een lineair verband tussen de afbreekfout en Δt. Dit was zoals verwacht, aangezien dit verband al af te lezen was in de formule van de voorwaartse Euler methode. We merken echter wel op dat hoe hoger de x-waarde, hoe hoger de afbreekfout wordt.