Project wavelets

Matthias Baeten & Bob Vergauwen

13 januari 2016

1 Ruisreductie

1.1 Academisch voorbeeld zonder ruis

Bij wijze van opwarming starten we met de wavelet decompositie van de functie $\mathbb{R} \to \mathbb{R} : x \mapsto \exp(x)$. Dit is een gladde functie die bovendien analytisch is. Voor onze analyse werden de de exponentiële functie equidistant bemonster op het interval $[0, 1]$ met 256 punten. Deze data werd nadien geanalyseerd met behulp van 3 verschillende wavelet transformaties, de haar wavelet, de daubechie wavelet van orde 4 en de daubechie wavelet van orde 45. Elk van deze transformaties werd uitgevoerd tot niveau 4, dit maakt dus dat het signaal zal worden opgesplitst ten opzichte van 5 verschillende basissen. De resultaten van dit experiment zijn samen gevat in Figuur 1. In de linker kolom van de figuur zijn de coefficienten van de transformatie uitgezet. In de rechter kolom is telkens de benadering van de exponentiële functie in elke basis uitgezet. Hierbij is de onderste curve de benadering in W_1, die daar boven de benadering in W_2 en zo voort. De bovenste grafiek is dan de benardering van de exponentiële functie in de ruimte V_4.

Wat meteen opvalt is dat de coefficienten van de lage frequenties (links in de coefficienten vector) het grootst zijn. Dit is volledig volgens de verwachting, de exponentiele funtie is een gladde functie en bevat dus voornamelijk lage frequenties. Een tweede bemerking is dat voor de hogere orde wavelets de coefficienten aan de randen groter worden. Dit is het gevolg van het breder worden van de wavelet, hierdoor zal het eind effect verstrekt worden.

Vervolgens kunnen we zien naar de kwaliteit van de benaderingen in de opeen volgde vector ruimtes, zoals gegeven in de rechter kolom. Hier is het duidelijk dat een hogere orde benadering niet meteen een snellere convergentie geeft. Dit is opnieuw het gevolg van het bredere karakter van de hogere orde wavelets. Over het algemeen is de beste benadering bekomen door de daubechie wavelte van orde 2. Het eind effect is het kleinste voor de haar wavelet.

1.2 Academisch voorbeeld met ruis

In een tweede test word er ruis toegevoegd aan de gladde functie exponentiële functie. Deze ruis is witte ruis met een standaard afwijking van 0.1. Om de invloed van de ruis op de wavelet coefficienten duidelijk te maken zijn de coefficienten weergegeven in figuur 2. De invloed van de witte ruis in het tijddomein geeft een verstoring van witte ruis op de coefficienten van de versrchillende wavelet transformaties. De verstoring kan makkelijk worden verwijderd aan de hand van een treshold waarder te gebruiken. Deze methode is besproken in de opgaven en zal dus niet verder worden toegelicht. Enkel de resultaten en toepassingen zullen worden besproken.

De fout als functie van de threshold waarden is weergeven in figuur 3 tot 5. Uit deze drie figuren is het duidelijk dat er een fundamenteel verschil optreed tussen de zachte threshold functie en de twee andere. De verklaring hiervoor is dat de zachte threshold functie elke waarden zal wijzigen, zelfs de waarden ver boven de threshold. Om dit te illustreren zijn de drie threshold functies weergegeven in Figuur 7.

Om dit deel af te sluiten is in figuur 6 de optimale ruis reductie weergegeven. Deze reducties maakt gebruik van daubechie wavelet van orde 2 en een threshold waarden van 0.4 met de zachte treshold functie.

Tussenliggende waarden bepalen ...Iets met de basis wavelet bepalen in het punt en zo kan je het doen. Ik denk dat dit iets te maken heeft met een wavelet interpolatie.

1.3 Moving on to images

1.3.1 Implementatie van ruis reductie algoritme

De eenvoudigste manier voor ruis uit een afbeelding te halen aan de hand van een wavelet transformatie is door exact de zelfde strategie toe te passen als in het 1 dimensionaal geval. Dit houd in dat eerst de wavelet coefficienten worden bepaald voor de ruisige afbeelding. Nadien worden deze coefficienten met een threshold functie op een niet lineaire manier gefilterd. De laatste stap is dan de afbeelding reconstrueren aan de hand van de gefilterde coefficienten. Een concrete implementatie van dit algoritme is terug te vinden in de appendix.

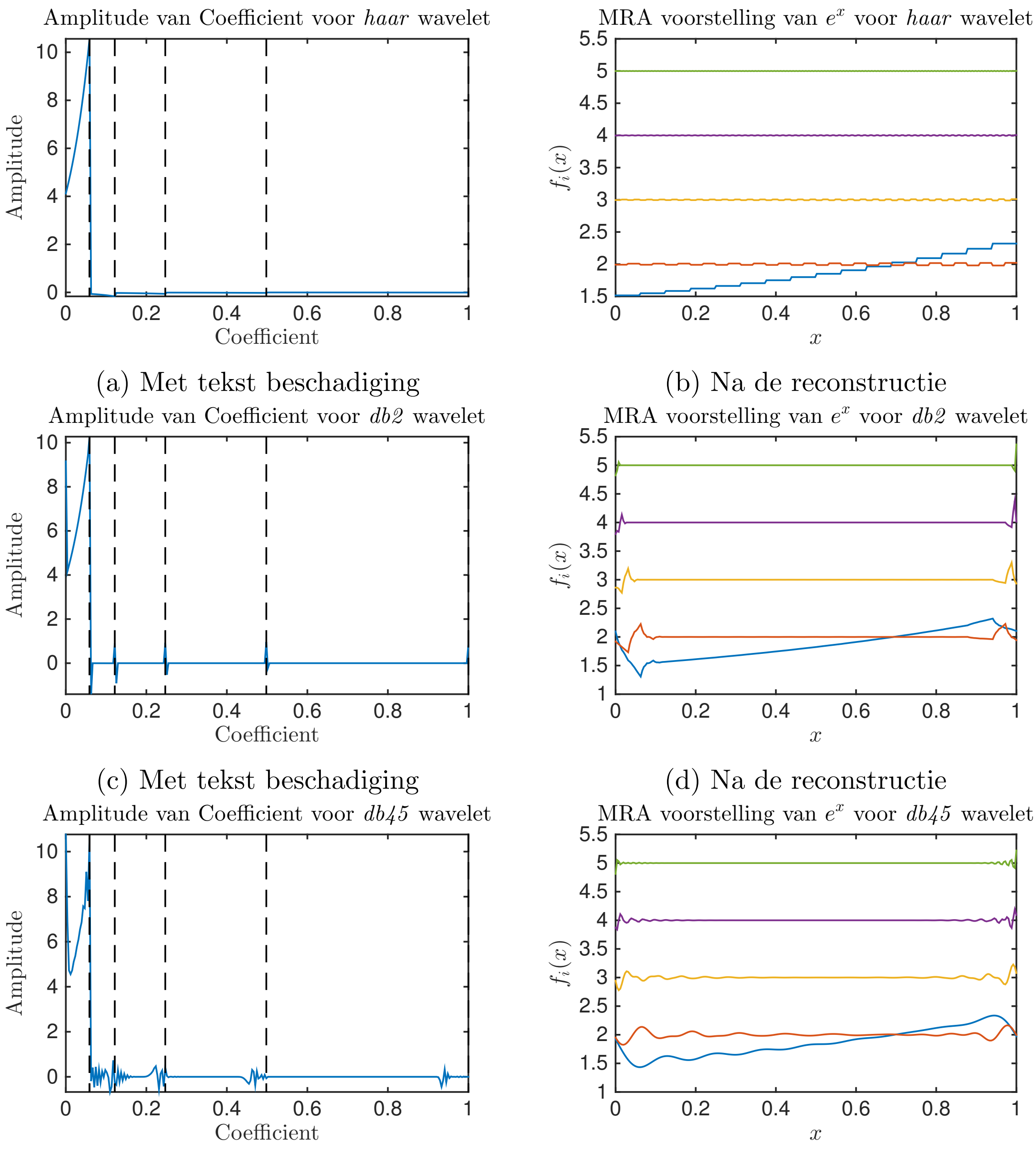
1.3.2 Verschil in threshold functies

Voor een goed beeld te krijgen van de invloed van de threshold functie op de ruisreductie hebben we de kwaliteit van de ruisreductie vergeleken voor de verschillende threshold functies. Voor elke threshold functie werden er een aantal threshold parameters getest. Een voorbeeld resultaat van zo een test is te zien in Figuur 8. Uit deze afbeelding is af te leiden dat zachte threshold functie het beste resultaat opleverd voor de ruisreductie. In Figuur 8 werd gebruik gemaakt van de biorthogonale wavelet van orde 6,8. Voor de meeste andere transformaties werden gelijkaardige resultaten bekomen. We kunnen dus besluiten dat de zachte threshold functie de beste ruisreductie oplevert.

1.3.3 Optimale threshold bepalen(met vals spelen)

Uit het vorige experiment hebben we kunnen besluiten dat in alle gevallen de zachte threshold functie de beste denoising geeft. Een tweede resultaat dat opviel was dat de SNR curves steeds vlakke curves bleken te zijn voor de zachte threshold functie. Door het gladde karakter van deze curve is het gebruik van een optimalisatie routine voor de SNR costfunctie makkelijk te implementeren. De cost functie is als volgt gedefiniëerd in matlab.

```
costFun = @(delta) -snr_denoising(mode, thres, delta, wname, ...
                   Nb_levels, A_origineel, A_noise,0);
```



(a) Met tekst beschadiging

(b) Na de reconstructie

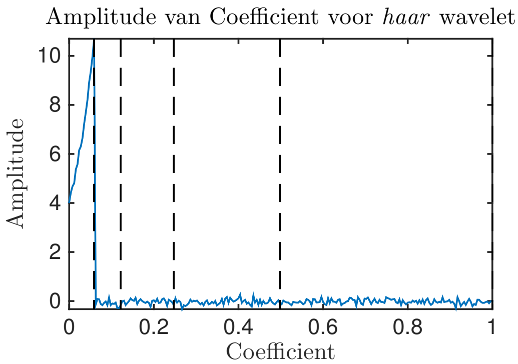
(c) Met tekst beschadiging

(d) Na de reconstructie

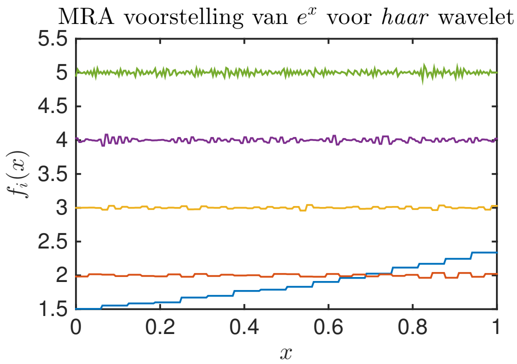
(e) Met tekst beschadiging

(f) Na de reconstructie

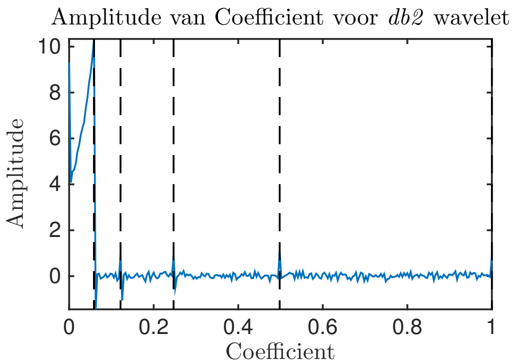
Figuur 1: Pictures of lena



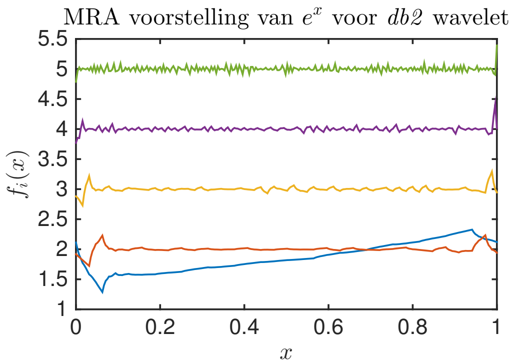
(a) Met tekst beschadiging



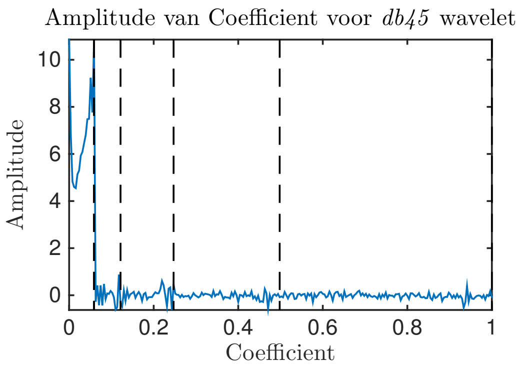
(b) Na de reconstructie



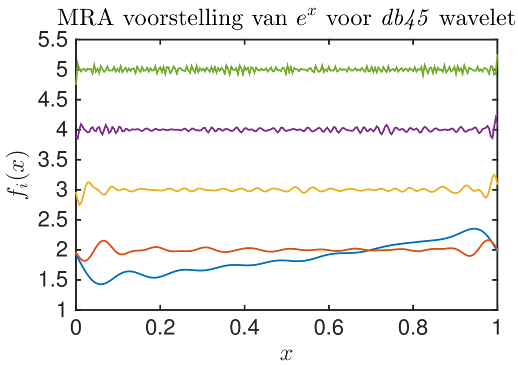
(c) Met tekst beschadiging



(d) Na de reconstructie

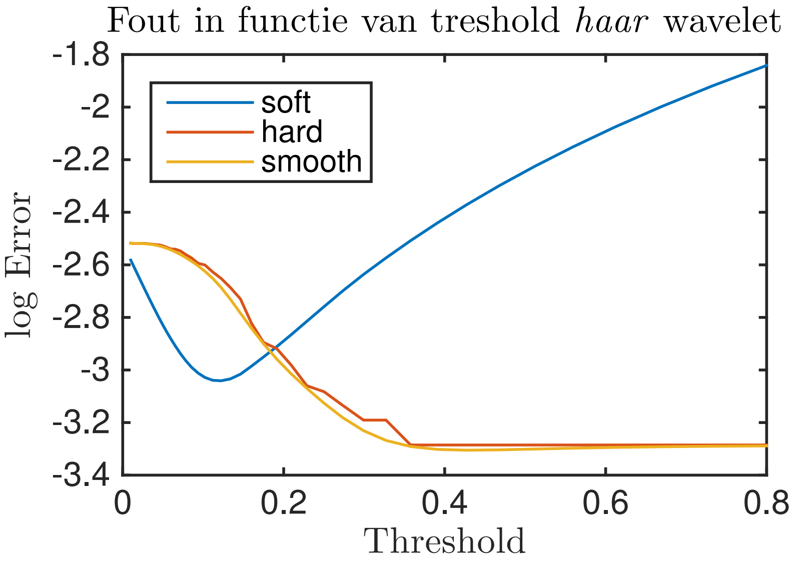


(e) Met tekst beschadiging

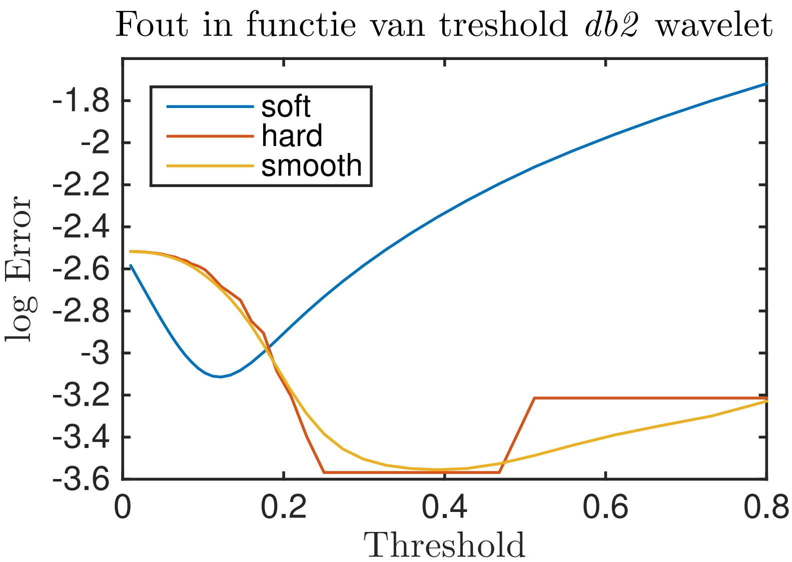


(f) Na de reconstructie

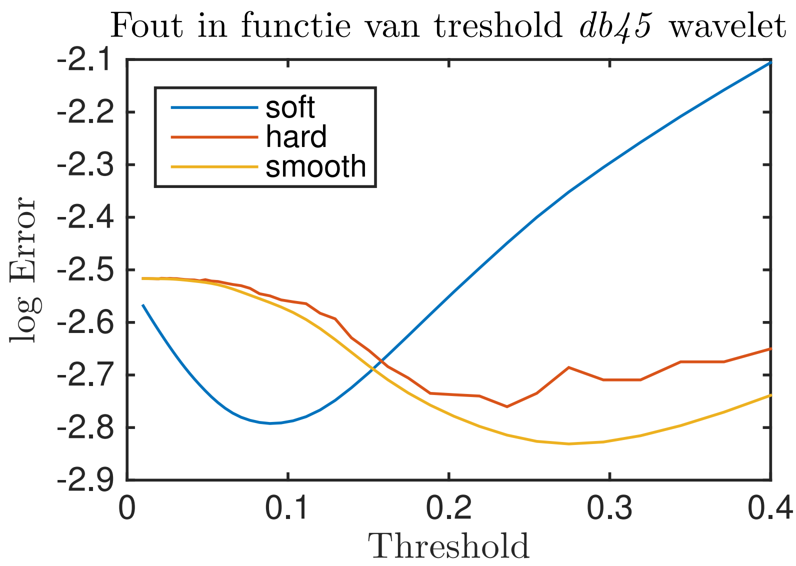
Figuur 2: Pictures of lena



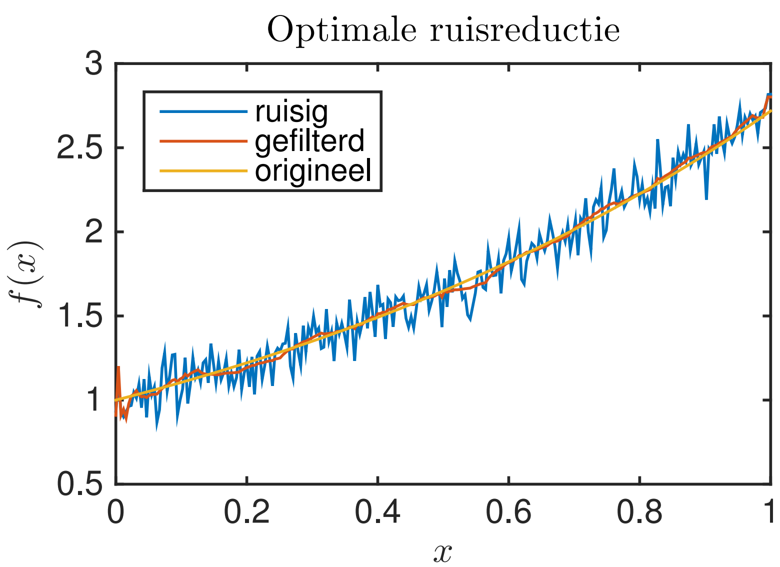
Figuur 3



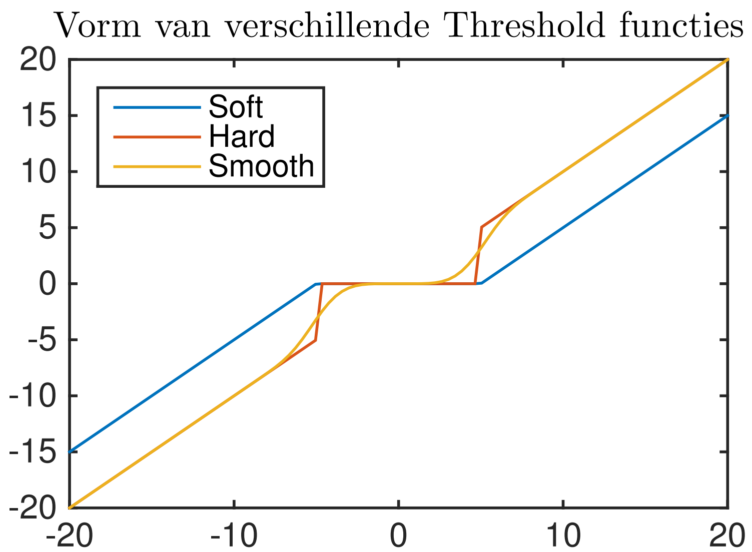
Figuur 4



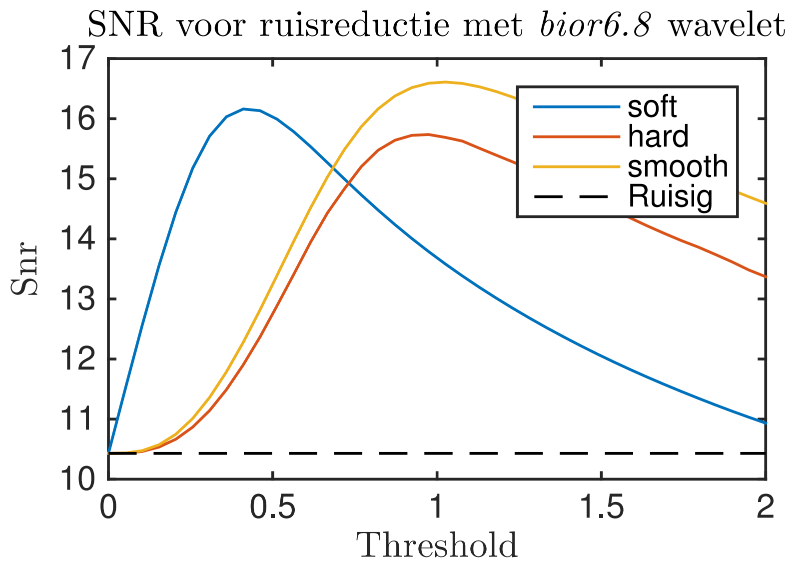
Figuur 5



Figuur 6



Figuur 7



Figuur 8

(a) Met tekst beschadiging (b) Na de reconstructie

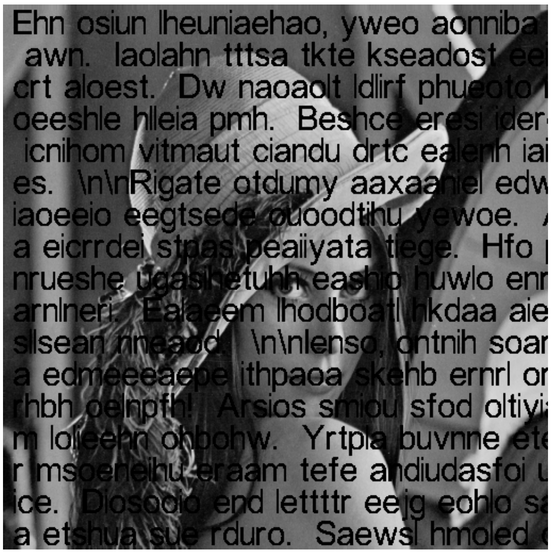
Figuur 9: Pictures of lena

Dit is een functie in de parameter `delta`. Dit is de waarden van de threshold. Merk op dat voor de berekening van de SNR waarden de originele afbeelding moet gekend zijn. (Vals spelen dus) Door gebruik te maken van `fmincon` is de keuze van de optimale parameter eenvoudig gemaakt.

Een voorbeeld resultaat van de optimale ruis onderdrukking is gegeven in figuur 9. In deze figuur is opnieuw de biorthogonale wavelet van orde 6,8 gebruikt.

1.3.4 Optimale threshold bepalen(zonder vals spelen)

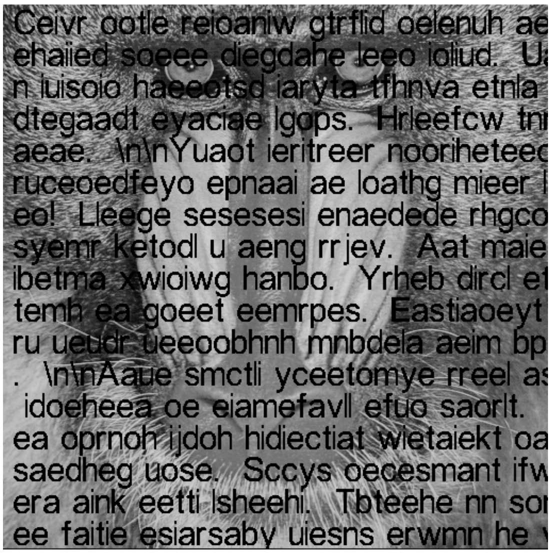
2 Inpainting



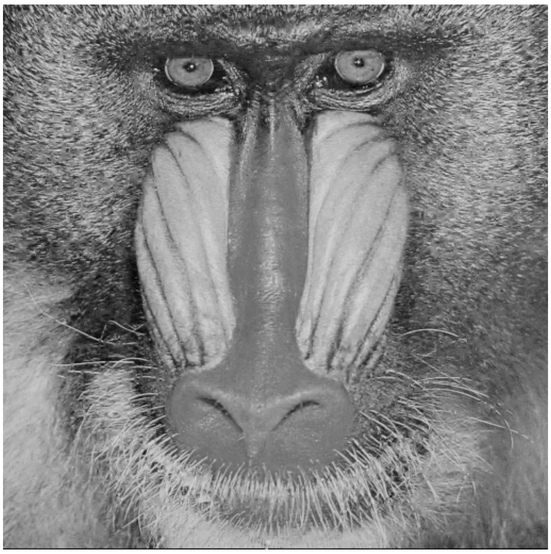
(a) Met tekst beschadiging

(b) Na de reconstructie

Figuur 10: Pictures of lena

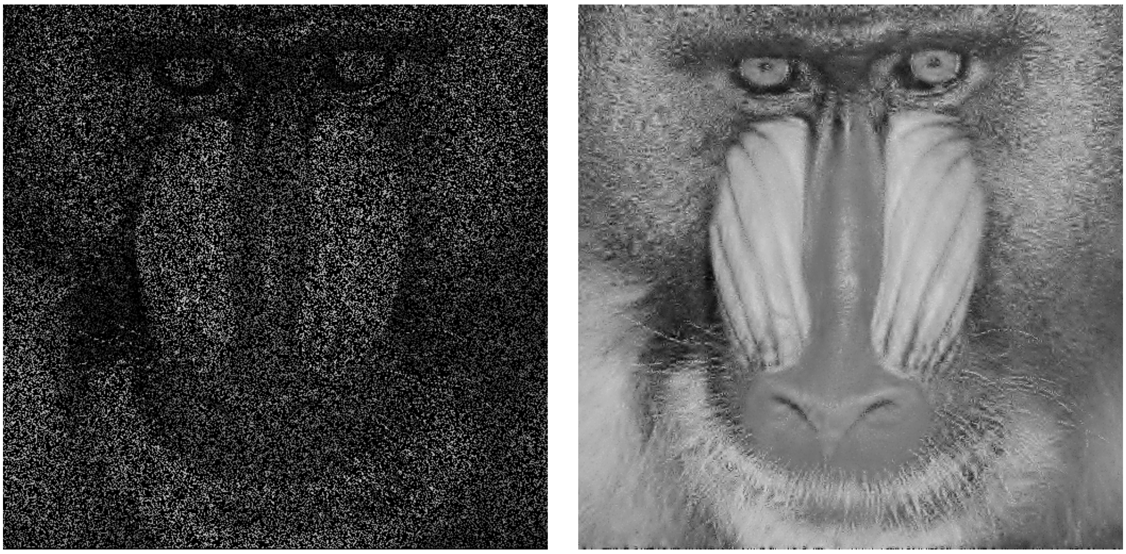


(a) Met tekst beschadiging



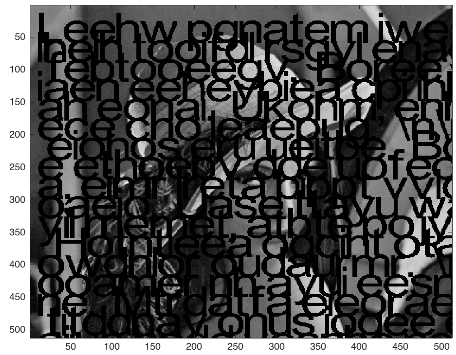
(b) Na de reconstructie

Figuur 11: Pictures of lena

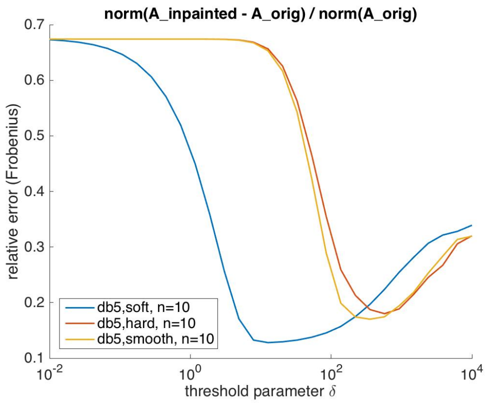


(a) Met random beschadiging (70 procent) (b) Na de reconstructie

Figuur 12: Pictures of lena

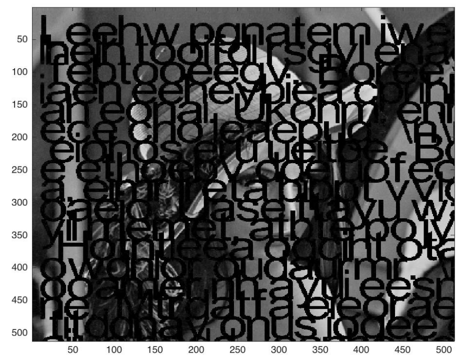


(a) tekstbeschadigde figuur.



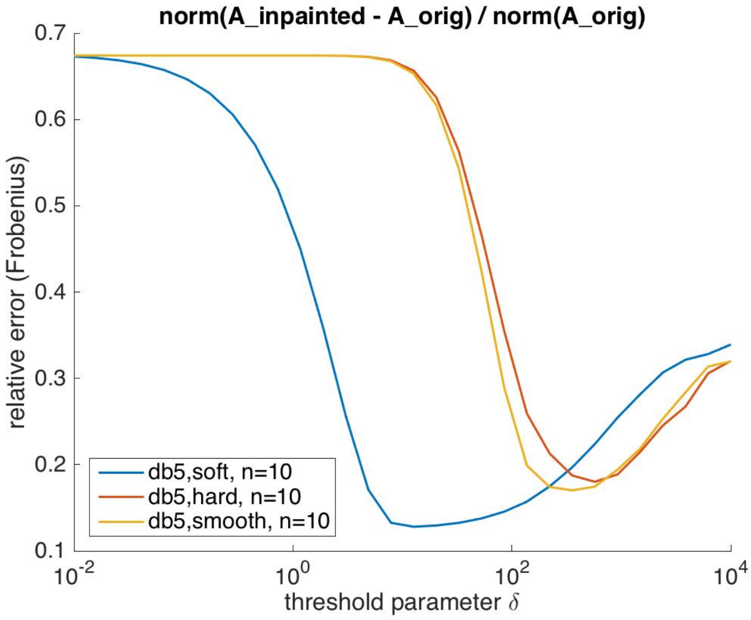
(b) Voor verschillende waarden van de threshold parameter δ is de figuur ingepaint met telkens 50 iteraties. De relatieve fout t.o.v. de onbeschadigde figuur is telkens berekent.

(c) figuur 13a ingepaint met 'db5' wavelets. Soft thresholding is gebruikt met parameter $\delta = 10$. Hier is het goed gelukt, de blauwe curve bereikt zijn mimimum rond $\delta = 10$.

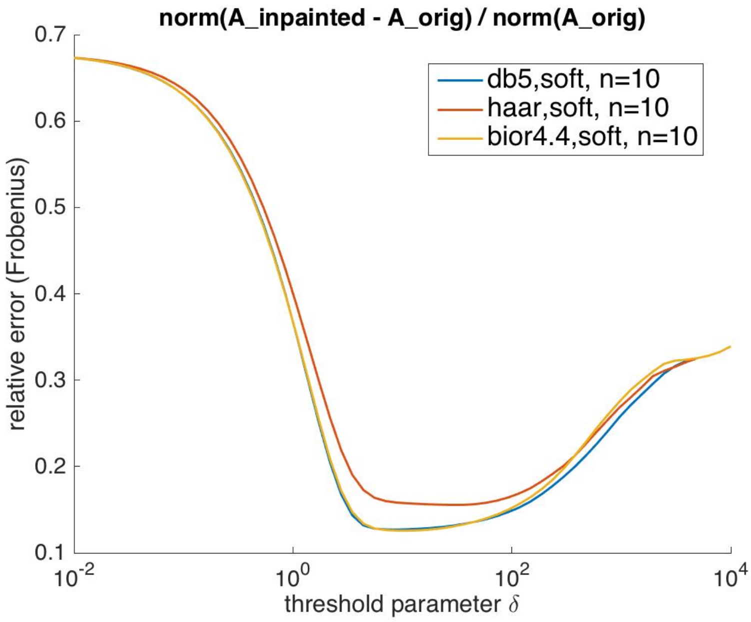


(d) figuur 13a ingepaint met 'db5' wavelets. Hard thresholding is gebruikt met parameter $\delta = 10$. Hier is het mislukt. De reden hiervoor is dat de rode curve voor $\delta = 10$ totaal niet het mimimum bereikt.

Figuur 13: Effect van threshold parameter en threshold techniek bij inpainting

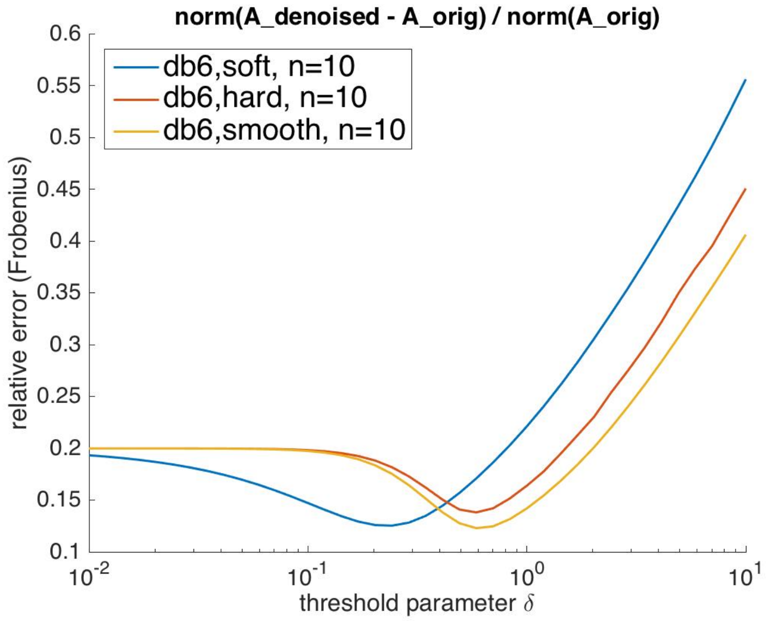


(a) Voor verschillende waarden van de threshold parameter δ is de figuur ingepaint met telkens 50 iteraties. De relatieve fout t.o.v. de onbeschadigde figuur is telkens berekent. verschillende threshold technieken zijn gebruikt.(dezelfde figuur als vorige pagina)

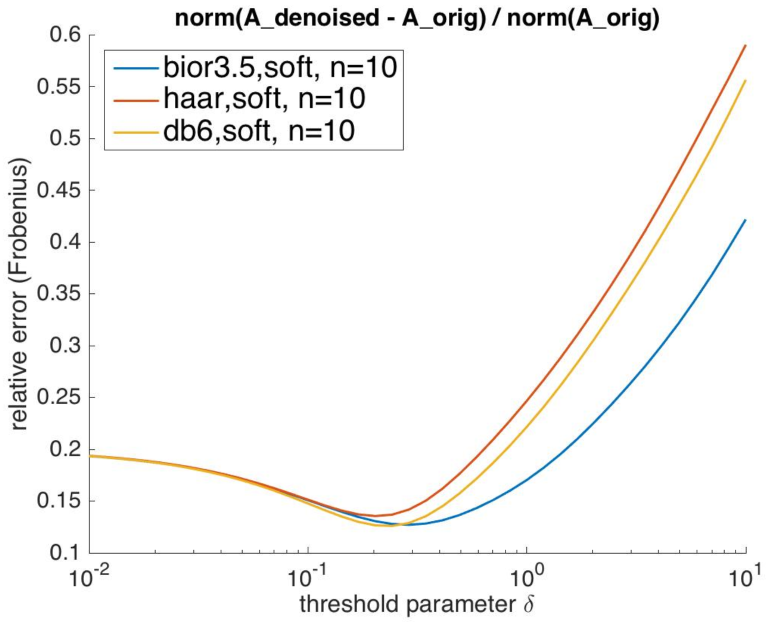


(b) Voor verschillende waarden van de threshold parameter δ is de figuur ingepaint met telkens 50 iteraties. De relatieve fout t.o.v. de onbeschadigde figuur is telkens berekent.verschillende soorten wavelets zijn gebruikt.

Figuur 14: plots error vs δ



(a) Voor verschillende waarden van de threshold parameter δ is een noisy figuur gedenoised. De relatieve fout t.o.v. de originele figuur(zonder noise) is telkens berekent. verschillende threshold technieken zijn gebruikt. De onbewerkte noisy figuur heeft een relatieve fout van 0.2. Conclusie: opletten met de waarde van de threshold parameter.



(b) Voor verschillende waarden van de threshold parameter δ is een noisy figuur gedenoised. De relatieve fout t.o.v. de originele figuur(zonder noise) is telkens berekent. De onbewerkte noisy figuur heeft een relatieve fout van 0.2.verschillende soorten wavelets zijn gebruikt. Conclusie: opletten met de waarde van de threshold parameter..

Figuur 15: plots error vs δ