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

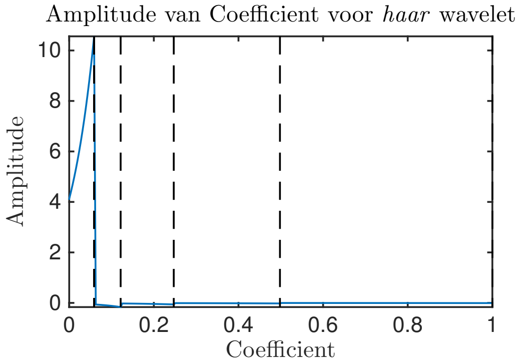
Voor de ruis experimenten op afbeelding is er steeds met een zelfde ruis gewerkt. De ruis was gaussische verdeeld met een standaard afwijking van 0.1. Verder werd de ingelezen afbeelding eerst genormaliseerd. Op deze manier is het eenvoudiger resultaten tussen verschillende afbeeldingen te vergelijken en een normalisatie komt ook meestal ten goede van de conditionering.

1.3.1 Implementatie van ruis reductie algoritme

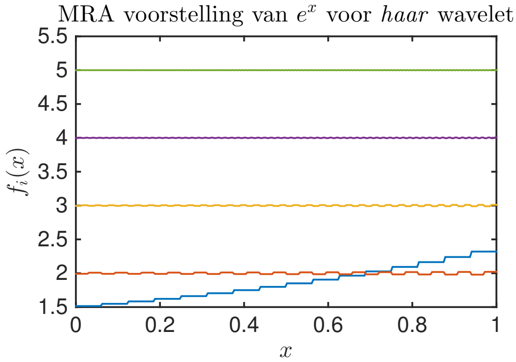
De eenvoudigste manier voor ruis uit een afbeelding te halen aan de hand van een wavelet transformatie is door exact de zelfde strategie toe te passen als in het 1 dimensionaal geval. Dit houd in dat eerst de wavelet coefficienten worden bepaald voor de ruisige afbeelding. Nadien worden deze coefficienten 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

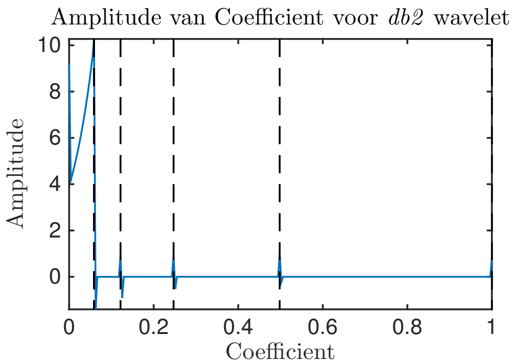
Voor een goed beeld te krijgen van de invloed van de threshold functie op de ruisreductie hebben we de kwaliteit van de ruisreductie vergeleken voor de verschillende threshold functies. Voor elke threshold functie werden er een aantal threshold parameters getest. Een voorbeeld resultaat van zo een test is te zien in Figuur 8. In deze figuur is de SNR waarden geplot voor verschillende threshold waarden en voor verschillende threshold functies. 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. Uit alle tests besluiten we dat de zachte threshold functie de beste ruisreductie oplevert.



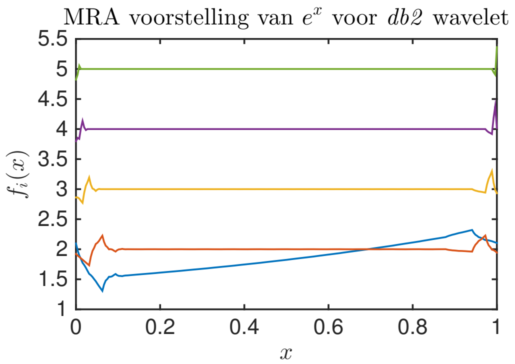
(a) Met tekst beschadiging



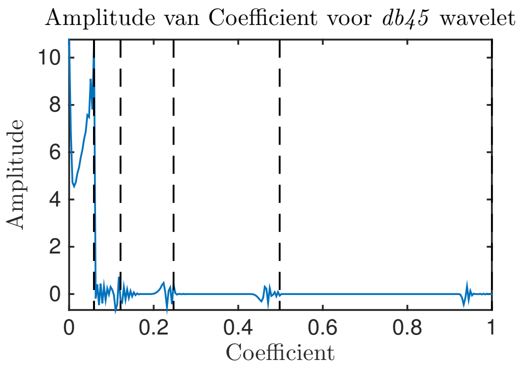
(b) Na de reconstructie



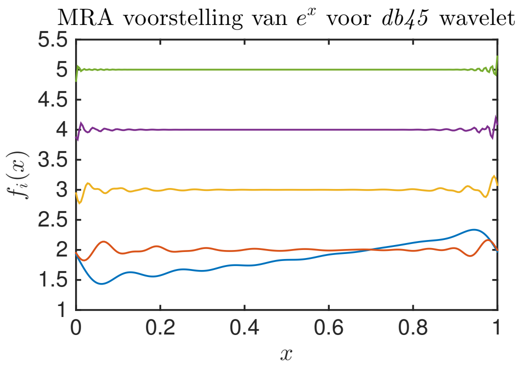
(c) Met tekst beschadiging



(d) Na de reconstructie

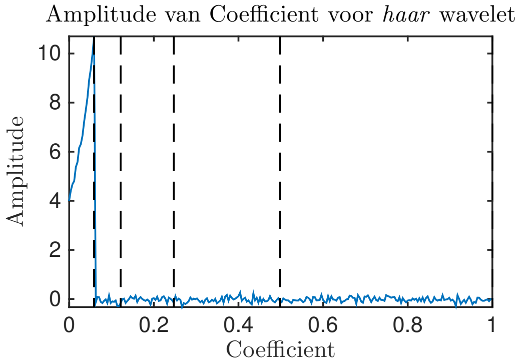


(e) Met tekst beschadiging

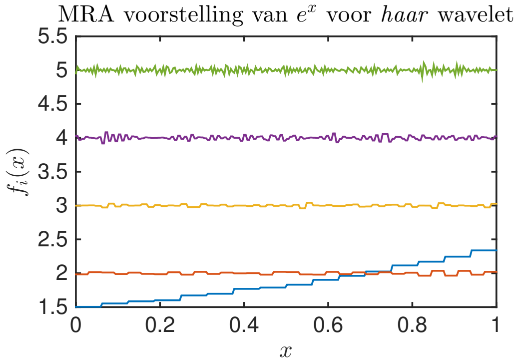


(f) Na de reconstructie

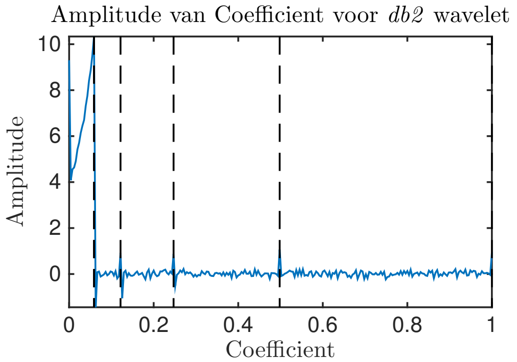
Figuur 1: Pictures of lena



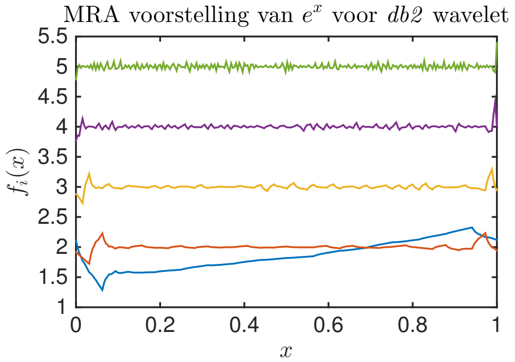
(a) Met tekst beschadiging



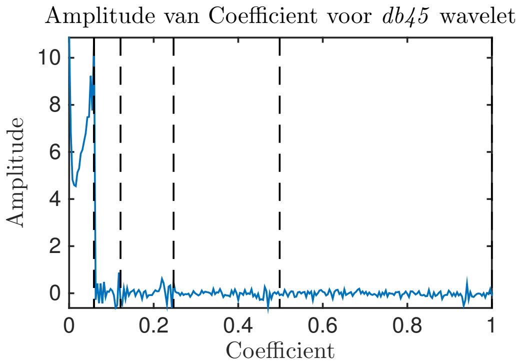
(b) Na de reconstructie



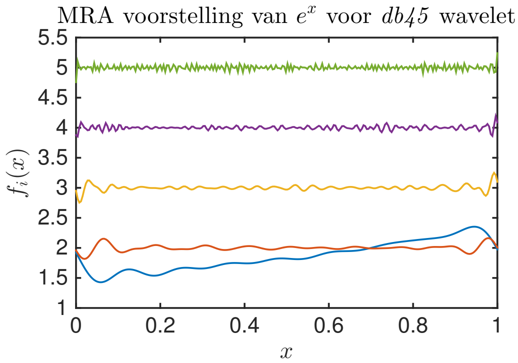
(c) Met tekst beschadiging



(d) Na de reconstructie

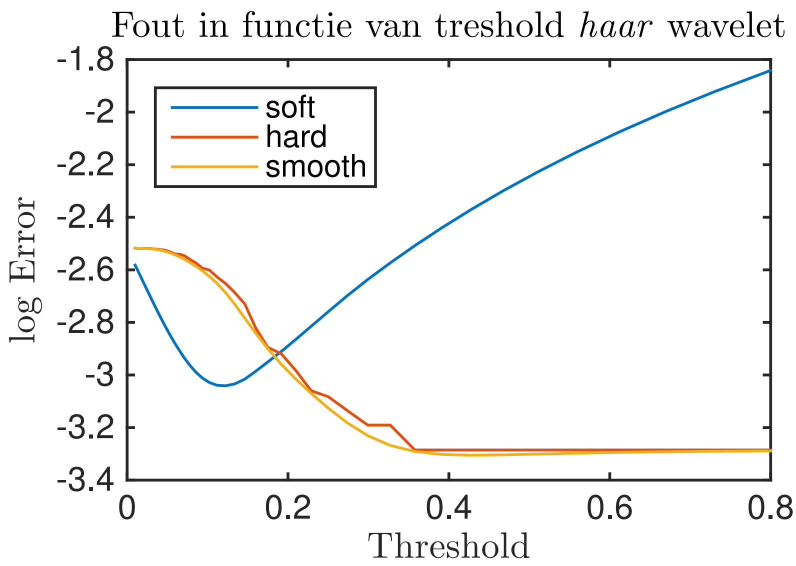


(e) Met tekst beschadiging

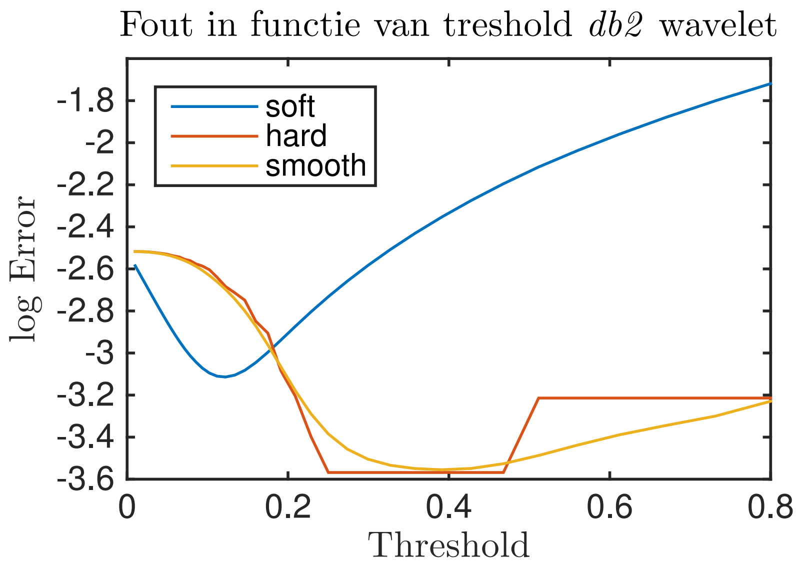


(f) Na de reconstructie

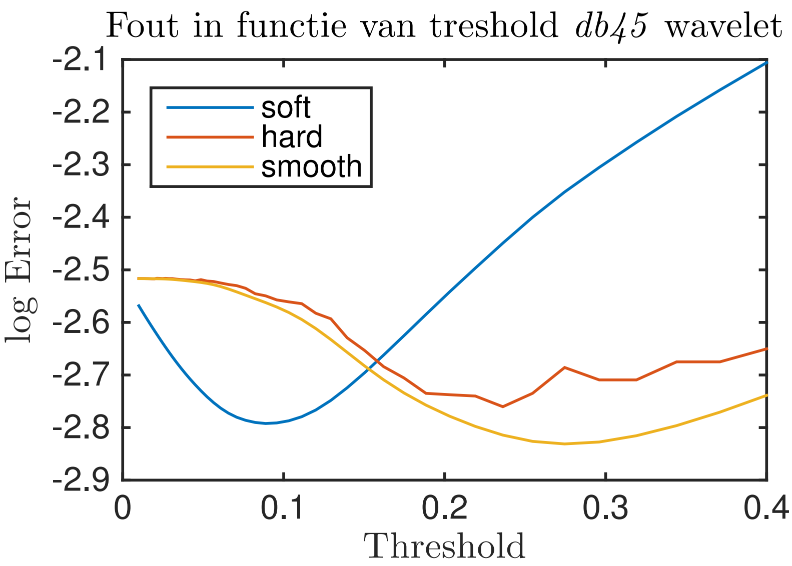
Figuur 2: Pictures of lena



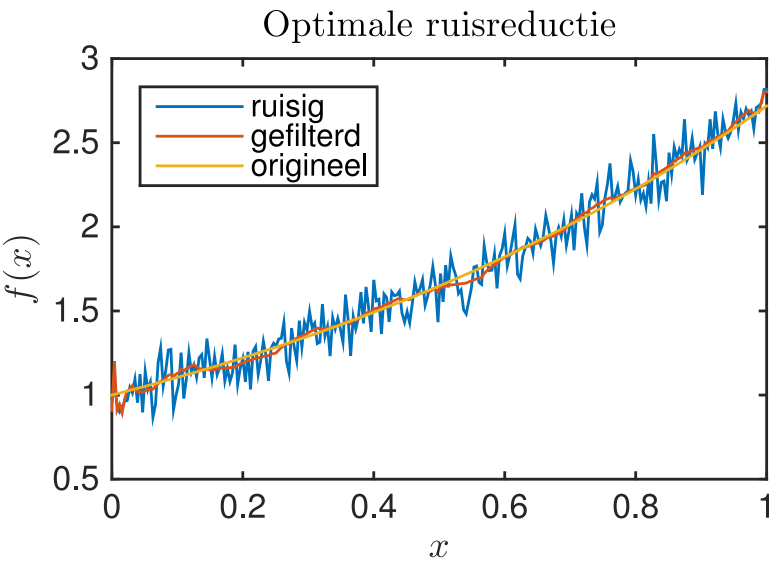
Figuur 3



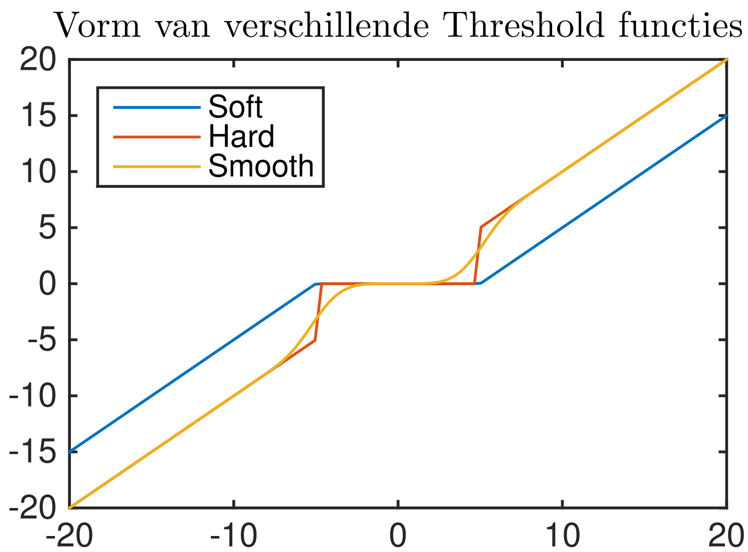
Figuur 4



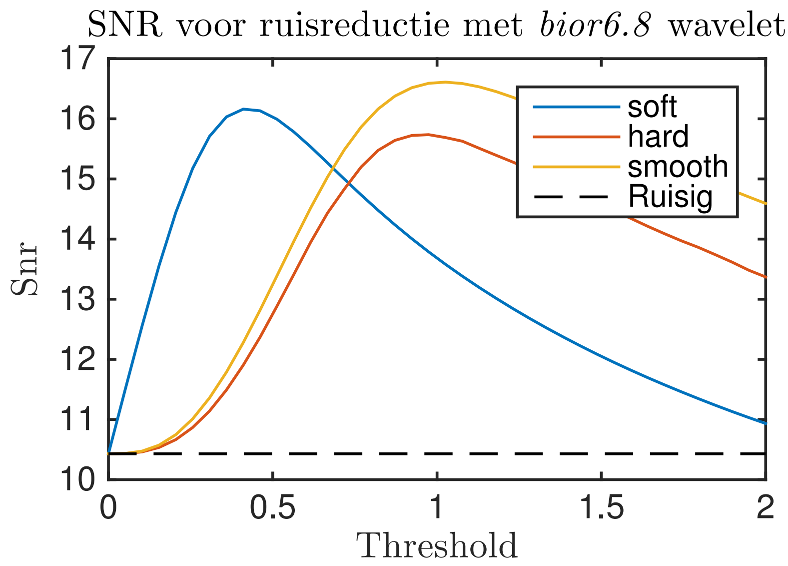
Figuur 5



Figuur 6



Figuur 7



Figuur 8

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

Figuur 9: Pictures of lena

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 gladde 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 = @(T) -snr_den(An,A,Nb_levels,wname,@(x) SmootThresh(x,T));
```

Dit is een functie in de parameter `T`, dit is de waarden van de threshold. `An` is de ruizige afbeelding, `A` de originele afbeelding, `Nb_levels` het aantal niveaus van de transformatie, `wname` de naam van de wavelte transformatie en `@(x) SmootThresh(x,T)` de threshold functie. Merk op dat voor de berekening van de SNR waarden de originele afbeelding moet gekend zijn (Vals spelen). Door gebruik te maken van de optimalisatie routine `fmincon` kan de optimale waarden voor de threshold snel worden gevonden.

```
[opt_thres,opt_snr] =fminunc(costFun,0.2);
```

Deze methode convergeerde steeds, soms was de convergentie naar een negatieve waarde van de threshold. Dit is geen probleem aangezien de threshold functie symmetrisch is in de threshold parameter.

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.

(a) Beschadigde foto met SNR van 8.30 dB (b) Reconstructie met SNR van 13.85 dB

Figuur 10: Resultaat voor reconstructie van ruizige foto aan de hand van *db6* wavelet met orde 5. Bij de reconstuctie werd er gebruik gemaakt van de redundante wavelet transformatie. Als threshold functie werd de zachte functie gebruikt. De threshold parameter werd optimaal gekozen aan de hand van een optimalisatie routine.

1.3.4 Optimale threshold bepalen(zonder vals spelen)

1.3.5 Beste strategie

1.4 Redundante wavelet transformatie

De redundante wavelet transformatie voor twee dimensionale foto's kan gebruikt worden via het commando `swt2`. Dit geeft als resultaat een drie dimensionale array terug met dimensies $H \times L \times N$, waarbij H, L respectievelijk de hoogt in pixels en de breedte in pixels van de foto is. De reconstructie van een ruizige foto kan met behulp van volgende twee lijnen code.

```
swc = swt2(X,N,wname);
Y = iswt2(thresHold(swc),wname);
```

Nadien kan de fout tussen de originele afbeelding X en de gereconstrueerde foto Y worden bepaald zoals voorheen. Dit kan nadien opnieuw worden gebruikt in een optimalisatie routine.

1.4.1 Resultaten

Het beste resultaat voor de denoising van afbeeldingen werd bekomen met de redundante wavelet transformatie. Als kost functie werd de afstand tot de originele afbeelding genomen. Als threshold functie werd er gekozen voor de zachte threshold. Hierbij werd de threshold waarden bepaald met behulp van `fmincon`. Op deze manier werd er een SNR van maximaal 13.85 bereikt waarbij de origine beschadigde foto een SNR van 8.30 dB had. Dit resultaat is weergegeven in Figuur 10

1.4.2 Rooster ruis

2 Inpainting

In dit hoofdstuk zullen we wavelets gebruiken om ontbrekende regio's in foto's in te kleuren. Deze methode wordt gebruikt om beschadigde foto's te reconstrueren. De schade op de foto wordt gemodelleerd met pixel waarden in de foto die verdwenen zijn. Om de verdwenen regio's in te kleuren wordt de iteratieve methode gebruikt die beschreven staat in de opgave. We hebben dit algoritme geïmplementeerd in Matlab met behulp van de Wavelet toolbox. Hierbij nemen we een bepaalde foto en verwijderen we de pixels in bepaalde regio's. Het algoritme probeert dan de verdwenen regio's in te kleuren.

In figuur 11 wordt het algoritme geïllustreerd met verschillende soorten regio's van pixels die verwijderd zijn. In figure 11a zijn er blokken pixels verwijderd, in figuur 11c zijn er random pixels verwijderd en in figuur 11e is de figuur overschreven met tekst. De resultaten van het 'inpainting' algoritme staan er steeds naast. Het algoritme geeft op het eerste zicht heel mooie resultaten. Alleen als we de ingekleurde resultaten in detail gaan bekijken merken we dat het niet de originele figuren zijn. Dit is het meest duidelijk bij de figuren die bewerkt zijn met blokken en met tekst.

Het 'inpainting' algoritme kan gebruikt worden op verschillende manieren. Zo zijn er verschillende soorten wavelets die gebruikt kunnen worden. De thresholding kan op verschillende manieren gebeuren en de threshold parameter δ moet gekozen worden. De effecten op het resultaat van al deze verschillende soorten instellingen zullen besproken worden in de volgende hoofdstukken.

2.1 Threshold technieken

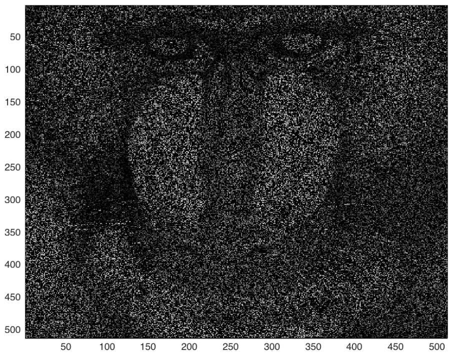
Er zijn 2 soorten technieken voor thresholding, namelijk soft thresholding en hard thresholding. Voor beide technieken moet ook een threshold parameter $\delta > 0$ gekozen worden. In figuur 12 is een figuur ingekleurd op twee manieren. Eén keer met soft thresholding en threshold parameter $\delta = 10$ en de andere keer met hard thresholding en threshold parameter $\delta = 100$. In het geval van soft thresholding was het gemakkelijk om een threshold parameter te vinden die redelijk goede resultaten geeft. Voor hard thresholding was het langer zoeken achter een geschikte threshold parameter $\delta = 100$. Dit was de meest optimale waarde van δ die ik op het eerste zicht kon vinden. Het is duidelijk dat voor soft thresholding betere resultaten kunnen bekomen worden als voor hard thresholding. Bij hard thresholding zijn er nog veel randen van letters zichtbaar. Bij soft thresholding zijn de resultaten veel beter.

Het is duidelijk dat het vinden van de optimale threshold parameter δ niet zo eenvoudig is. Daarom zullen we het volgende experiment uitvoeren. We voeren het 'inpainting' algoritme uit voor een reeks van verschillende waardes van δ. Voor elke waarde van δ zal het relatieve verschil tussen de ingekleurde foto en de originele onbeschadigde foto berekent worden. Dit verschil wordt voorgesteld op de volgende manier:

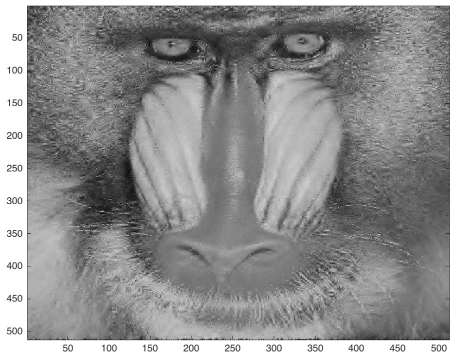
$$\frac{\|A_{\text{ingekleurd}} - A_{\text{onbeschadigd}}\|_F}{\|A_{\text{onbeschadigd}}\|_F} \tag{1}$$

(a) Foto van Lena met vierkante blokjes pixels verwijderd (zwart gemaakt).

(b) Foto 11a ingekleurd.



(c) Foto van aap met ongeveer 70 procent van de pixels verwijderd (zwart gemaakt).



(d) Foto 11c ingekleurd.

(e) Foto van lena overschreven met tekst.

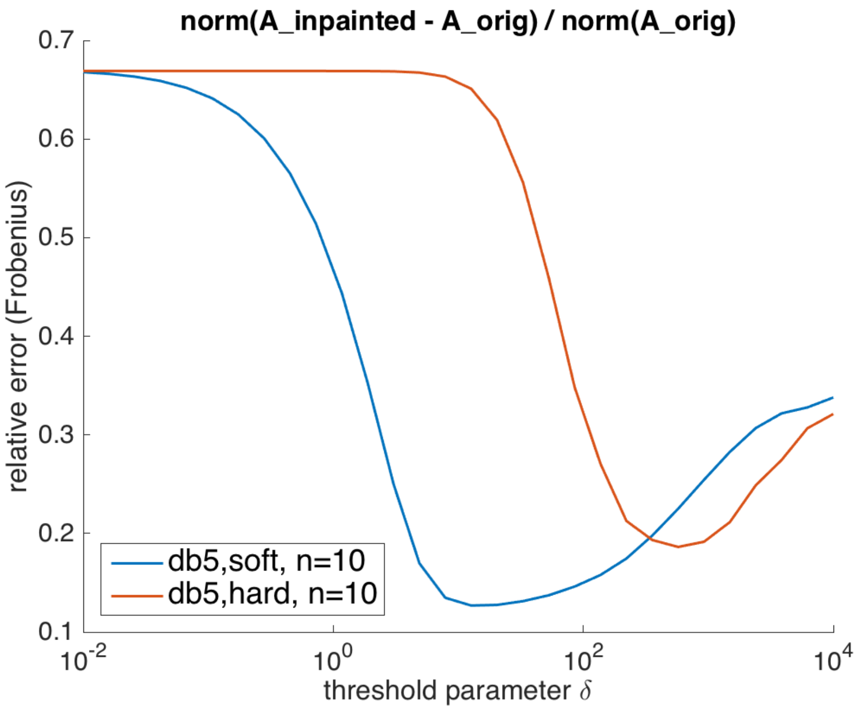
(f) Foto 11e ingekleurd.

Figuur 11: Resultaten van het 'Inpainting' algoritme. Instellingen algoritme: wavelet: 'db5', level $N = 10$, soft thresholding, threshold parameter $\delta = 10$, 200 iteraties.

(a) **Soft thresholding ($\delta = 10$).**

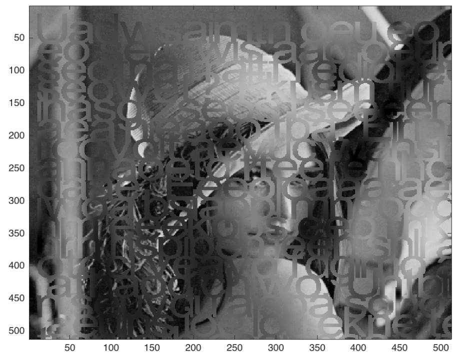
(b) **Hard thresholding ($\delta = 100$).**

Figuur 12: Met tekst overschreven figuur 11e ingekleurd met 'inpainting' algoritme. Instellingen algoritme: wavelet: 'db5', level $N = 10$, 200 iteraties.



Figuur 13: Het relatieve verschil beschreven (1) in functie van de threshold parameter δ voor zowel soft als hard thresholding. Instellingen algoritme: wavelet: 'db5', level $N = 10$, 50 iteraties voor elke waarde van δ. Het relatieve verschil tussen de onbeschadigde en de beschadige foto is ongeveer 0.67. De foto van lena overschreven met tekst van in figuur 11e is gebruikt.

(a) **Hard thresholding** met te lage waarde van $\delta = \mathbf{10}$. (volgens figuur 13)



(b) **Hard thresholding** met de optimale waarde van $\delta = \mathbf{1000}$. (volgens figuur 13)

Figuur 14: Met tekst overschreven figuur 11e ingekleurd met 'inpainting' algoritme. Instellingen algoritme: wavelet: 'db5', level $N = 10$, hard thresholding, 200 iteraties. De inkleuring in de linkse figuur is volleding mislukt vanwege een blijkbaar te lage waarde van δ.

met A de matrix met pixel waarden corresponderend met de foto. In figuur 13 kan men duidelijk het verschil zien tussen soft en hard thresholding. Soft thresholding bereikt een lager minimum rond de optimale waarde $\delta = 10$. Het minimum bij hard thresholding ligt iets hoger en wordt bereikt voor veel grotere waarden van δ. Opmerkelijk is dat hard thresholding het heel slecht doet indien de parameter δ relatief klein in tegenstelling tot soft thresholding. In figuur 14 is hard thresholding gebruikt met een te lage waarde van δ en de optimale waarde van δ (minimum rode curve figuur 13). Met $\delta = 10$ mislukt het inkleuren volledig. Dit was te verwachten als we kijken naar figuur 13. Volgens figuur 13 zou de optimale waarde van δ bij hard thresholding rond 1000 moeten liggen. Hoewel figuur 14b ($\delta = 1000$) niet heel slecht is, verkies ik persoonlijk toch figuur 12b ($\delta = 100$) boven figuur 14b alhoewel het relatieve verschil voor resultaat 14b minder is. In figuur 14b zijn nog heel duidelijk de letters te zien. De conclusie is dus dat men niet louter naar het minimale verschil in norm moet kijken maar ook naar het bekomen resultaat. Een tweede conclusie is dat soft thresholding het veel beter doet als hard thresholding voor meer verschillende ordes van δ. Soft thresholding heeft minder last met de randen van de letters in tegenstelling tot hard thresholding.

2.2 Redundant vs non-redundant wavelet transformaties

In dit hoofdstuk zullen we kort het verschil bestuderen tussen redundant en non-redundant wavelet transformaties. In figuur 15 zijn 6 ingekleurde figuren getoond. Hierbij zijn 3 soorten wavelets gebruikt en voor elke soort wavelet is de non-redundant en redundant wavelet transformatie uitgetest. Bij elke figuur is de corresponderende SNR waarde getoond. Het is duidelijk dat de bekomen resultaten veel beter zijn bij de redundant wavelet transformaties. Dit is zowel te zien in de figuren als in de SNR waardes. De 'overbodige' informatie dat de redundant wavelet transformatie gebruikt is blijkbaar toch niet zo overbodig in het geval van 'inpainting' toepassingen. Opmerkelijk is dat voor de biorthogonale spline wavelet gebruikt in figuur 15 het resultaat volledig mislukt is voor de non-redundant wavelet transformatie en voor de redundant wavelet transformatie het resultaat juist heel goed is. Dit is te zien in figuur 15e en figuur 15f.

Het enigste nadeel bij redundant wavelet transformaties is dat de rekentijd bij wavelet transformaties een stuk hoger ligt. Deze rekentijd bij redundant wavelet transformaties stijgt ook veel bij hoger gebruikte levels in de transformatie. In ons geval kon het iteratieproces in het 'inpainting' algoritme gemakkelijk 10 keer zoveel rekentijd vragen voor de redundant wavelet transformaties in vergelijking met de non-redundant wavelet transformaties.

2.3 Wavelet soorten

In dit hoofdstuk zal kort het verschil in de resultaten tussen verschillende type wavelets besproken worden. In figuur 16 zijn de 'inpainting' resultaten getoond voor 6 verschillende soorten wavelets. De SNR waarden zijn er steeds bij vermeld. In figuur 16a is de Haar wavelet gebruikt. In deze figuur zijn kleine blokjes te zien, Dit is omdat de Haar wavelet niet geschikt is voor continue overgangen. De corresponderende SNR waarde is 16.19. De 2 Daubechies wavelets in figuur 16b en 16c doen het beter als de haar wavelet. Dit is zowel te zien in de figuur als de SNR waarde. De reden hiervoor is dat Daubechies wavelets beter geschikt zijn voor continue overgangen. In figuur 16d is een biorthogonale spline wavelet gebruikt. In deze figuur is een heel slecht resultaat bekomen. De beste resultaten zijn bij de Coiflet en Symlet wavelet in figuur 16e en figuur 16f.

(a) **Non-redundant Haar wavelet, SNR = 15.96**

(b) **Redundant Haar wavelet, SNR = 18.15**

(c) **Non-redundant Daubechies (db5) wavelet, SNR = 17.97**

(d) **Redundant Daubechies (db5) wavelet, SNR = 19.61**

(e) **Non-redundant CDF (bior3.3) wavelet, SNR = 10.93**

(f) **Redundant CDF (bior3.3) wavelet, SNR = 20.50**

Figuur 15: Resultaten van het 'Inpainting' algoritme voor verschillende soorten wavelets. Instellingen algoritme: level $N = 6$, soft thresholding, threshold parameter $\delta = 10$, 200 iteraties.

(a) **Haar wavelet, SNR = 16.19**

(b) **Daubechies 'db2' wavelet, SNR = 17.73**

(c) **Daubechies 'db6' wavelet, SNR = 18.26**

(d) **CDF wavelet 'bior3.3', SNR = 11.92**

(e) **Coiflet wavelet 'coif4', SNR = 18.52**

(f) **Symlet wavelet 'sym5', SNR = 18.53**

Figuur 16: Resultaten van het 'Inpainting' algoritme voor verschillende soorten wavelets. Instellingen algoritme: level $N = 10$, soft thresholding, threshold parameter $\delta = 10$, 200 iteraties.