ISTANBUL TECHNICAL UNIVERSITY

FACULTY OF ELECTRICAL AND ELECTRONICS ENGINEERING

IMAGE INPAINTING

BSc Thesis by

Aziz KOÇANAOĞULLARI

040090356

Department : Electronics Engineering

Programme : Electronics Engineering

Supervisor : Associate Professor İlker BAYRAM

May 2014

ISTANBUL TECHNICAL UNIVERSITY

FACULTY OF ELECTRICAL AND ELECTRONICS ENGINEERING

IMAGE INPAINTING

BSc Thesis by

Aziz KOÇANAOĞULLARI

040090356

Department : Electronics Engineering

Programme : Electronics Engineering

Supervisor : Associate Professor İlker BAYRAM

May 2014

Special thanks to my supervisor İlker BAYRAM who has never spared his help, attention and valuable time, for providing me such an opportunity for this graduation project and for the idea of 'Image Inpainting'.

Also thanks to my entire family, especially to my mother, father and brother for the unending support and encouragement.

May 2014 Aziz KOÇANAOĞULLARI

Summary

Inpainting is the process of reconstructing lost or deteriorated parts of images or videos. Image inpainting is used wherever image reconstruction is required. For example, photo reconstruction, real world object removal, biological image restoration/inpainting. In each case, the goal is to merge inpainted region into the original image. In order to prevent typical user to be aware of any modification occurred.

The ancestral processors of inpainting operation are artists. Considering the damaged artistic landmarks the reconstruction is still carried on by professionals. But it is impossible to reconstruct each image of a video by hand. The aim of this project is to perform inpainting process automatically by using an algorithm. Image inpainting problem is explained with all steps for constructing a solution for the problem as well.

Denoising is another focus of this project. Denoising process can be considered as an inpainting since it aims to reconstruct the image. The difference is in denoising operation one has the knowledge of the form of the noise. As statistical measurements estimate properties of the noise, the algorithm that cleanses the image is highly dependent on this form.

Özet

İç boyama bozulmaya uğramış görüntü veya videoların onarımı işlemi olarak tanımlanır. Görüntü iç boyamadan, görüntü işlemenin, düzeltmenin gerekli görüldüğü her alanda yararlanılır. Fotoğraf onarımı, resimden obje çıkarılması, tıbbi sistemlerdeki görüntü düzeltme işlemleri kullanım alanına örnek olarak verilebilir. Verilen her durumda da esas olan amaç bozulmuş bölgenin onarımı ve görüntünün eski haline getirilmesidir. Bu işlemin kalitesi ise değiştirmenin ve bozulmanın mümkün olduğunca görsel olarak dikkat çekmemesi ile ölçülür.

Görüntü iç boyama tarih boyunca süre gelmiş bir işlemdir. Bu işlem geçmişte ressamlar tarafından gerçekleştirilmiştir. Günümüzde sanat eserleri göz önüne alındığında, bozulmaların onarılması hala uzmanlar tarafından yürütülmektedir. Videoları ele aldığımızda ise bu işlemin elle yapılması gereksiz zaman harcamasına sebebiyet verecektir. Bu nedenle bu işlemin otomatik olarak yapılmasına ihtiyaç duyulmaktadır. Bu tezde iç boyama işlemini otomatik olarak gerçekleştirebilecek bir algoritma tanımlanmış, algoritmanın oluşumu problem tanımından itibaren basamaklarıyla açıklanmıştır.

Gürültü azaltma ise tezde yer alan bir diğer odak noktasıdır. Gürültü giderme işlemi de iç boyamaya benzer bir amaç gütmektedir. Amacı görüntüyü daha iyi bir hale getirmek, onarmaktır. Aralarındaki fark ise gürültü giderme işleminde resim üzerindeki bozulmaların belli bir formda olması ve bu formun kestirilebiliyor olmasıdır. İstatistiksel olarak gürültü biçimi çok net bir biçimde belirlenebildiğine göre çözüm de bu yönde ilerlemelidir. Tezde çözümün basamakları incelenmiştir.

Contents

Chapter 1

Introduction

This BSc thesis includes solutions and examples on inpainting and denoising problems on images.

The second chapter defines the inpainting problem and investigates the theory. The problem is recovering damaged parts of an image using the undamaged parts. In order to solve this problem, one requires a discrete wavelet transform and an iterative algorithm to solve the minimization problem. In this chapter Douglas-Rachford algorithm is defined. Also required discrete wavelet transforms are introduced. The section also introduces that the Algorithm yields the result 'SOFT Thresholding';

$$m^* = \begin{cases} Wz+\lambda & \text{if } Wz \in (-\infty,-\lambda) \\ 0 & \text{if } Wz \in [-\lambda,\lambda] \\ Wz-\lambda & \text{if } Wz \in (\lambda,\infty) \end{cases}$$

The following two chapters include solutions of the inpainting problem in 1D and 2D. In 1D problem discrete cosine transform is used as the wavelet transform. In 2D Haar Wavelet is used as the transform. In 2D section the algorithm is simplified as;

- z_a=z_b $\leftarrow$ damaged image
- Repeat;
  - $z_a(damaged\,part)$=$z_b(corresponding)$
  - m_c = Threshold (DWT($2z_a - z_b$))
  - $z_b^* = IDWT(m_c) - (z_b - z_a)$
  - $z_b = z_b^*$
- Finish $z_{fin} = za$ with $z_a(damaged\,part)$=$z_b^*(corresponding)$

The fourth chapter also includes the effects of the variables. Each condition is explained in details with examples. To give a brief visualization of the process; There exists

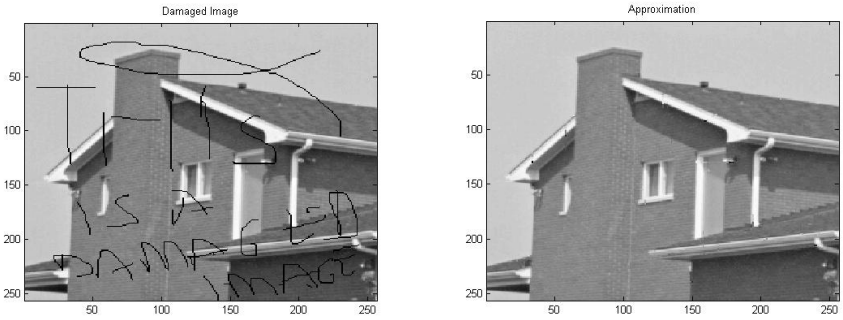


Figure 1.1: Image Inpainting

a GUI for the inpainting process where the user can determine operation variable values.

The fifth chapter is about 'Denoising' problem. In this section denoising solution yields SOFT thresholding under assuming that noise is in Gaussian form and signal is in Laplacian form. Threshold value λ is calculated by using SURE Estimation. This is a requirement to decide the quality of the result from the input as one has no information about the original form of the image. To visualize the process;

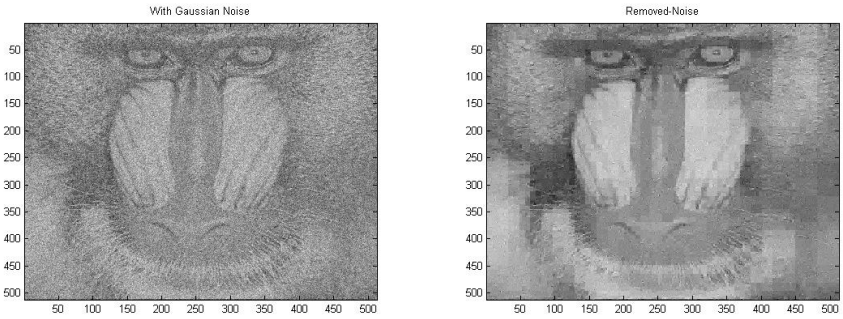


Figure 1.2: De-Noising Process

Chapter 2

Inpainting Problem

The inpainting problem involves the recovery of damaged parts of an image using the undamaged parts. There are two major points to be taken in consideration while modelling the problem [1];

Since one has the information of location of the damaged parts it forms a constraint for the problem. The damaged parts and the original ones should be treated differently.

Secondly, there are different orthogonal transformation matrices that yield sparse results for images and this characteristic is true for any image. By sparsity one should understand a data with mostly '0' values. This can be used for estimating the image using the deteriorated one. By using these two the inpainting problem can be formed as;

$$\arg\min_{z\in C}\|WI\|_1 \tag{2.1}$$

Where W:=discrete wavelet transform, I:= input image, P:= Index values of uncorrupted samples. We can define a new set of values using these indices (P).
$C = \{z : z(n) = x(n)\ for\ \forall n \in P\}$ Other elements are corresponding to the iteration vector.

The following sections include detailed explanation of the solution of the problem.

2.1 Douglas-Rachford Algorithm

The Algorithm [2] gives a solution for optimization problems in the form of;
$\arg\min_z f(z) + g(z)$

So the original inpainting optimization problem should be modified.The original problem is based on minimization of the l1 norm of the transformation of the given signal wrt.

the known elements;

$$\arg\min_{z\in C}\|Wz\|_1 = \arg\min_{z}\|Wz\|_1 + i_c(z) \tag{2.2}$$

Where the function $i_c(z)$ applies the constraint:$=\begin{cases} 0 & \text{if } z\in C \\ \infty & \text{if } z\notin C \end{cases}$

The optimization problem is in the desired form where f(z)= $\|\mathrm{Wz}\|_1$ and g(z)=$i_c(z)$. This problem should be solved by using the algorithm.

Douglas-Rachformd Algorithm is;

Repeat: $z \leftarrow J_f^{\lambda}\left(2J_g^{\lambda}(z)-z\right)+\left(z-J_g^{\lambda}(z)\right)$

Until: converge $z \leftarrow J_g^{\lambda}(z)$

Where $J_h^{\lambda}(z) = \arg\min_{u}\frac{1}{2\lambda}\|z-u\|_2^2 + h(u)$

First the J_g^{λ} should be solved. The minimization problem has the aim to minimize the distance between two points ($\|z-u\|_2$). Since the solution is highly dependent on $i_c(z)$ the solution should be separated into two parts;

$$u^* = \arg\min_{u\in C}\|z-u\|_2^2 \rightarrow \begin{cases} u^*(n) = x(n) & \text{if } n\in P \\ u^*(n) = z(n) & \text{if } n\notin P \end{cases}$$

In short; for the elements of damaged parts the function directly gets the values from the iterated vector.

Secondly J_f^{λ} is required to be solved;

$J_f^{\lambda} = u^* = \arg\min_{u}\frac{1}{2\lambda}\|z-u\|_2^2 + \|Wu\|_1$

Since W is orhagonal it doesnt change the distance between two vector values. In other words the l2 norm does not change. Because the logarithm is a strictly increasing function for positive real numbers. ($\|z-u\|_2^2 = \|Wz-Wu\|_2^2$) and with the change of parameter

$$m = Wu; \tag{2.3}$$

$$m^* = \arg\min_{m}\frac{1}{2\lambda}\|Wz-m\|_2^2 + \|m\|_1 \tag{2.4}$$

This directly corresponds to the 'Soft Thresholding Problem'. The solution of the given problem is;

$$m^* = (m|\frac{\partial f(m,z,\lambda)}{\partial m} = 0) = g(z,\lambda) \tag{2.5}$$

$$\begin{aligned} \frac{\partial\left(\frac{1}{2\lambda}(Wz-m)^2+m\right)}{\partial m} = 0 \ \rightarrow Wz = m+\lambda \mid m>0 \\ \frac{\partial\left(\frac{1}{2\lambda}(Wz-m)^2-m\right)}{\partial m} = 0 \ \rightarrow Wz = m-\lambda \mid m<0 \end{aligned} \tag{2.6}$$

Since the variable z is analogously m^*;

$$m^* = \begin{cases} Wz+\lambda & \text{if } Wz \in (-\infty,-\lambda) \\ 0 & \text{if } Wz \in [-\lambda,\lambda] \\ Wz-\lambda & \text{if } Wz \in (\lambda,\infty) \end{cases}$$

Since $m = Wu$, directly $u^* = W^T m^* = J_f^{\lambda}(z)$.

2.2 DWT (Discrete Wavelet Transform)

The optimization problem is highly dependent on a transform which satisfies the orthogonality condition and yields a sparse result. There should be an inverse transform W^T for receiving the image after operation as well where;
$W^T \times$ W = I(identity matrix)

As mentioned before there are variety of options. Hence it is enough to examine two possibilities of these transformations. The first 'discrete cosine transform' which is going to be referred in the **'1D inpainting'** section, the second 'haar decomposition' which is the decomposition applied for **'image inpainting'**.

2.2.1 Sinusoidal Decomposition

Sinusoidal decomposition [6] is nothing but decomposing the signal into periodic functions and the most common periodic functions are sinusoidal .

Sinusoidal decomposition can be explained by using Fourier transform. Fourier transform for discrete signals is defined as;

$$X(k) = \sum_{n=-\infty}^{\infty} x(n)e^{-jwn} \tag{2.7}$$

Where $w = \frac{2\pi k}{N}$ N:=length of the signal

Instead of decomposing the signal using complex exponentials we can use real valued periodic functions as operators. Sine and cosine functions are suitable since they form an orthonormal basis as their inner product over one period is '0' and their magnitude is '1'. It stays the same for any harmonic.

e.g.: $\int_{t=-\frac{T}{2}}^{\frac{T}{2}} cos(nwt)sin(mwt)dt = -\frac{cos\left(wt(m-n)\right)}{2w(m-n)} - \frac{cos\left(wx(m+n)\right)}{2w(m+n)}\Big|_{-\frac{T}{2}}^{\frac{T}{2}} = 0$

since m and n are integer values and cos(-x)=cos(x)

The sinusoidal representation of an periodic signal is;

$f_n(t) = a_0 + \sum_{n=1}^{\infty} a_n cos(nw_0t) + b_n sin(nw_0t)$ where $w_0 = \frac{2\pi}{T}$

The construction can be represented in matrix notation;

$f(n) = W_{NxN} \times Coefficents_{Nx1}$ where;

$W = [f, g]$ where the elements are defined as;

$f_k(n) = sin\frac{2\pi}{N}kn \;\; k \in \left[1, \frac{N}{2}\right]$

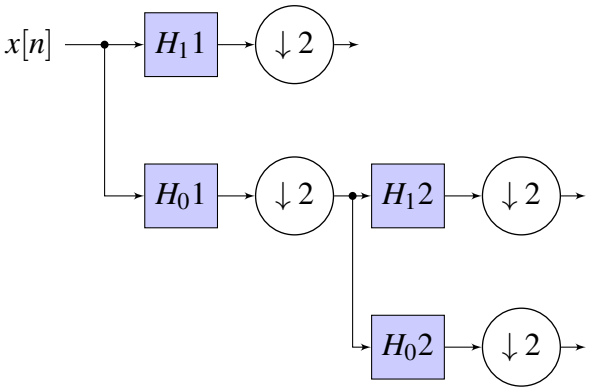
$g_k(n) = cos\frac{2\pi}{N}kn \;\; k \in \left[0, \frac{N}{2} - 1\right]$

It is clear that as the n $\in$ N the W matrix will be NxN.

2.2.2 Haar Wavelet

Haar Decomposition

Haar decomposition is used for splitting the data into two parts, high-pass coefficients and low-pass coefficients. The important part is the length of the input matrix always divided by '2' after each operation. Consider c[n] as approximation and d[n] as details. It can be clearly said that size(merge(c[n],d[n])=size(x[n]). It doesn't matter how many steps are used. The merged data of all decompositions is the same size as the input. Haar filter bank is shown below [10];



$$H0 = \frac{x[n] + x[n-1]}{\sqrt{2}} \quad H1 = \frac{x[n] - x[n-1]}{\sqrt{2}} \tag{2.8}$$

The process is applied on both vertical and horizontal directions one after the other. Discrete wavelet transform is the 2D application of 'Haar' wavelet. The definitions of the Haar Decomposition and the Discrete Wavelet Transform are;

```
function [ c,d ] = HaarAnalysis( x )
% Input variables
% x : input signal
% Output Variables
% c : Lowpass coefficients
% d : Highpass coefficients

if mod(length(x),2),
    x = [x 0];
end

c = (x(1:2:end) + x(2:2:end))/sqrt(2);
d = (x(1:2:end) - x(2:2:end))/sqrt(2);
end
```

It's clear that analysis operation can be defined as elementary additions and subtractions where functions are just a linear operation between coefficients and the signal values (↓2 in analysis is included.).

```
function [ II , step ] = DWT( I , step )
%I     := Input signal (uint8)
%steup := number of steps
%II    := Output signal (double)

%c  := Lowpass coefficents
%d  := Highpass coefficents

%Conversion to double required
%uint8 varies between 0-255 which can't hold negative numbers
%negative numbers are recieved while calculating d(highpass coef.)
I=double(I);
```

```
13
14 [m,n]=size(I);
15
16 %number of steps
17 for(x = 0 : step-1)
18
19     %Rowwise analysis
20     for(i=1:m/(2^x))
21
22             [c,d]=HaarAnalysis(I(i,1:m/(2^x)));
23             I(i,1:m/(2^x))=[c d];
24     end
25
26     %Columnwise analysis
27     for(i=1:n/(2^x))
28
29         [c,d]=HaarAnalysis(transpose(I(1:n/(2^x),i)));
30         I(1:n/(2^x),i)=[transpose(c);transpose(d)];
31     end
32 end
33
34     II=I;
35
36 end
```

It should be proven that the decomposition matrix 'W' is orthogonal and yields sparse results. The conditions are required for solving the inpainting optimization problem using Douglas Rachford algorithm. The conditions are checked for DWT with '5' steps (It's possible to run the algorithm over the same image for several times.).

The first condition required for W is to be orthogonal. DWT is orthogonal and can be checked by using the MATLAB code;

```
1 % Creation of the 128x128 DWT matrix of '5' steps
2 test = DWT(eye(128),5)
3
4 %sum(<1st col, 2nd col>)
5 sum(test(:,1).*test(:,2));
6
7     ans =
8         0
```

This also can be measured by checking total power values. Since the transformation is orthonormal it applies no change to the power of the signal;

$$X \sim \mathcal{N}(10, 10^2) \quad Y_{6X6} = DWT((X_{6X6}), 1)$$
$$PX \text{ and } PY := powers$$

```
1  X=10*randn(6,6)+10;
2  Y=DWT(X,1);
3  PX=sum(X(:).^2);
4  PY=sum(Y(:).^2);
```

DWT is applied just for one step. This is because the length of the signal is 6 where the c and d matrices have length 3 in one operation and it can't be divided into two;

```
PX = 8.2308e+003
PY = 8.2308e+003
```

clearly shows that their powers are equal.

Since the Wx transformation has to be 'sparse', DWT should satisfy the same condition as well. To visualize;

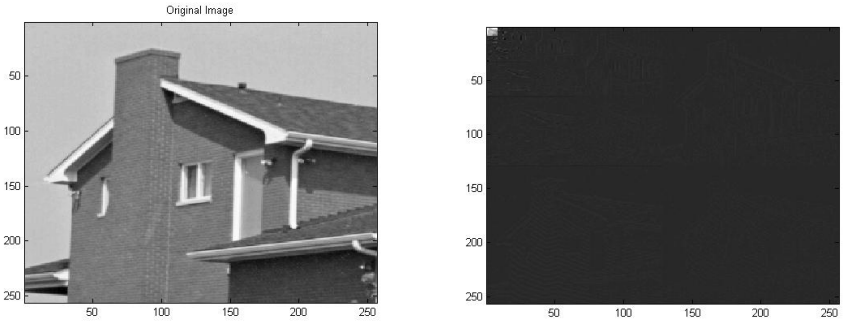
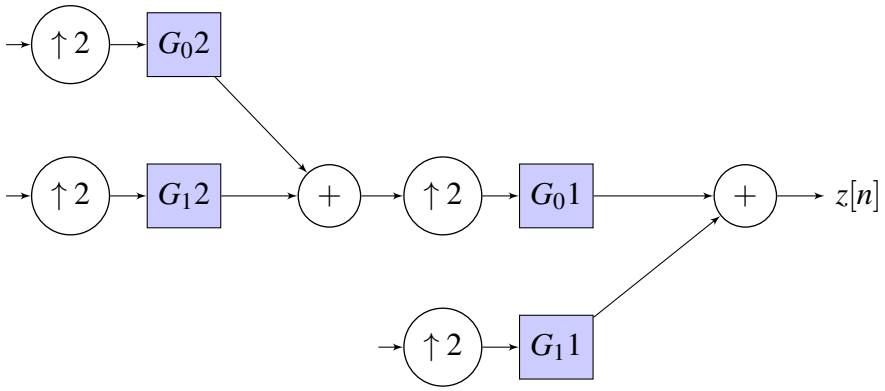


Figure 2.1: original fig.,DWT(fig,5)

The DWT with 5 steps seems pretty sparse and can be used for inpainting problem. But there is a need for W^T as well. As the transformation matrix is orthogonal it is possible to define one.

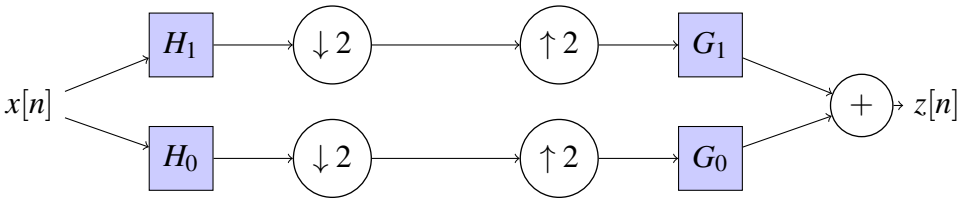
Haar Composition

DWT is a linear operation which means it can be inverted. Inverse discrete wavelet transform is the filter operation which synthesizes the signal from orthonormal basis [10].



$$G0 = \frac{x[n]+x[n+1]}{\sqrt{2}} \quad G1 = \frac{x[n]-x[n+1]}{\sqrt{2}} \tag{2.9}$$

The synthesis filter is highly dependent on the analysis filter. Considering a simple system with 1 decomposition and 1 synthesis steps;



Consider each block as an LTI system in discrete time. Then they can be represented with their 'z' transforms in frequency domain. Down-sampling with 2 (1) and up-sampling with 2 (2) in 'z' domain are [5];

(1) $X'(z) = \frac{1}{2}\left[X(z^{\frac{1}{2}}) + X((-z)^{\frac{1}{2}})\right]$
(2) $X'(z) = X(z^2)$

So that any signal going through up-sampling after down-sampling has the form;

$$X'(z) = \frac{1}{2}\left[X(z) + X(-z)\right]$$

The input signal is X(z)H0(z) and is used as an input of the system G0(z), it's the same for the second row as well. The total equation of the system is;

$$X(z) = \frac{1}{2} \times \left[G_0(z)[H_0(z)X(z) + H_0(-z)X(-z)] + G_1(z)[H_1(z)X(z) + H_1(-z)X(-z)]\right]$$

In matrix notation;

$$X(z)=\frac{1}{2}\begin{bmatrix} G_0(z) & G_1(z) \end{bmatrix}\begin{bmatrix} H_0(z) & H_0(-z) \\ H_1(z) & H_1(-z) \end{bmatrix}\begin{bmatrix} X(z) \\ X(-z) \end{bmatrix}$$

$$\begin{bmatrix} G_0(z) & G_1(z) \end{bmatrix}\begin{bmatrix} H_0(z) & H_0(-z) \\ H_1(z) & H_1(-z) \end{bmatrix}=\begin{bmatrix} 2 & 0 \end{bmatrix}$$

$$\begin{bmatrix} H_0(z) & H_0(-z) \\ H_1(z) & H_1(-z) \end{bmatrix}=H_m$$

$$\begin{bmatrix} G_0(z) \\ G_1(z) \end{bmatrix}=\frac{2}{\left(det(H_m(z)\right)}\times\begin{bmatrix} H_1(-z) \\ -H_0(-z) \end{bmatrix} \quad *$$

Using (*);

$det(H_m(x))=2z^{-1}\quad H_0(z)=\frac{1+z^{-1}}{\sqrt{2}}\quad H_1(z)=\frac{1-z^{-1}}{\sqrt{2}}$

$$\begin{bmatrix} G_0(z) \\ G_1(z) \end{bmatrix}=\begin{bmatrix} \frac{1+z}{\sqrt{2}} \\ \frac{1-z}{\sqrt{2}} \end{bmatrix}$$

Inverse 'z' transforms of the given filters are the exact values defined at the beginning.

Matlab definitions of the Haar Synthesis and 'Inverse Discrete Wavelet Transform' are;

```
function [ y ] = HaarSynthesis( c,d )
%Input variables
%c : Lowpass coefficents
%d : Highpass coefficents
%Output variables
%y : Output signal

    y = zeros(1,2*length(c));
    y(1:2:end) = (c + d)/sqrt(2);
    y(2:2:end) = (c - d)/sqrt(2);
```

Just like in analysis operation, synthesis is just a linear operation between coefficients and the signal values (↑2 in synthesis is included.).

```
function [ II ] = IDWT( I , step )
%I  := Input signal (double)
%II := Output signal (uint8)

%c  := Lowpass coefficents
```

```
%d  := Highpass coefficents

[m,n]=size(I);

for( k = 0 : step-1 )

    x = step-1-k;

    c=I(1:m/(2^(x+1)),1:n/(2^x));
    d=I(m/(2^(x+1))+1:m/(2^x),1:n/(2^x));

    %Columnwise synthesis
    for(i=1:n/(2^x))

        I(1:m/(2^x),i)=transpose(HaarSynthesis(..
        ..transpose(c(:,i)),transpose(d(:,i))));
    end

    c=I(1:m/(2^x),1:n/(2^(x+1)));
    d=I(1:m/(2^x),n/(2^(x+1))+1:n/(2^x));

    %Rowwise synthesis
    for(i=1:m/(2^x))

        I(i,1:n/(2^x))=HaarSynthesis(c(i,:),d(i,:));
    end

end

%Conversion to 8bit
II=(I);

end
```

To verify that the IDWT is perfectly the DWT^{-1} we can simply process any data and check the RMS. First we can again use the random sequence of X used before and apply the wavelet transforms over '5' steps as;

$$X \sim \mathcal{N}(10,10^2) \quad Y_{32X32} = DWT((X_{32X32}),5) \quad Z_{32X32} = IDWT(Y,5)$$

```
X=10*randn(32,32)+10;
Y=DWT(X,5);
Z=IDWT(Y,5);
RMS=sum(sum((X-Z).^2));

RMS =
  2.9770e-010
```

Which is suitable.

Chapter 3

Inpainting in 1D

This section is about solution of inpainting problem in 1D. Douglas Rachford Algorithm is applied, using discrete cosine transform as the orthogonal transformation.

3.1 Discrete Cosine Transform

For sparse signals in 1D the discrete cosine transformation is expected to be sparse. And there is a linear relation between coefficients and the data. We take the IDCT of a sparse vector and then set an interval of the produced signal to zero, in order to construct the 'observation signal'. The discrete cosine transform matrix is not defined by hand but a function of Matlab is used.The matrix W can be created simply by using the dct() function in MATLAB. The inverse discrete cosine transform is a linear operation. We denote it as W. we know that multiplying with identity gives the matrix itself. By W can be computed, also x can be computed by given coefficients.

```
%SIGNAL CONSTRUCTION
%DCT matrix
W=(idct(eye(128)))';

coeff=zeros(128,1);

for(n=1:5)
    coeff(randi(128,1),1)=abs(5*randn(1));
end

x=W'*coeff;
xd=x;
xd(45:65,1)=0;
```

It's clear that with idct(), W^T is computed then it's transpose is taken. The damaging process is applied for the values for indexes i $\in$ [45:65].

Results;

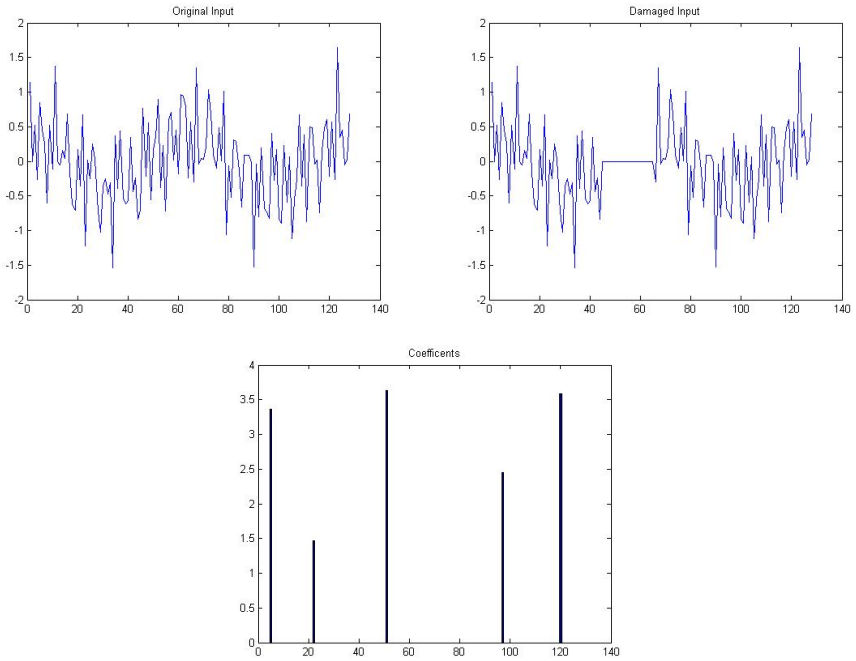


Figure 3.1: Signal construction and damaging results

3.2 Douglas Rachford Algorithm and Application

Now one should apply the algorithm on the damaged image to make an approximation. The algorithm solution can be simplified as;

- z_a=z_b $\leftarrow$ damaged signal
- Repeat;
  - $z_a(damaged part)$=$z_b(corresponding)$
  - m_c = Threshold (W($2z_a - z_b$))
  - $z_b^* = W^T m_c - (z_b - z_a)$
  - $z_b = z_b^*$
- Finish $z_{fin} = za$ with $z_a(damaged part)$=$z_b^*(corresponding)$

$z_a(damaged) = z_b$ corresponds to the projection of the given signal. As given in the problem solution the minimization problem gets the exact values if the point is an element of the original signal, otherwise it takes the exact value as well in order to minimize the l1 norm.

The threshold part corresponds to single valued thresholding since the required function is in the form of;

$$x^* = \begin{cases} y+\lambda & \text{if } y \in (-\infty, -\lambda) \\ 0 & \text{if } y \in [-\lambda, \lambda] \\ y-\lambda & \text{if } y \in (\lambda, \infty) \end{cases}$$

The matlab algorithm for soft thresholding is;

```
function [xmin] = MinCF ( y, lambda)

% f(x) = (((y - x)^2)/2) + lambda * abs(x)
% xmin = arg ( min F(y, lambda))

[m,n] = size(y);
xmin = zeros(m,n);

for ( ii=1 : m)
    for ( i=1 : n)

        if ( lambda < y(ii,i) )
            xmin(ii,i) = y(ii,i) - lambda;

        elseif( y(ii,i) < -lambda )
            xmin(ii,i) = y(ii,i) + lambda;

        else xmin(ii,i) = 0;

        end
    end
end
```

The application of the algorithm is as follows;

```
%APPROXIMATION
za=xd;          %xd:=Damaged Signal
zb=xd;

for n=1:itnum   %itnum:=number of iterations

    za(45:65,1)=zb(45:65,1);
    zb=W'*MinCF(W*(2*za-zb),lambda)+(zb-za);

end

za(45:65,1)=zb(45:65,1);
```

Results after application of the algorithm on damaged signal;

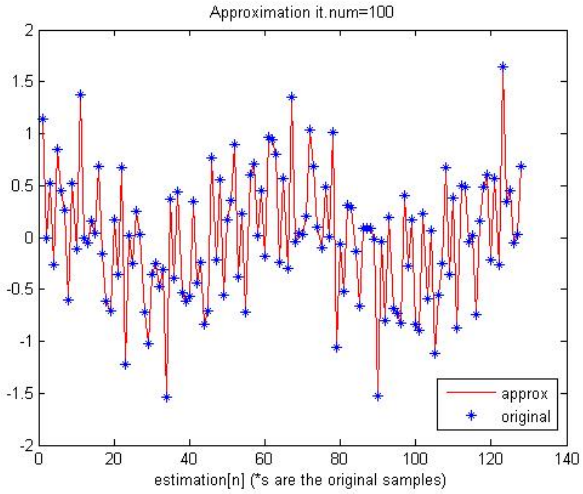


Figure 3.2: '1D' inpainting

The algorithm depends on the condition that, the damaged interval is known. E.g. in the given signal x[45:65,1] is damaged and the algorithm is built on restoring this interval. Also the iteration number is chosen as '100' which is extremely high. Depending on empirical results '20' is enough for perfect fitting. By empirical results one should understand RMS values on different iteration numbers (While λ is constant.).

Chapter 4

Inpainting in 2D (Image Inpainting)

Since the algorithm yields almost perfect results in 1D and proven itself, the solution can be extended for 2D.

4.1 The Orthogonal Transformation Matrix

It's discussed in the previous sections that what to use in order to satisfy the conditions for inpainting problem. Since Haar Wavelets are suitable for the problem. Hence one can use DWT constructed on Haar Analysis basis and IDWT which has Haar Synthesis basis as W and W^T.

Just to draw attention again, Haar decomposition is in the form;

$c[n] = \frac{x[2n]+x[2n-1]}{\sqrt{2}}$ and $d[n] = \frac{x[2n]-x[2n-1]}{\sqrt{2}}$ then $HD(x[n]) = [c[n], d[n]]$

Since the linear transformation is formed. There's also a requirement for it's inverse. Which is also defined as Haar Reconstruction. Where;

$c^*[n] = \uparrow (c[n], 2)$ and $d^*[n] = \uparrow (d[n], 2)$ which correspond to up-sampling.

$c^{**} = \frac{c^*[n]+c^*[n+1]}{\sqrt{2}}$ and $d^{**}[n] = \frac{d^*[n]-d^*[n+1]}{\sqrt{2}}$ Then;

$HR(c[n], d[n]) = c^{**}[n] + d^{**}[n]$

4.2 Douglas Rachford Algorithm

The minimization problem $\arg\min_{z\in P}\|Wz\|_1 = \arg\min_{z}\|Wz\|_1 + i_c(z)$ was solved using Douglas Rachford Algorithm in the previous sections. The Algorithm can be modified for the 2D data as;

Repeat: $z \leftarrow J_f^{\lambda}\left(2J_g^{\lambda}(z) - z\right) + \left(z - J_g^{\lambda}(z)\right)$

Until: converge $z \leftarrow J_g^{\lambda}(z)$

Where $J_h^{\lambda}(z) = \arg\min_{u}\frac{1}{2\lambda}\|z-u\|_2^2 + h(u)$

J_g^{λ} yields the projection of the damaged points directly into approximation. J_f^{λ} yields the SOFT thresholding wrt. 'λ'.

Simplified algorithm is;

- z_a=z_b ← damaged image
- Repeat;
  - $z_a(damaged\,part)$=$z_b(corresponding)$
  - m_c = Threshold (DWT($2z_a - z_b$))
  - $z_b^* = IDWT(m_c) - (z_b - z_a)$
  - $z_b = z_b^*$
- Finish $z_{fin} = za$ with $z_a(damaged\,part)$=$z_b^*(corresponding)$

It's clear that there's no difference in modelling the problem between '1D' and '2D'. The MATLAB code for the Algorithm is;

```
%APPROXIMATION
za=id;
zb=id;
for iterations=1:itnum
    for(count=1:(length(m)))
        za(m(count),n(count))=zb(m(count),n(count));
    end
    zb=IDWT(MinCF(DWT((2*za-zb),step),lambda),step)+(zb-za);
end
 for(count=1:(length(m)))
        za(m(count),n(count))=zb(m(count),n(count));
 end
```

Where $[m,n]$:= indices of the damaged parts.

4.3 Application

As the requirements are fulfilled the only thing is to observe the results. Since the results are highly dependent on factors, 'λ', 'iteration numbers' and the 'damage' itself all factors are taken in consideration. All these factors are going to be examined;

4.3.1 Number of Iterations

Increase in iteration numbers directly increases the approximation quality. However, the trade-off should be handled. Although the more iterations increase the quality it also increases the process time length. As mentioned in '1D' section it can be found by calculating the RMS values for different 'number of iteration's. Where;

$$RMS = \sum_m \sum_n (z_a(m,n) - i(m,n))^2 \text{ for } z_a := approximation\ i := image.$$

As the RMS values are calculated while other variables are constant, it's seen that the iteration quality wrt. RMS converges.For the damaged image;

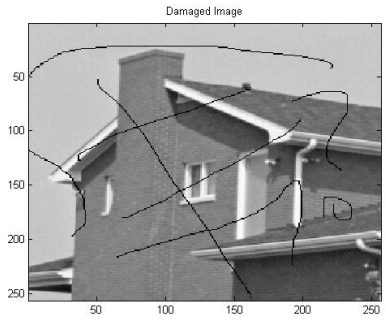


Figure 4.1: damaged image

Reconstruction algorithm is run for different values of iteration numbers and they are calculated as;

Iterations;
$It.Num. = 1 \rightarrow RMS = 9.5325e \times 10^6$
$It.Num. = 20 \rightarrow RMS = 3.3320 \times 10^5$
$It.Num. = 30 \rightarrow RMS = 3.3225 \times 10^5$
$It.Num. = 80 \rightarrow RMS = 3.3053 \times 10^5$
$It.Num. = 150 \rightarrow RMS = 3.2916 \times 10^5$

It's clear that iterating '100' times is enough.

4.3.2 Characteristics of the Damage

The inpainting problem is approximating the original image from it's damaged form. If there's a critical loss of information on the image the algorithm can not expect the original form.

Assume a set of damaged pixels with radius 'r' in the image. DWT contains the information about differences and means of the image and because of the continuity of the damage it is harder to expect the characteristics of the original form. Hence, it's harder to reconstruct the pixel at the center as 'r' increases.

To visualize this, the damage can be modified. But there are no mathematical values calculated. This is possible to see the quality of the approximation;

λ:=50 It.Num.:=100

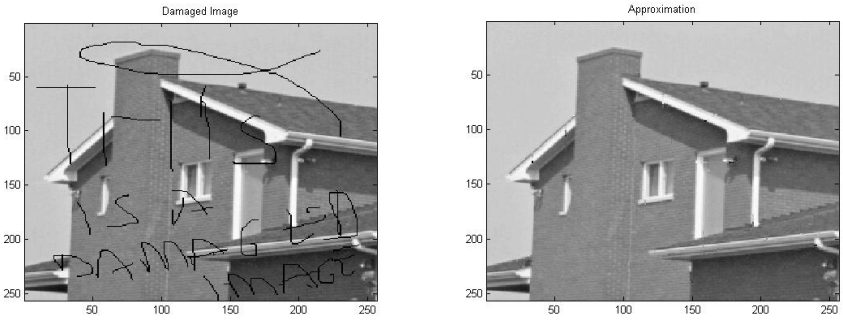


Figure 4.2: damaged image and nearly perfect reconstruction

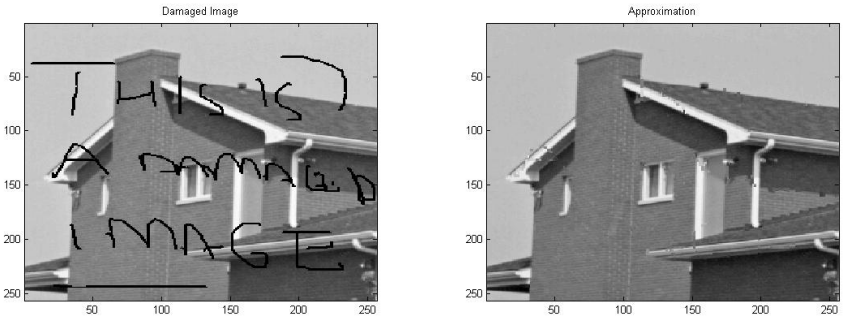


Figure 4.3: damaged image with thicker continuous sets and it's approximation

This is clear that the characteristics of the damage also effect the reconstruction performance. Critically damaged area is filled with the gray tone of the neighbouring pixels after '200+' iterations which is the most brute approximation.

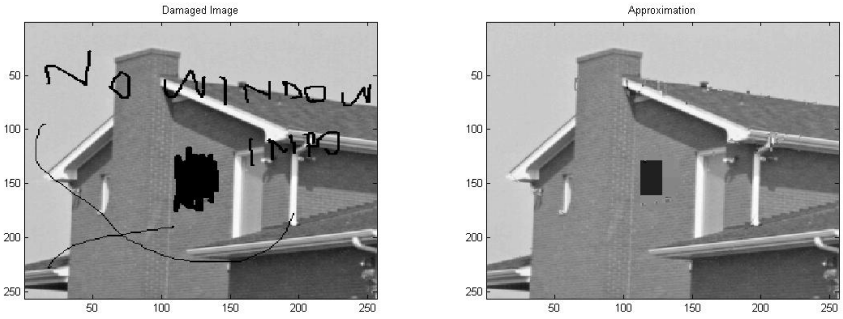


Figure 4.4: critically damaged image with a huge number of neighbouring pixels

4.3.3 SOFT Thresholding

Inpainting problem highly rely on the SOFT thresholding [7]. The thresholding is computed during the Douglas Rachford Algorithm examination, l1 minimization problem yields single valued thresholding.

Thresholding is applied on the DWT of the image to converge each part to the neighbouring ones. Since the operation is highly dependent on λ so does the inpainting. Depending on the different values of λ transformed pixels are pressured around zero.

There is an optimum λ value that minimizes the RMS. It's clear that with λ=0 there is no thresholding, then image stays as it is. On the other hand, if λ is increased above requirement, the original image will be thresholded and become undesirably smoother. The effects of λ on the image can be simply visualised by calculating RMS value on an image;

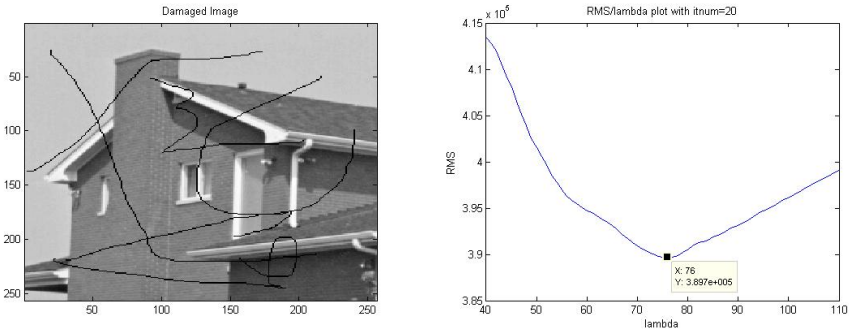


Figure 4.5: damaged image, RMS values for different λ values

4.4 Graphical User Interface

As the problem solution is highly dependent on variables like λ, iteration numbers and DWT step size, a graphical user interface that processes the image in real time can be formed by using MATLAB;

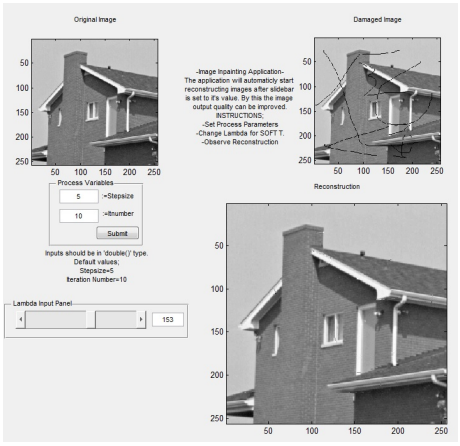


Figure 4.6: Preview of the GUI

The GUI calculates the reconstruction of the image wrt. given input variables λ and iteration numbers. The default values of the variables are, λ=5, it.num.=10. The determination of the lambda is free of any mathematical expression and determined by vision only. Since the wellness of the image is determined visually the approximation is left to the user.

4.4.1 Instructions

The GUI is simple. The only requirement is to define the process variables and submit them. Then moving the λ cursor automatically starts the process. Although it is supposed to be real time process, the run time of the algorithm limits the ability.

Also changing input file requires manual adjustment in the code, by input file one should understand both the original image and the damaged one. Original image is shown in the GUI just to provide the user the ability of comparison.

The simplified instructions are;

- Run the GUI.
- The GUI starts with default variable values.
- Adjust the process variables.
- Observe the difference in inpainting using slide bar for λ.

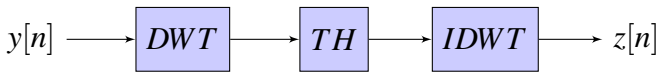
Chapter 5

Denoising

5.1 Denoising Problem

Given an input image x[n] and noise n[n]. De-noising process is the estimation of x[n] from the noisy data y[n]=x[n]+n[n]

De-noising using filter banks [8] can be considered as;



$$z[n] := cleared\ input\ signal$$

For understanding the problem one should observe the probability operations of 'Bayesian Interface' that directly corresponds to SOFT thresholding. DWT and IDWT functions are the Haar wavelets that are used in solving the inpainting problem before.

5.1.1 Maximum Likelihood Estimation

Maximum likelihood [3] is a determination method the true data probability. Given data set $D(x_1, x_2, ..., x_n) \mid x_i \in R^d$, Definition;

θ_{MLE} is a MLE for θ

$$\begin{aligned} \theta_{MLE} &= \arg\max_{\theta \in \Theta} p(D|\theta) \\ p(D|\theta) &= p(x_1, x_2, ..., x_n|\theta) = \prod_{i=1}^{n} p(x_i|\theta) \end{aligned} \tag{5.1}$$

The aim of the MLE is to find the θ value which maximizes the probability density function of the given data. The problem is there's no condition if the mass is concentrated around the maximized probability. The maximum value can be represented by a spike in the pdf hence the concentration of mass can be higher somewhere different.

5.1.2 Bayesian Interface

The aim of Bayesian Interface [3] is to determine the quantity Θ which can be represented as finite collection of random variables wrt. observation $X = (X_1, X_2, ..., X_n)$. The probability distribution of the variable Θ can be found by using the Bayes' Rule;

$$p_{\Theta|X}(\theta|x) = \frac{p_{\Theta}(\theta)p_{X|\Theta}(x|\theta)}{\sum_{\theta'} p_{\Theta}(\theta')p_{X|\Theta}(x|\theta')} \tag{5.2}$$

Where X represents the observations and all are discrete variables. Since the pixel values are discrete this formulation is sufficient.

The estimation can be calculated for multidimensional values if required.

5.1.3 MAP Estimation

Posterior probability is a measurement of a random event after taking some information in consideration, as it has a close meaning of statistical probability. The approach can be simplified as;

1. The desired calculation of probability of parameters θ given the evidence X : p(θ|x)
2. Given the prior information about θ p(θ)
3. The evidence X: p(x|θ)

Then; $p(\theta|x) = \frac{p(\theta)p(x|\theta)}{p(x)}$

The posterior probability can be written in terms of Bayes' Theorem as well.

The MAP estimation aims to maximize the value of the posterior distribution. The

MAP rule is;

$$\theta_{MAP} = \arg\max_{\theta} \; p(\theta|x) \tag{5.3}$$

As the probability density function is a function dependent on θ the maximum value wrt. θ can be calculated by computing;

$$\overset{*}{\theta} \leftarrow \frac{d}{d\theta}(p(\theta|x)) \;\Big|_{\theta=\overset{*}{\theta}} = \frac{d}{d\theta}\Big(\frac{p(\theta)p(x|\theta)}{p(x)}\Big) \;\Big|_{\theta=\overset{*}{\theta}} = 0 \tag{5.4}$$

It's clear that the probability of the observed values $p(x)$ is not dependent on θ values. As $p(x)$ is nothing but a constant for the operation it can be ignored while calculating arg max().

In denoising problem we have the noisy data as the observation and have the prior beliefs that the original signal has the probability distribution of a Laplacian and the noise has the characteristics of a Gaussian;
y(:=noisy signal) $= x$(:=original signal) $+ n$(:=noise)

$p(x|y) = \frac{p(x)p(y|x)}{p(y)}$

Where; $p(x)$ =Laplacian and $p(y|x) = p(n)$ =Gaussian

It's clear that the estimated value of $P_{x|y}[x \mid y]$ is in the form of Gaussian as it they are related with the noise n. Also the aim is to find the difference dependent on x where $P_y[y]$ makes no difference. Also taking logarithm of the function doesn't effect the result. The equation can be simplified as;

$$\begin{aligned} \arg\max_{x}\Big(\frac{(y-x)^2}{2} + \lambda|x|\Big) &= \arg\max_{x} f(x,y,\lambda) \\ x^* = (x|\frac{\partial f(x,y,\lambda)}{\partial x} = 0) &= g(y,\lambda) \end{aligned} \tag{5.5}$$

The result is dependent on sig(x) also the argument x is a function of λ and y. This shows the argument is also dependent on sig(g(y,λ)).
The final result is;

$$x^* = \begin{cases} y+\lambda & \text{if } y \in (-\infty, -\lambda) \\ 0 & \text{if } y \in [-\lambda, \lambda] \\ y-\lambda & \text{if } y \in (\lambda, \infty) \end{cases}$$

Thresholding function is used to smooth the image and remove noise effects. y is one of the values of the input signal, λ is threshold. Which is widely examined in inpainting.

It's unnecessary to give MATLAB codes for thresholding and wavelet transforms again since they are already explained in the inpainting section.

5.2 Estimation of λ (SURE)

As one of the prior problems of inpainting, determining the threshold value in denoising problem is important as well. Although there's no criterion for λ in inpainting problem, it is possible to approximate the best λ value from the noisy signal in denoising under some assumptions [9].

Estimation of λ is highly dependent on the power of the signal. Since the reconstruction of the signal should match the original one, λ can be calculated by using RMS or SNR. λ value is the one which satisfies RMS_{min} or SNR_{max}

$$SNR = 10 \times \log\left(\frac{P_{sig}}{P_{noise}}\right) \qquad RMS = \sum_i (TH(y_i) - y_i)$$

Given methods require the original signal for calculation. The problem is to estimate the original signal from the given input. Again if the noise is considered to have a Gaussian form it's possible to calculate RMS by using SURE.

$$\begin{aligned} E(|(TH(y)-x)|^2) = E(|(TH(y)-(y-n))|^2) \\ = E(|TH(y)-y|^2) + 2 \times E(|(TH(y)-y) \times n|) + E(n^2) \end{aligned} \tag{5.6}$$

$n \sim \mathcal{N}(0, \sigma^2)$ (the noise has normal distribution with zero mean and with variance σ.)

$if\ E(n) = 0\ and\ E(n^2) - E^2(n) = \sigma^2\ then\ E(n^2) = \sigma^2$

There are finitely many numbers which have their exact values to calculate

$$E(|TH(y)-y|^2) = (\sum_i (TH(y_i) - y_i))^2$$

Second term is can be calculated by using Gaussian pdf and integration by parts.
The final SURE statement;

$$\overset{*}{SURE} = \sum_i |TH(y_i) - y_i|^2 + 2\sigma^2 TH'(y_i) - \sigma^2 \tag{5.7}$$

Where TH is the thresholding function. It's clear that the variance is assumed to be known since σ^2 can be calculated almost perfectly by statisticians. SURE estimation depends on the thresholding function. The function we use is;

$$TH(y) = \begin{cases} y+\lambda & \text{if } y \in (-\infty, -\lambda) \\ 0 & \text{if } y \in [-\lambda, \lambda] \\ y-\lambda & \text{if } y \in (\lambda, \infty) \end{cases}$$

Then the SURE estimation can be modified by using this function as;

$$SURE_{RMS} = \begin{cases} \sum \left[|y_i|^2 - \sigma^2 \right] & \text{if } |y_i| \leq \lambda \\ \\ \sum \left[\lambda^2 + \sigma^2 \right] & \text{if } |y_i| > \lambda \end{cases}$$

The *MATLAB* function exactly follows the original one. $RMSandSURE_{RMS}$ functions;

```
SURE=sum(sum(R(abs(R)<=lambda).^2))+sum(sum(abs(R)>lambda))*...
    (lambda^2+2*stdev^2)-stdev^2*numel(R);
RMS(n)=sum(sum((TR-I).^2));
```

Example results for $SNR_{init} = 10$;

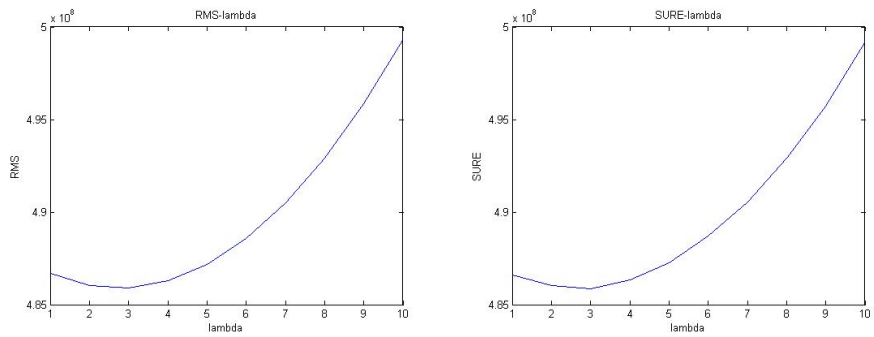


Figure 5.1: RMS-SURE values

$DIFF(n) = (RMS(n) - SURE(n)/RMS(n))values$;

```
min(DIFF);
    ans = 2.1184e-004
max(DIFF)
    ans = 2.6466e-004
```

The factorization values are in $[2\%, 3\%]$ which is acceptable. The difference between between *RMS* and $SURE_{RMS}$ increases as SNR_{init} decreases. This is because *SURE* estimation is mainly related with received signal. Lesser information about the main signal leads to more difficulties in approximation. In contrast, RMS can only calculated by using the original form of the image (The results are checked for $SNR_{init} \in (Z^{+}$ and $[5, 100])$).

5.3 Combination

As there's a solution for the denoising problem and there exists a well approximation for the single thresholding value, the problem can be solved. The image undergoes DWT in first place, this is because it decreases the effect of the noise in each pixel, and as the transformation is orthogonal it does not change the power density of the matrix and does not effect the SURE estimation.

MATLAB code and resultsfor DWT step-size 5;

```
 %I:=input signal
 %r:=noisy signal
 %step:=5
 R=DWT(r,step);
 org=(DWT(I,step));

for(n=1:length(lambda))

    Z=MinCF(R,lambda(n));

    RMS(n)=sum(sum(abs(Z-org).^2));
    SURE(n)=sum(sum(R(abs(R)<=lambda(n)).^2))...
         +sum(sum(abs(R)>lambda(n)))*(lambda(n)^2+2*stdev^2)-...
         stdev^2*numel(R);
    DIFF(n)=abs((RMS(n)-SURE(n))/RMS(n));

    if(n==1) lambdax=lambda(1);
    else if(RMS(n)<RMS(n-1)) lambdax=lambda(n);
        end
    end

    if(n==1) lambday=lambda(1); SUREmin=SURE(1); Zmin=Z;
    else if(SURE(n)<SUREmin) lambday=lambda(n);...
         SUREmin=SURE(n); Zmin=Z;
        end
    end

```

```
28 end
29 II=IDWT(Zmin,step);
```

Sample results;

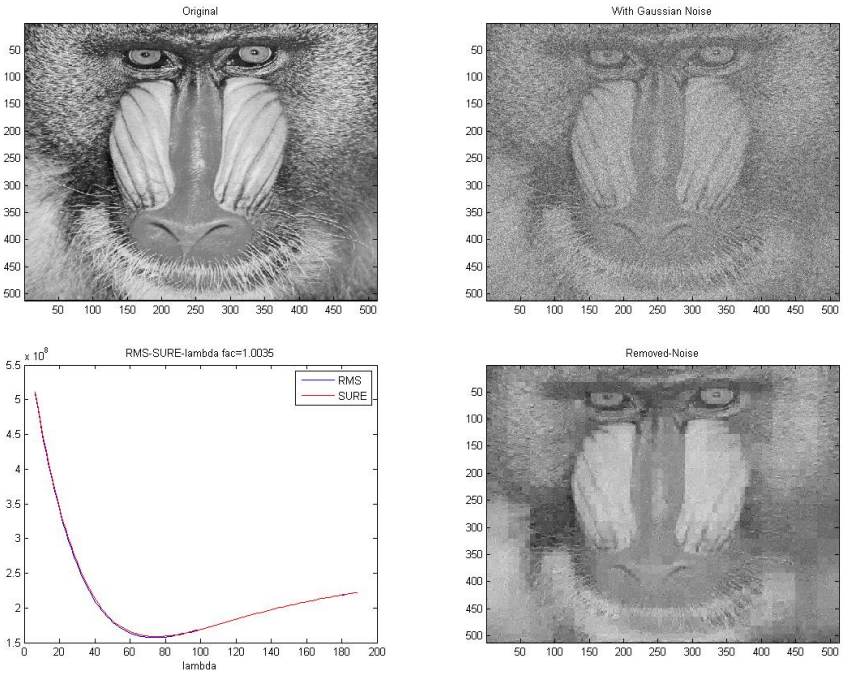


Figure 5.2: De-Noising Process $SNR_{init} = 9$

The lambda difference is;

```
1  lambdax-lambday
2      ans = -3.1473
```

Results

In this project, 'Image Inpainting problem is constructed and solved. As can be seen a damaged image can be restored into its original form approximately with sufficient thresholding values. Also the algorithm should be repeated efficient times which has a trade off with time and wellness.

This project also includes important information about wavelet operations and their effects. All the operations are the results of the solutions of problems. Only Douglas Rachford algorithm does not have an interpretation in the project. However, the inpainting problem has been changed into suitable form.

During the preparation, Denoising and Stein's Unbiased Risk Estimation is studied. The solution is modelled and verified by using MATLAB. Almost entire MATLAB implementation algorithms are products of this project.

Finally, the signal processing has a wide area of usage. Inpainting problem is one of the prior problems in the discipline. The project clarifies the mathematical background between the inpainting operation and can be used for expanding the understanding of such problems. In addition by using this solution better algorithms for inpainting can be developed.

Bibliography

[1] Afonso, M., Bioucas-Dias, J., and Figueiredo, M. (2010). Fast image recovery using variable splitting and constrained optimization. *Image Processing, IEEE Transactions on*, 19(9):2345–2356.

[2] Bayram, I. and Kamasak, M. (2013). A simple prior for audio signals. *Image Processing, IEEE Transactions on*, 21(6):95–110.

[3] Bertsekas, D. and Tsitsiklis, J. (2002). *Introduction To Probability*. Athena Scientific books. Athena Scientific.

[4] Mathworks (n.d.). Matlab documentation. *http://www.mathworks.de/de/help/-matlab/*. Accessed: May 2014.

[5] Oppenheim, A. V., Schafer, R. W., and Buck, J. R. (1999). *Discrete-time Signal Processing (2Nd Ed.)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

[6] Oppenheim, A. V., Willsky, A. S., and Nawab, S. H. (1996). *Signals &Amp; Systems (2Nd Ed.)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

[7] Selesnick, I. (2013). A derivation of the soft-thresholding function. *http://eeweb.poly.edu/iselesni/lecturenotes/*. Accessed: May 2014.

[8] Vaidyanathan, P. P. (1990). Multirate digital filters, filter banks, polyphase networks, and applications: A tutorial. *Proceedings of the IEEE*, 78(1):56–93.

[9] Van De Ville, D. and Kocher, M. (2009). Sure-based non-local means. *Signal Processing Letters, IEEE*, 16(11):973–976.

[10] Vetterli, M. and Kovačević, J. (1995). *Wavelets and subband coding*. Prentice Hall PTR.

Appendix

Haar Analysis

```
function [ c,d ] = HaarAnalysis( x )
% Input variables
% x : input signal
% Output Variables
% c : Lowpass coefficients
% d : Highpass coefficients

if mod(length(x),2),
    x = [x 0];
end

c = (x(1:2:end) + x(2:2:end))/sqrt(2);
d = (x(1:2:end) - x(2:2:end))/sqrt(2);
end
```

Haar Synthesis

```
function [ y ] = HaarSynthesis( c,d )
%Input variables
%c : Lowpass coefficents
%d : Highpass coefficents
%Output variables
%y : Output signal

    y = zeros(1,2*length(c));
    y(1:2:end) = (c + d)/sqrt(2);
    y(2:2:end) = (c - d)/sqrt(2);
```

DWT

```
function [ II , step ] = DWT( I , step )
%I     := Input signal (uint8)
%steup := number of steps
%II    := Output signal (double)

%c  := Lowpass coefficents
%d  := Highpass coefficents
```

```
8
9  %Conversion to double required
10 %uint8 varies between 0-255 which can't hold negative numbers
11 %negative numbers are recieved while calculating d(highpass coef.)
12 I=double(I);
13
14 [m,n]=size(I);
15
16 %number of steps
17 for(x = 0 : step-1)
18
19     %Rowwise analysis
20     for(i=1:m/(2^x))
21             [c,d]=HaarAnalysis(I(i,1:m/(2^x)));
22             I(i,1:m/(2^x))=[c d];
23     end
24     %Columnwise analysis
25     for(i=1:n/(2^x))
26         [c,d]=HaarAnalysis(transpose(I(1:n/(2^x),i)));
27         I(1:n/(2^x),i)=[transpose(c);transpose(d)];
28     end
29 end
30     II=I;
31 end
```

IDWT

```
1  function [ II ] = IDWT( I , step )
2  %I  := Input signal (double)
3  %II := Output signal (uint8)
4
5  %c  := Lowpass coefficents
6  %d  := Highpass coefficents
7
8  [m,n]=size(I);
9
10 for( k = 0 : step-1 )
11
12     x = step-1-k;
13
14     c=I(1:m/(2^(x+1)),1:n/(2^x));
15     d=I(m/(2^(x+1))+1:m/(2^x),1:n/(2^x));
16
17     %Columnwise synthesis
18     for(i=1:n/(2^x))
19         I(1:m/(2^x),i)=transpose(HaarSynthesis..
20         ..(transpose(c(:,i)),transpose(d(:,i))));
21     end
22     c=I(1:m/(2^x),1:n/(2^(x+1)));
23     d=I(1:m/(2^x),n/(2^(x+1))+1:n/(2^x));
24     %Rowwise synthesis
25     for(i=1:m/(2^x))
26         I(i,1:n/(2^x))=HaarSynthesis(c(i,:),d(i,:));
27     end
28 end
29
```

```
II=I;
end
```

MINCF

```
function [xmin] = MinCF ( y, lambda)

% f(x) = (((y - x)^2)/2) + lambda * abs(x)
% xmin = arg ( min F(y, lambda))

[m,n] = size(y);
xmin = zeros(m,n);

for ( ii=1 : m)
    for ( i=1 : n)

        if ( lambda < y(ii,i) )
            xmin(ii,i) = y(ii,i) - lambda;

        elseif( y(ii,i) < -lambda )
            xmin(ii,i) = y(ii,i) + lambda;

        else xmin(ii,i) = 0;

        end
    end
end
```

Demo Noise

```
% Noise addition / removal demo
close all;
clear all;

%INPUT DATA
I=imread('baboon.bmp');
I=double(I);

%INPUT VALUES
SNRinit=12;                         %initial value of SNR
step=5;                             %step-size of HAAR decomposition
%-------------------------------------------------------


lambdax=0;
[m,n]=size(I);
originalP = sum(sum(I.^2));

stdev=(sqrt(originalP/(10^(SNRinit/10)*m*n)));
% r=imnoise(I,'gaussia  n',0,0.01);

noise=stdev*randn(size(I));
r=I+noise;

```

```
Pnoise=sum(sum((noise).^2));

SNRstartdiff=SNRinit-(10*(log(originalP/Pnoise)/log(10)));

figure;
imagesc(I);
colormap(gray);
title('Original');

figure;
imagesc(r);
colormap(gray);
title('With Gaussian Noise');

 R=DWT(r,step);

 figure;
 imagesc(R);
 colormap(gray);
 title('Transformation of Noisy Image');

 org=(DWT(I,step));



fa = 50;
lambda=(1:0.25:30)*(fa * 10^(-SNRinit/10));    %lambda value interval

DIFF=zeros(1,length(lambda));
SURE=zeros(1,length(lambda));
RMS=zeros(1,length(lambda));


for(n=1:length(lambda))

    Z=MinCF(R,lambda(n));

    RMS(n)=sum(sum(abs(Z-org).^2));
    SURE(n)=sum(sum(R(abs(R)<=lambda(n)).^2))..
    ..+sum(sum(abs(R)>lambda(n)))*(lambda(n)^2+2*stdev^2)-..
    ..stdev^2*numel(R);
    DIFF(n)=abs((RMS(n)-SURE(n))/RMS(n));

    if(n==1) lambdax=lambda(1);
    else if(RMS(n)<RMS(n-1)) lambdax=lambda(n);
        end
    end

    if(n==1) lambday=lambda(1); SUREmin=SURE(1); Zmin=Z;
    else if(SURE(n)<SUREmin) lambday=lambda(n); SUREmin=SURE(n); Zmin=Z;
        end
    end

end

II=IDWT(Zmin,step);
```

```
83
84 fac=min(RMS(n))/min(SURE(n));
85 figure();
86 plot(lambda,RMS,'b');
87 title(strcat('RMS-SURE-lambda',' fac=',num2str(fac)));
88
89 hold;
90 plot(lambda,SURE*fac,'r');
91 legend('RMS','SURE');
92 xlabel('lambda');
93
94
95 figure;
96 imagesc(II);
97 colormap(gray);
98 title('Removed-Noise');
99
100 figure;
101 imagesc(r-double(I));
102 colormap(gray);
103 title('First Difference');
104
105 figure;
106 imagesc(II-double(I));
107 colormap(gray);
108 title('Last Difference');
```

Demo Inpainting

```
1  %DEMO for Image INPAINTING (2D)
2
3  close all;
4  clear all;
5
6  %INPUT image
7  i=double(imread('house.bmp'));
8  %PNG format should be used for paint modification. Otherwise the Matlab
9  %transforms the histogram and changes the outlook.
10 id=double(rgb2gray(imread('houseasd.png')));
11 itnum=20;      %Number of iterations
12 step=5;        %DWT,IDWT stepsize
13 lambda=40:110;
14 RMS=zeros(length(lambda));
15
16 %Damaging the Signal
17 % id=i;
18 % id(60:120,70)=0;
19
20 %Indexes of '0' (damaged) elements. This is required to determine the
21 %region to be transferred.
22 [m,n]=find(id<1);
23
24 figure();
25 imagesc(i);
26 title('Original Image');
27 colormap('gray');
```

```
28
29 figure();
30 imagesc(id);
31 title('Damaged Image');
32 colormap('gray');
33
34 %APPROXIMATION
35
36
37 for lambdacounter=1:length(lambda)
38 za=id;
39 zb=id;
40 for iterations=1:itnum
41
42     for(count=1:(length(m)))
43         za(m(count),n(count))=zb(m(count),n(count));
44     end
45
46     zb=IDWT(MinCF(DWT((2*za..
47     ..-zb),step),lambda(lambdacounter)),step)+(zb-za);
48
49 end
50
51  for(count=1:(length(m)))
52         za(m(count),n(count))=zb(m(count),n(count));
53
54  end
55  RMS(lambdacounter)=sum(sum((za-i).^2));
56 end
57
58 figure();
59 imagesc(za);
60 title('Approximation');
61 colormap('gray');
```

Inpainting Function

```
1  function [ za ] = InpaintingFunct( image,lambda,stepsize,iteratesize )
2  %Inpainting function basicly gets the image as an input,
3  %applies DWT and IDWT functionts wrt. 'stepsize', soft
4  %thresholds the output wrt. 'lambda' and completes the
5  %process after 'iteratesize' number of interations
6
7  [m,n]=find(image<1);
8
9  za=image;
10 zb=image;
11
12 for iterations=1:iteratesize
13
14     for(count=1:(length(m)))
15         za(m(count),n(count))=zb(m(count),n(count));
16     end
17
18     zb=IDWT(MinCF(DWT((2*za-zb),stepsize),lambda),stepsize)+(zb-za);
19
```

```
20 end
21
22  for(count=1:(length(m)))
23          za(m(count),n(count))=zb(m(count),n(count));
24  end
25
26 end
```

GUI

Gui data is not necessary to be shown. Since the only thing that the 'callback' functions do is to declare public variables, computations or activating others. The linkage does not include lethal information about the project.