Some applications using the Discrete Wavelet Transform

Preliminaries

The Hadamard matrix

In[372]:=
```
H = 1/Sqrt[2] {{1, 1}, {1, -1}}; H // MatrixForm
```

Out[372]//MatrixForm=

$$\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}$$

The 1D-Haar transform

1D-HT, one level, variable length

In[373]:=
```
ht[A_,k_]:=Module[{AA,len,head,tail,e1,e2,e3,e4},
AA=A;
len=Length[AA];
If[2 k>len,Throw[k "is too big"]];
head=Take[AA,{1,2 k}];
tail=Take[AA,{2 k+1,len}];
e2=Partition[head,2];
e3=H.Transpose[e2];
e4=Flatten[e3];
Simplify[Flatten[{e4,tail}]]
]
```

In[374]:=
```
{ht[{a,b},1],ht[ht[{a,b},1],1]}
```

Out[374]= $\left\{\left\{\frac{a+b}{\sqrt{2}}, \frac{a-b}{\sqrt{2}}\right\}, \{a, b\}\right\}$

In[375]:=
```
L = {a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p};
```

In[376]:=
```
ht[L, 3]
```

Out[376]= $\left\{\frac{a+b}{\sqrt{2}}, \frac{c+d}{\sqrt{2}}, \frac{e+f}{\sqrt{2}}, \frac{a-b}{\sqrt{2}}, \frac{c-d}{\sqrt{2}}, \frac{e-f}{\sqrt{2}}, g, h, i, j, k, l, m, n, o, p\right\}$

■ 1D-HT, several levels, variable length

In[377]:=
```
ht[A_,k_,n_] :=Module[{tmp,len,head,tail,efflen,j},
len=Length[A];
efflen=2^n k;
If[efflen>len,Throw["list is too short"]];
     head=Take[A,efflen];
     tail=Take[A,{efflen+1,len}];
     tmp=head;
     For[j=1,j≤n,j++,tmp=ht[tmp,efflen 2^(-j)]];
     Simplify[Flatten[{tmp,tail}]]
]
```

In[378]:= `ht[L, 2, 2]`

Out[378]= $\left\{\frac{1}{2}(a+b+c+d), \frac{1}{2}(e+f+g+h), \frac{1}{2}(a+b-c-d), \frac{1}{2}(e+f-g-h), \frac{a-b}{\sqrt{2}}, \frac{c-d}{\sqrt{2}}, \frac{e-f}{\sqrt{2}}, \frac{g-h}{\sqrt{2}}, i, j, k, l, m, n, o, p\right\}$

- 1D-inverseHT, one level, variable length

In[379]:=
```
iht[A_,k_]:=Module[{len,head,tail,e1,e2,e3},
len=Length[A];If[2 k>len,Throw[k "is too big"]];
head=Take[A,2 k];
tail=Take[A,{2 k+1,len}];
e1=Partition[head,k];
e2=H.e1;
e3=Flatten[Transpose[e2]];
Simplify[Flatten[{e3,tail}]]
]
```

- 1D-iHT, several levels, variable length

In[380]:=
```
iht[A_,k_,n_] :=Module[{tmp,head,tail,len,j},
len=Length[A];
If[(2^n) k>len,Throw["list is too short"]];
     head=Take[A,(2^n) k];
     tail=Take[A,{(2^n) k +1,len}];
     tmp=head;
     For[j=1,j≤n,j++,tmp=iht[tmp,2^(j-1) k]];
     Simplify[Flatten[{tmp,tail}]]
]
```

- 1D-HT

In[381]:=
```
ht[A_]:=Module[{},
If[OddQ[Length[A]],Throw["length odd"]];
ht[A,Length[A]/2]
]
```

- 1D-iHT

In[382]:=
```
iht[A_]:=Module[{},
If[OddQ[Length[A]],Throw["length odd"]];
iht[A,Length[A]/2]
]
```

The 2D-Haar transform

■ 2D-matrix-HT, one level

In[383]:=
```
HrowM[A_]:=Module[{AH,k},
AH=A;
Do[ AH[[k]]=ht[AH[[k]]],{k,1,Dimensions[AH][[1]]}];
AH
]
```

In[384]:=
```
HcolM[A_]:=Module[{HA,k},
HA=Transpose[A];
Do[HA[[k]]=ht[HA[[k]]],{k,1,Dimensions[HA][[1]]}];
Transpose[HA]
]
```

In[385]:=
```
HtransM[A_]:=Module[{HAH},
HAH=A;
HcolM[HrowM[HAH]]
]
```

■ 2D-matrix-HT, several levels

In[386]:=
```
HtransM[A_,n_]:=Module[{AA,r,c},
{r,c}=Dimensions[A];
AA=HtransM[A];
If[n==1,Return[AA]];
AA[[1;;r/2,1;;c/2]] = HtransM[AA[[1;;r/2,1;;c/2]],n-1];
AA
]
```

■ 2D-matrix-iHT, one level

In[387]:=
```
iHrowM[A_]:=Module[{AH,k},
AH=A;
Do[ AH[[k]]=iht[AH[[k]]],{k,1,Dimensions[AH][[1]]}];
AH
]
```

In[388]:=
```
iHcolM[A_]:=Module[{HA,k},
HA=Transpose[A];
Do[HA[[k]]=iht[HA[[k]]],{k,1,Dimensions[HA][[1]]}];
Transpose[HA]
]
```

In[389]:=
```
iHtransM[A_]:=Module[{HAH},
HAH=A;
iHcolM[iHrowM[HAH]]
]
```

■ 2D-matrix-iHT, several levels

In[390]:=
```
iHtransM[A_,n_]:=Module[{AA,r,c},
AA=A;
{r,c}=Dimensions[A];
If[n==1,Return[iHtransM[AA]]];
AA[[1;;r/2,1;;c/2]] = iHtransM[AA[[1;;r/2,1;;c/2]],n-1];
iHtransM[AA]
]
```

In[391]:= `Lmat = Partition[L, 4]; Lmat // MatrixForm`

Out[391]//MatrixForm=

$$\begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{pmatrix}$$

In[392]:= `Y = HrowM[Lmat]`

Out[392]=
$$\left\{\left\{\frac{a+b}{\sqrt{2}}, \frac{c+d}{\sqrt{2}}, \frac{a-b}{\sqrt{2}}, \frac{c-d}{\sqrt{2}}\right\}, \left\{\frac{e+f}{\sqrt{2}}, \frac{g+h}{\sqrt{2}}, \frac{e-f}{\sqrt{2}}, \frac{g-h}{\sqrt{2}}\right\}, \left\{\frac{i+j}{\sqrt{2}}, \frac{k+l}{\sqrt{2}}, \frac{i-j}{\sqrt{2}}, \frac{k-l}{\sqrt{2}}\right\}, \left\{\frac{m+n}{\sqrt{2}}, \frac{o+p}{\sqrt{2}}, \frac{m-n}{\sqrt{2}}, \frac{o-p}{\sqrt{2}}\right\}\right\}$$

In[393]:= `Z = HcolM[Y]`

Out[393]=
$$\left\{\left\{\frac{1}{2}(a+b+e+f), \frac{1}{2}(c+d+g+h), \frac{1}{2}(a-b+e-f), \frac{1}{2}(c-d+g-h)\right\}, \left\{\frac{1}{2}(i+j+m+n), \frac{1}{2}(k+l+o+p), \frac{1}{2}(i-j+m-n), \frac{1}{2}(k-l+o-p)\right\}, \left\{\frac{1}{2}(a+b-e-f), \frac{1}{2}(c+d-g-h), \frac{1}{2}(a-b-e+f), \frac{1}{2}(c-d-g+h)\right\}, \left\{\frac{1}{2}(i+j-m-n), \frac{1}{2}(k+l-o-p), \frac{1}{2}(i-j-m+n), \frac{1}{2}(k-l-o+p)\right\}\right\}$$

In[394]:= `Simplify[HtransM[Lmat] - Z]`

Out[394]= {{0, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}}

In[395]:= `Simplify[Lmat - iHtransM[Z]]`

Out[395]= {{0, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}}

- 2D-image-HT, one level, with decomposition

In[396]:=
```
Htrans[A_]:=Module[{Adata,r,c},
{r,c}=ImageDimensions[A];
Adata=ImageData[A];
ImagePartition[Image[HcolM[HrowM[Adata]]],{r/2,c/2}]
]
```

- 2D-image-HT, several levels, without decomposition

In[397]:=
```
Htrans[A_,n_]:=Module[{},
Image[HtransM[ImageData[A],n]]
]
```

- 2D-image-HT, several levels, without decomposition

In[398]:=
```
Htrans[A_,n_]:=Module[{},
Image[HtransM[ImageData[A],n]]
]
```

- 2D-image-iHT, one level, with decomposition

In[399]:=
```
glue4[{{A_,B_},{C_,D_}}]:=Module[{},
adim=ImageDimensions[A];
bdim=ImageDimensions[B];
cdim=ImageDimensions[C];
ddim=ImageDimensions[D];
If[adim≠bdim || bdim≠cdim ||cdim≠ddim,Throw["Dimensionen passen nicht"]];
ImageAssemble[{{A,B},{C,D}}]
]
```

In[400]:=
```
iHtransgl[{{A_,B_},{C_,D_}}]:=Module[{gl},
gl=glue4[{{A,B},{C,D}}];
Image[iHtransM[ImageData[gl]]]
]
```

- 2D-image-iHT, several levels, without decomposition

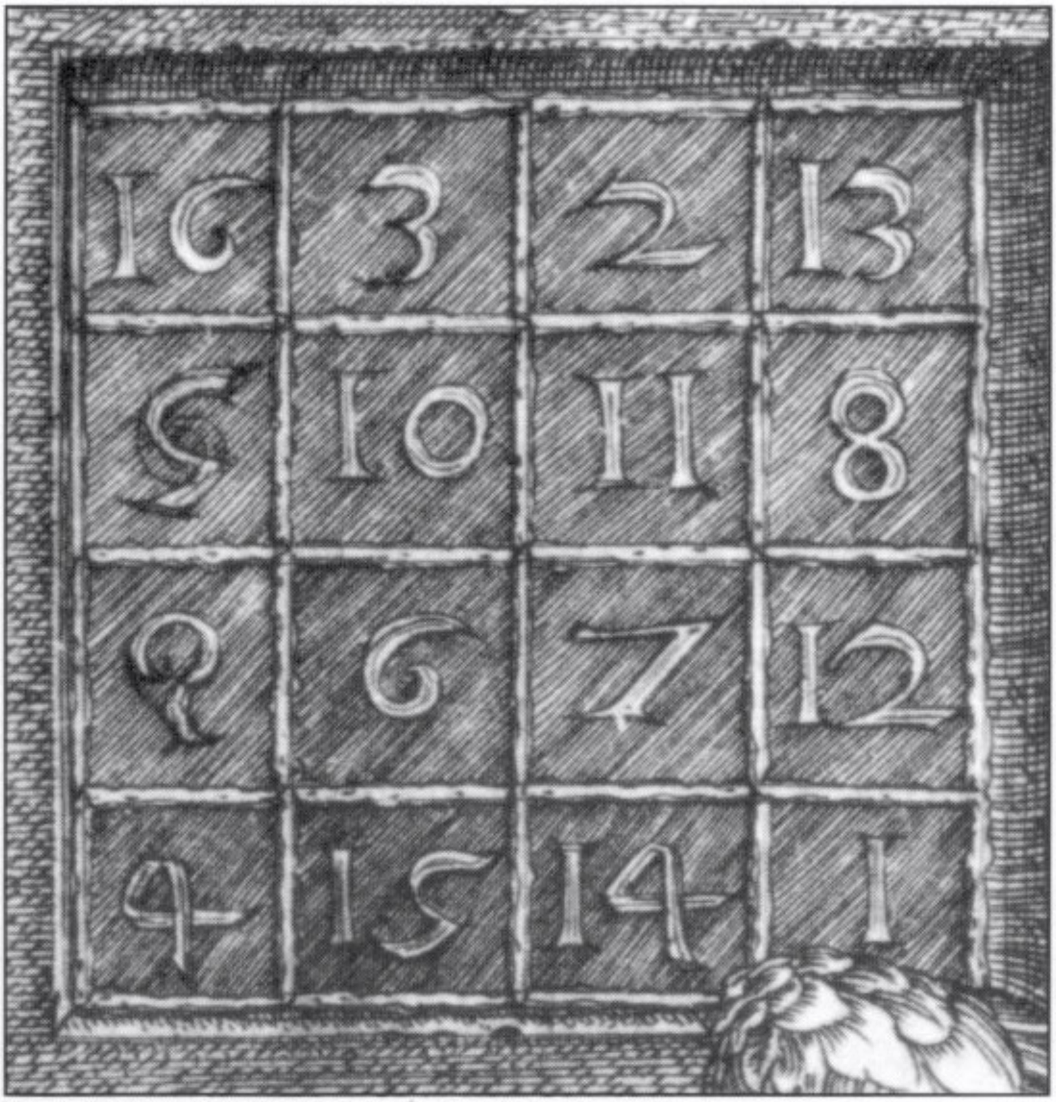
In[401]:=
```
iHtrans[A_,n_]:=Module[{},
Image[iHtransM[ImageData[A],n]]
]
```

A test image

- The image

In[402]:=
```
durer = Import["~/LEHRE/Wavelets-All/WTBV-11/Bilder/magic-square.jpg"]
```

Out[402]=



In[403]:=
```
ImageDimensions[durer]
```

Out[403]=
```
{555, 578}
```

In[404]:=
```
ImageData[durer][[100 ;; 103, 100 ;; 103]] // MatrixForm
```

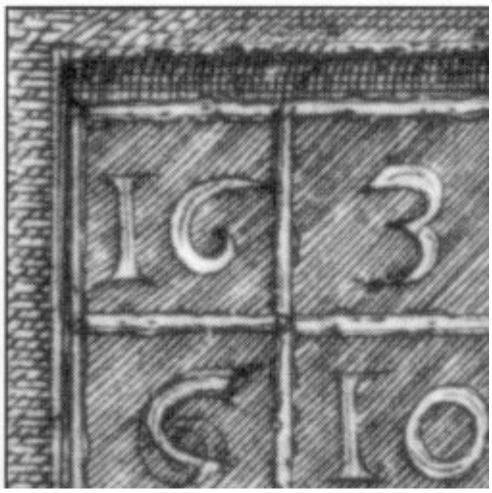
Out[404]//MatrixForm=

(0.439216, 0.439216, 0.447059)	(0.337255, 0.337255, 0.345098)	(0.345098, 0.345098, 0.352941)	(0.431373, 0.431373, 0.439216)
(0.388235, 0.388235, 0.396078)	(0.403922, 0.403922, 0.411765)	(0.505882, 0.505882, 0.513725)	(0.541176, 0.541176, 0.54902)
(0.431373, 0.431373, 0.439216)	(0.584314, 0.584314, 0.592157)	(0.666667, 0.666667, 0.67451)	(0.698039, 0.698039, 0.705882)
(0.635294, 0.635294, 0.643137)	(0.666667, 0.666667, 0.67451)	(0.745098, 0.745098, 0.752941)	(0.705882, 0.705882, 0.713725)

a selection from the test image

In[405]:=
```
durerb = ColorConvert[ImageTake[durer, {1, 256}, {1, 256}], "Grayscale"]
```

Out[405]=



In[406]:=
```
ImageData[durerb][[100 ;; 103, 100 ;; 103]] // MatrixForm
```

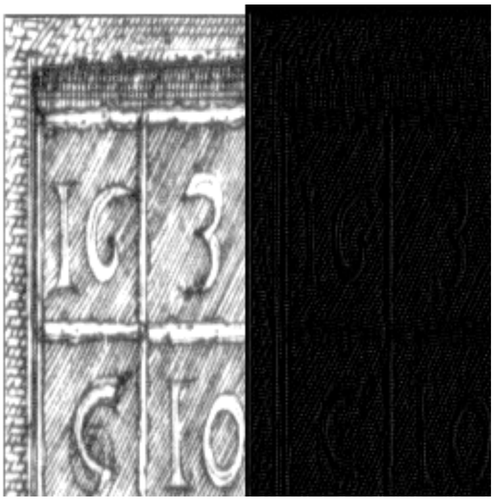
Out[406]//MatrixForm=

0.439216	0.337255	0.345098	0.431373
0.388235	0.403922	0.505882	0.541176
0.431373	0.584314	0.666667	0.698039
0.635294	0.666667	0.745098	0.705882

■ Haar-DWT on the rows

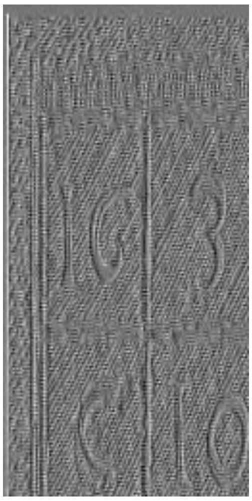
In[407]:=
```
Image[HrowM[ImageData[durerb]]]
```

Out[407]=



In[408]:=
```
ImageAdjust[ImageTake[Image[HrowM[ImageData[durerb]]], {1, 256}, {129, 256}]]
```

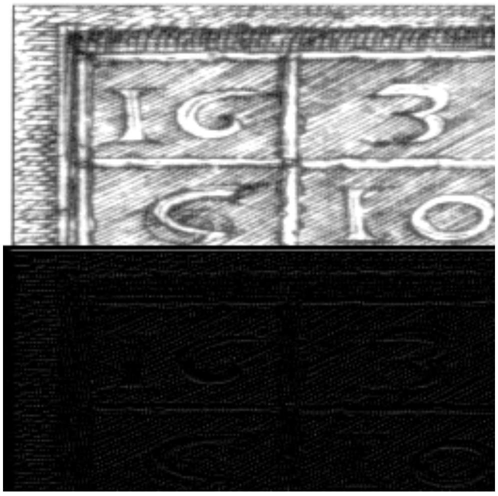
Out[408]=



- Haar-DWT on the columns

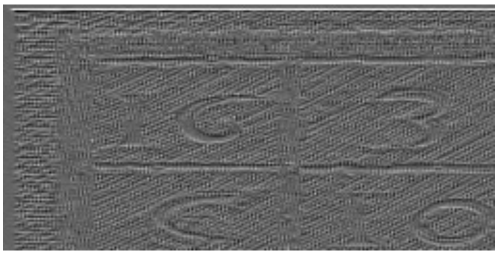
In[409]:=
```
Image[HcolM[ImageData[durerb]]]
```

Out[409]=



In[410]:=
```
ImageAdjust[ImageTake[Image[HcolM[ImageData[durerb]]], {129, 256}, {1, 256}]]
```

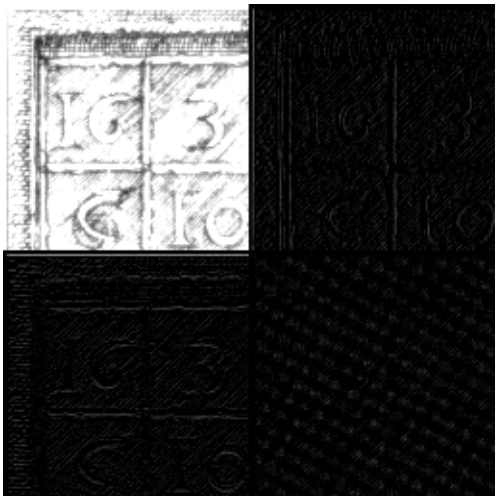
Out[410]=



- one-level Haar transform

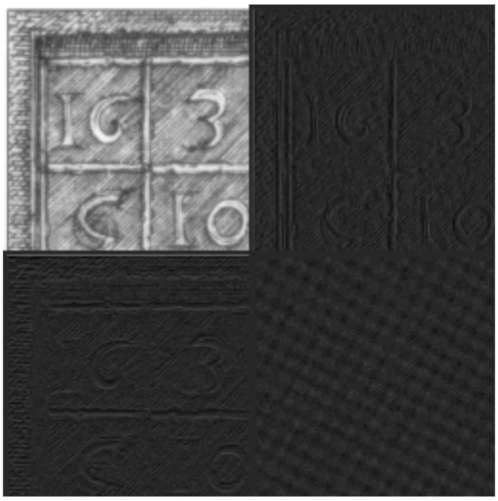
In[411]:=
```
Image[HtransM[ImageData[durerb]]]
```

Out[411]=



In[412]:=
```
ImageAdjust[%]
```

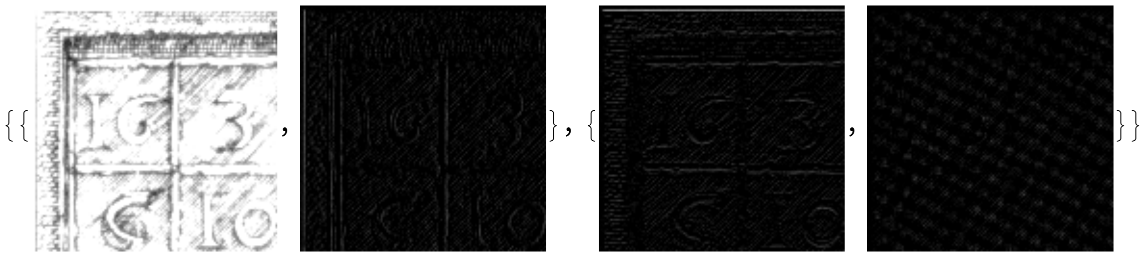
Out[412]=



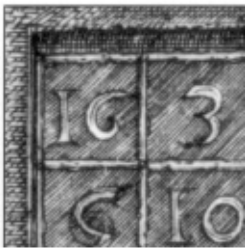
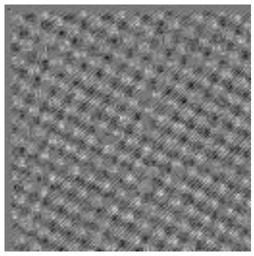
- one-level Haar transform with decomposition

In[413]:=
```
H1durerb = Htrans[durerb]
```

Out[413]=



In[414]:=
```
Map[ImageAdjust, H1durerb, {2}]
```

Out[414]= {{, }, {, }}

- energy distribution for the subimages

In[415]:=
```
Map[Norm[ImageData[#]]^2 &, H1durerb, {2}] // MatrixForm
```

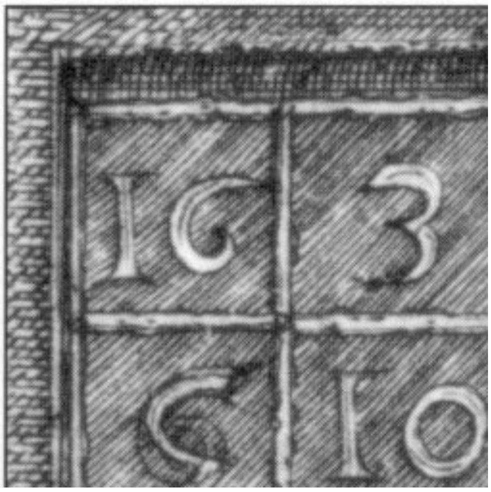
Out[415]//MatrixForm=

$$\begin{pmatrix} 17821.7 & 20.9918 \\ 53.1315 & 9.11351 \end{pmatrix}$$

- inverse Haar transform

In[416]:=
```
iHtransgl[H1durerb]
```

Out[416]=



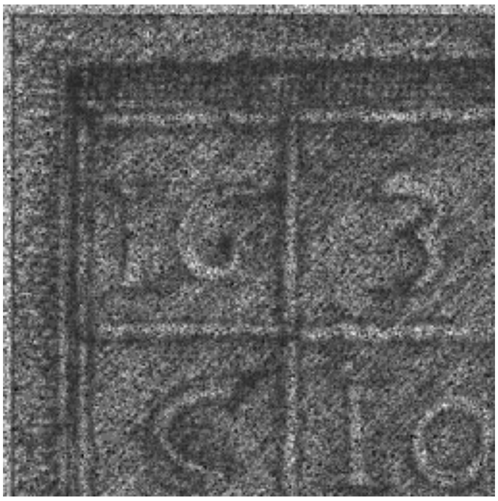
- difference between original and reconstruction

In[417]:=
```
Norm[ImageData[ImageSubtract[durerb, %]]]
```

Out[417]= 4.7768×10^{-14}

In[418]:=
```
ImageAdjust[ImageSubtract[durerb, iHtransgl[H1durerb]]]
```

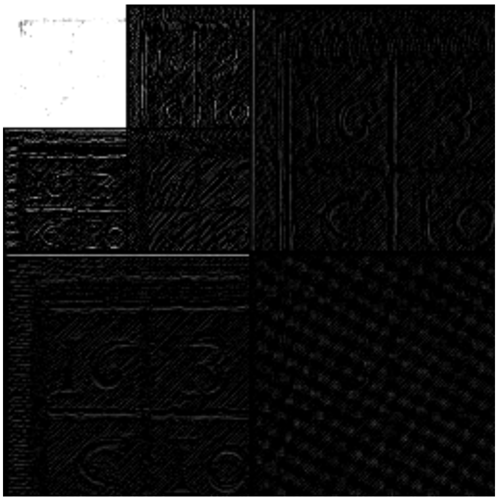
Out[418]=



- two-level Haar transform

In[419]:=
```
H2durerb = Image[Htrans[durerb, 2]]
```

Out[419]=

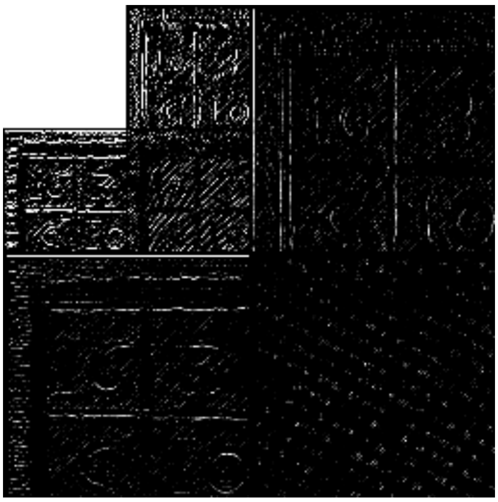


In[420]:=
```
ImageAdjust[H2durerb]
```

Out[420]=

In[421]:=
```
ImageAdjust[H2durerb, {1, 2}]
```

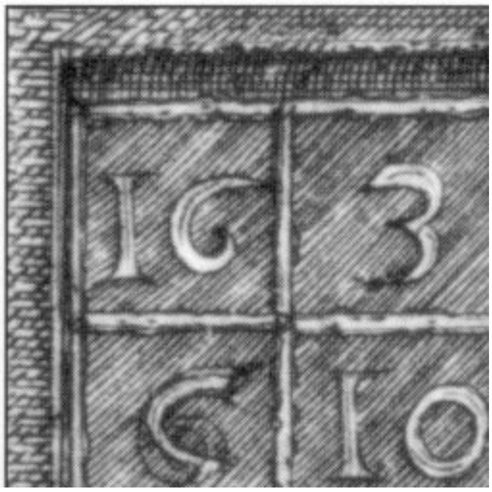
Out[421]=



- inverse of the two-level transform

In[422]:=
```
iHtrans[H2durerb, 2]
```

Out[422]=



In[423]:=
```
Norm[ImageData[ImageSubtract[durerb, %]]]
```

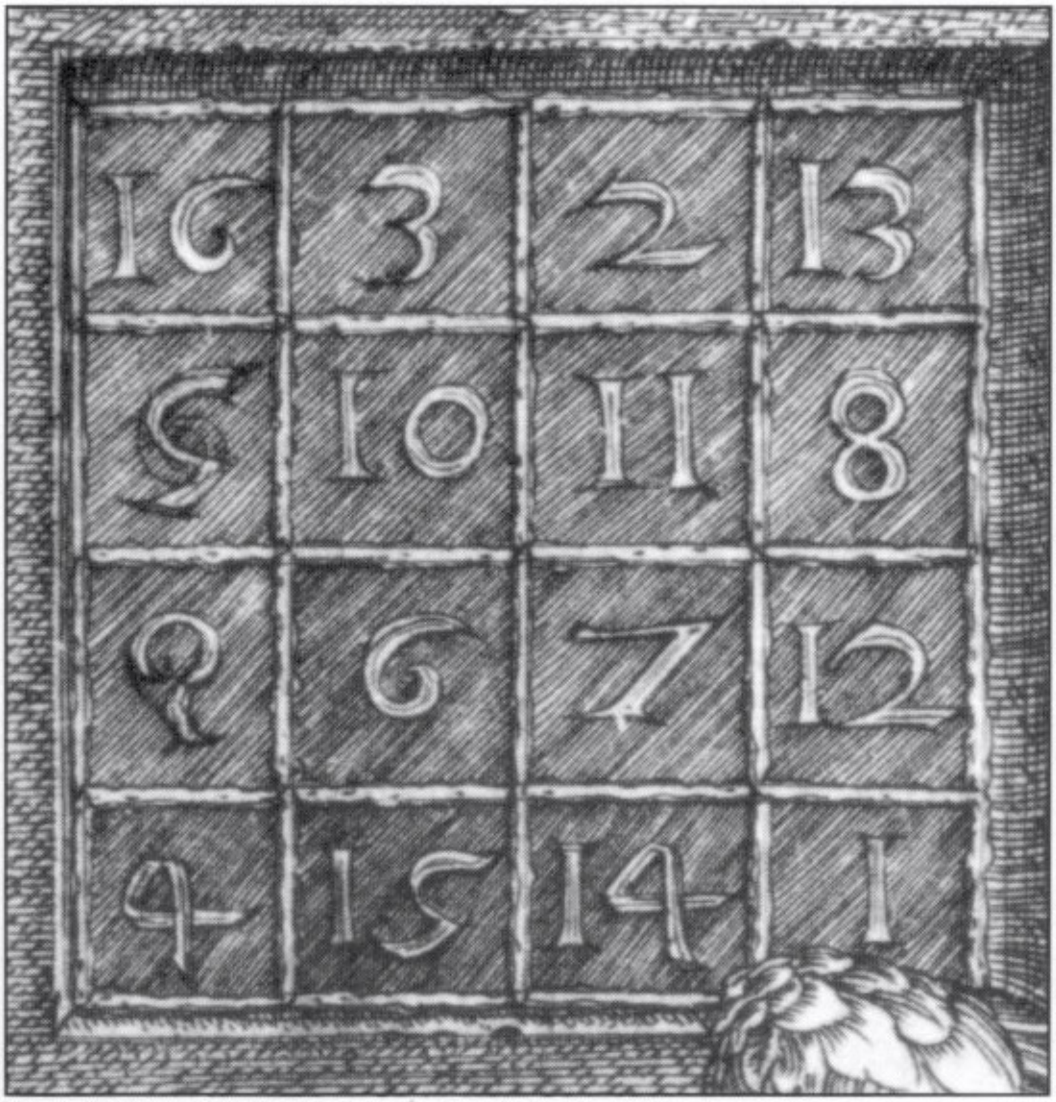
Out[423]= 9.51276×10^{-14}

Compression

- The image

In[424]:=
```
durer
```

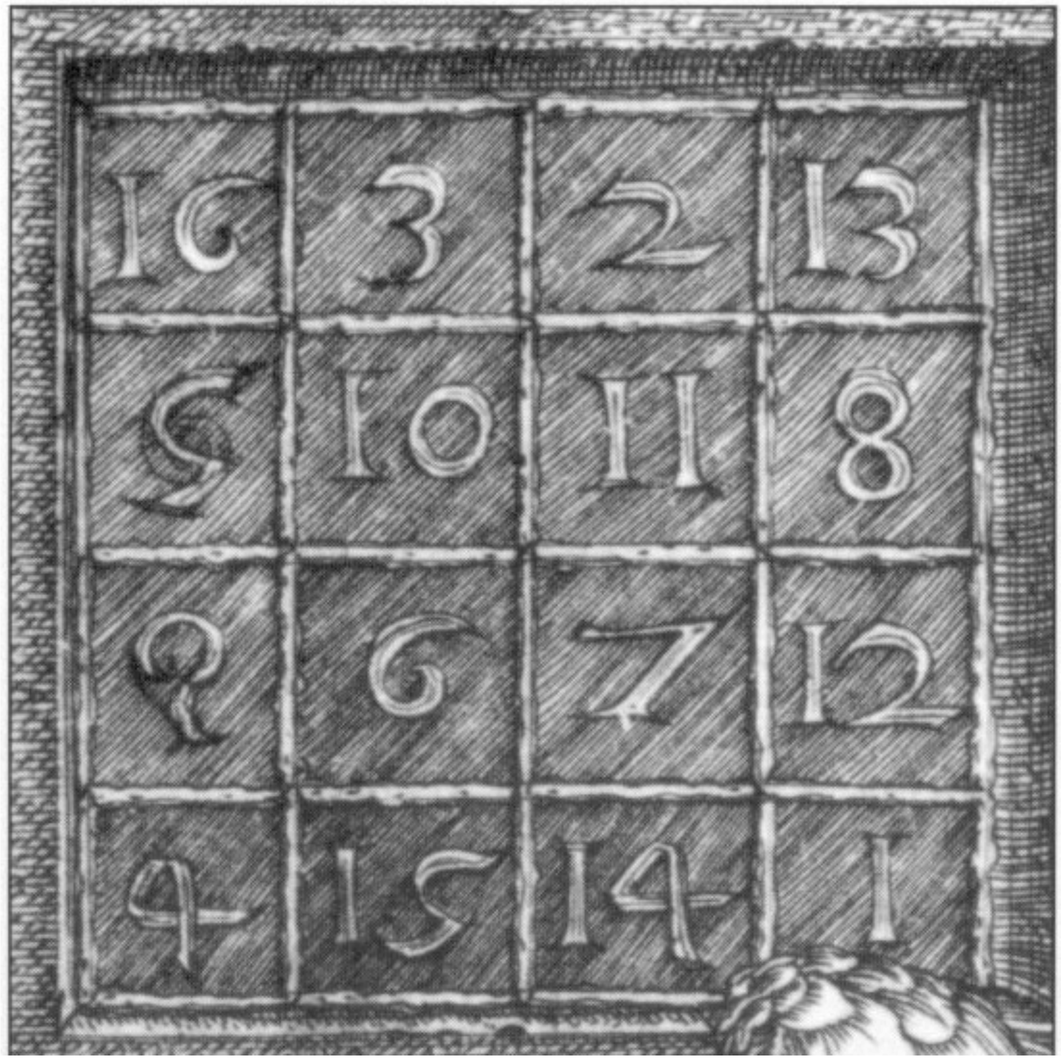
Out[424]=



take a square subimage of length divisible by 8 (to be able to perform a 3-level Haar transform)

In[425]:=
```
durerc = ColorConvert[ImageTake[durer, {1, 552}, {1, 552}], "Grayscale"]
```

Out[425]=



maximum and minimum of grey values

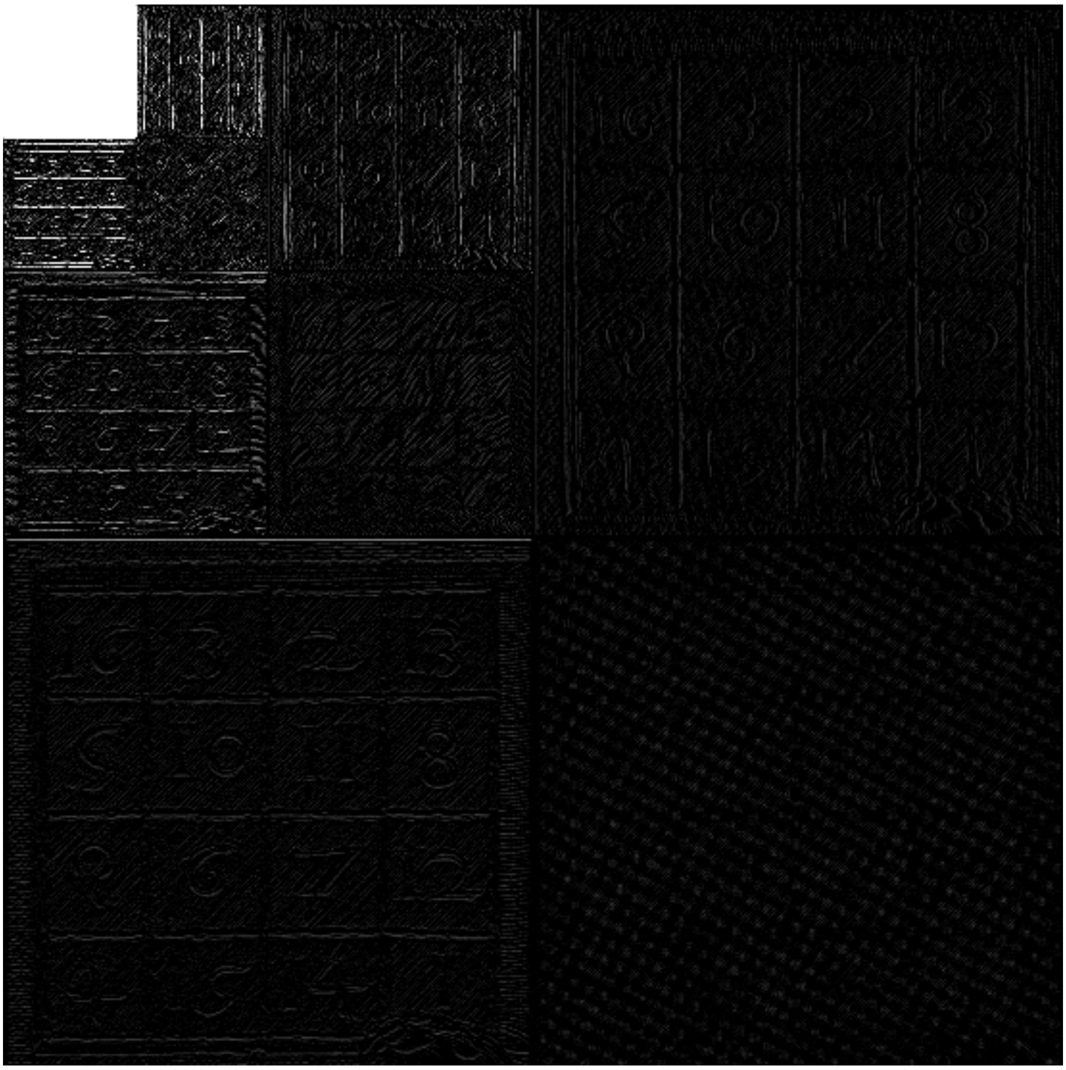
In[426]:=
```
durercdata = ImageData[durerc]; {Max[durercdata], Min[durercdata]}
```

Out[426]=
```
{1., 0.0117647}
```

Three-level Haar transform

In[427]:=
```
H3durerc = Htrans[durerc, 3]
```

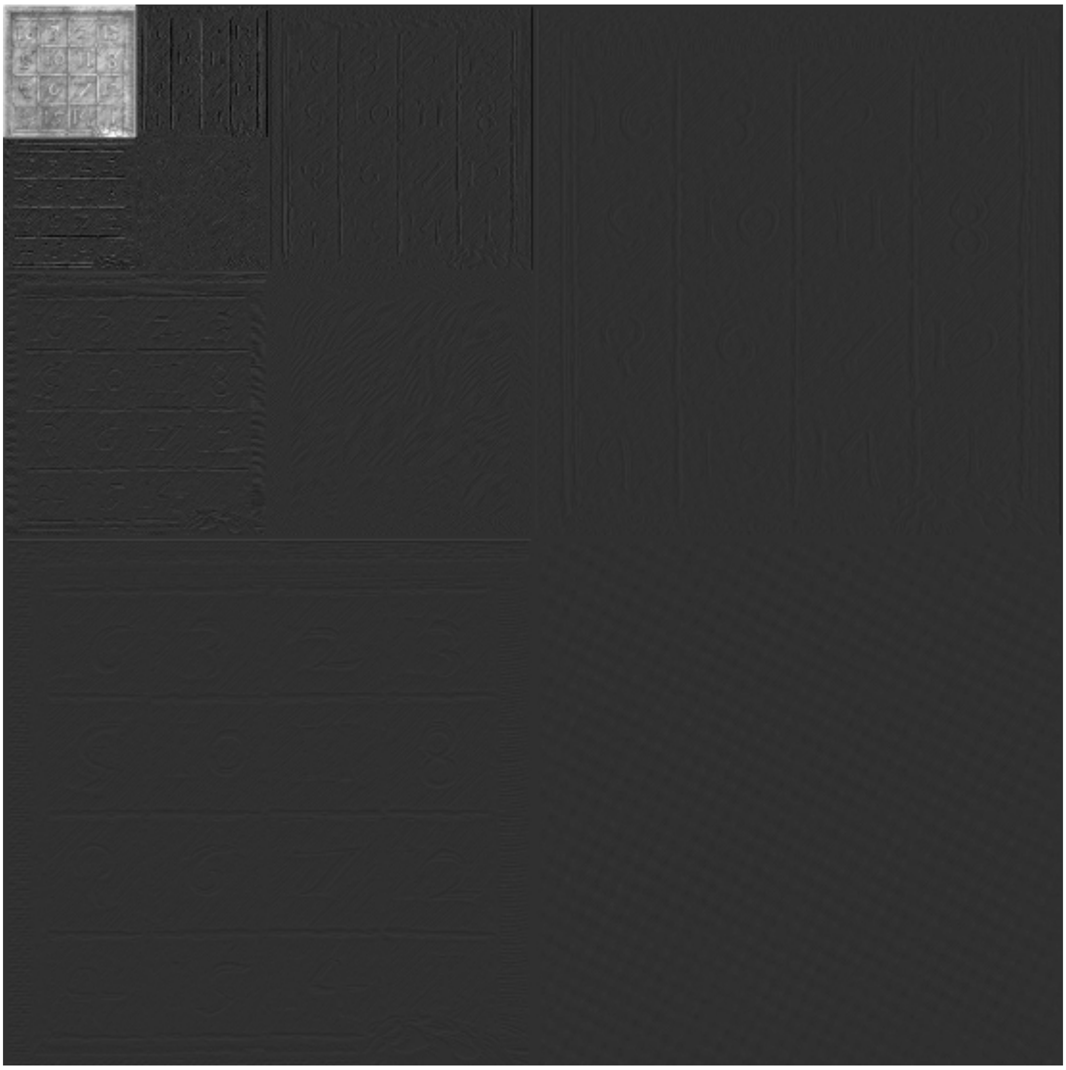
Out[427]=



intensities adjusted

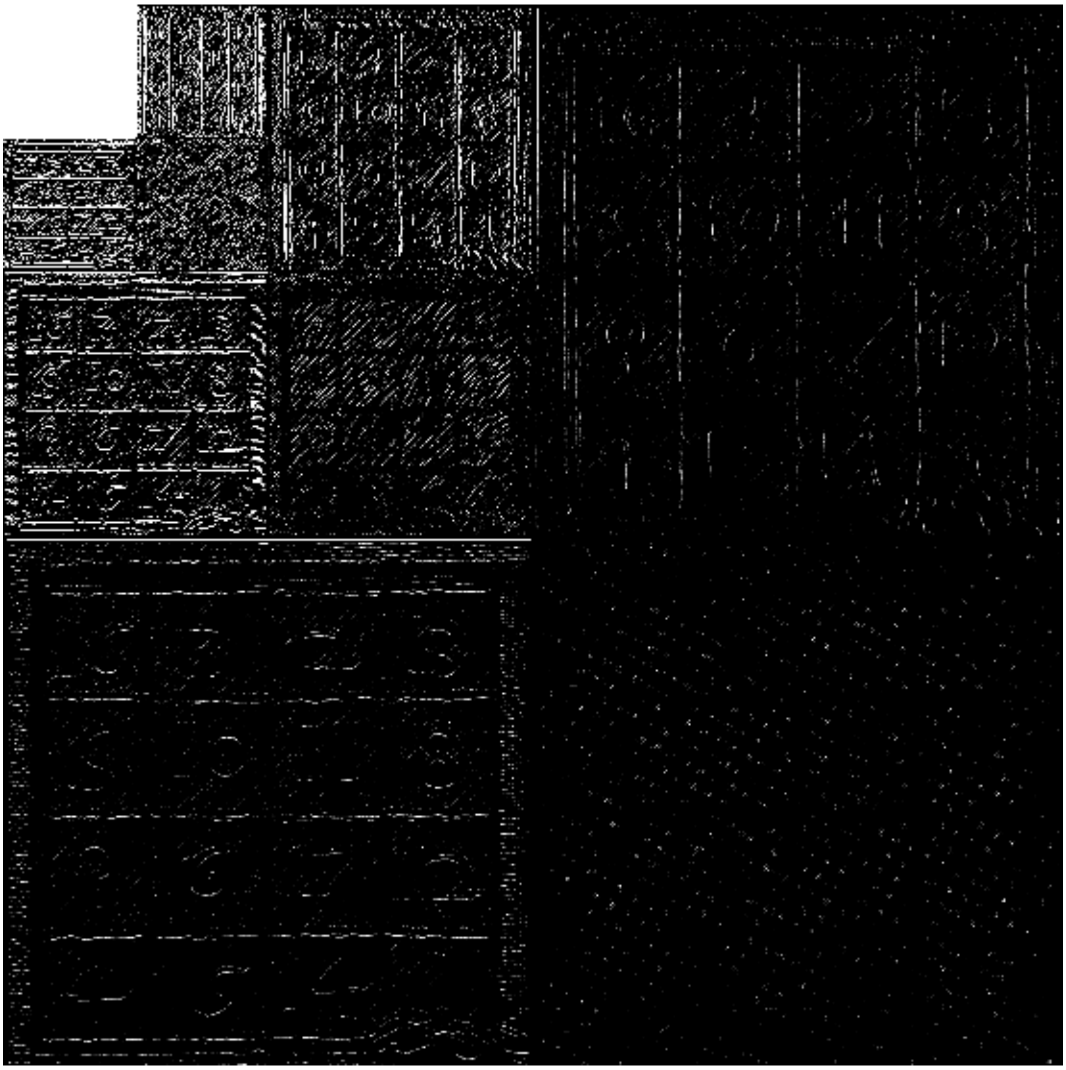
In[428]:=
```
ImageAdjust[H3durerc]
```

Out[428]=



In[429]:=
```
ImageAdjust[H3durerc, {2, 2}]
```

Out[429]=



the range of the values has been extended!

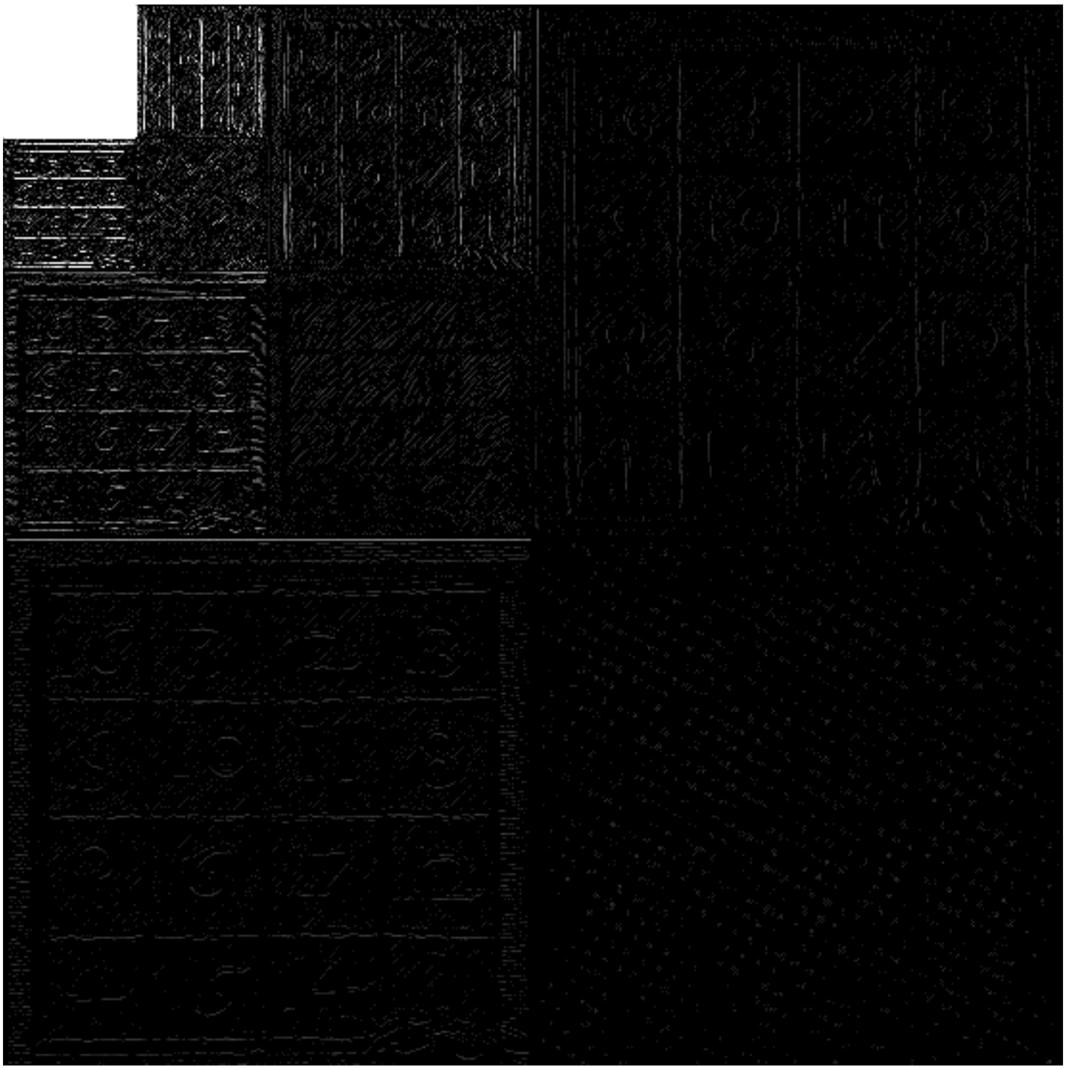
In[430]:=
```
H3durercdata = ImageData[H3durerc];
{Max[H3durercdata], Min[H3durercdata]}
```

Out[430]=
```
{7.79608, -1.8348}
```

- Thresholding with level 0.1

In[431]:=
```
new1 = Threshold[H3durerc, 0.1]
```

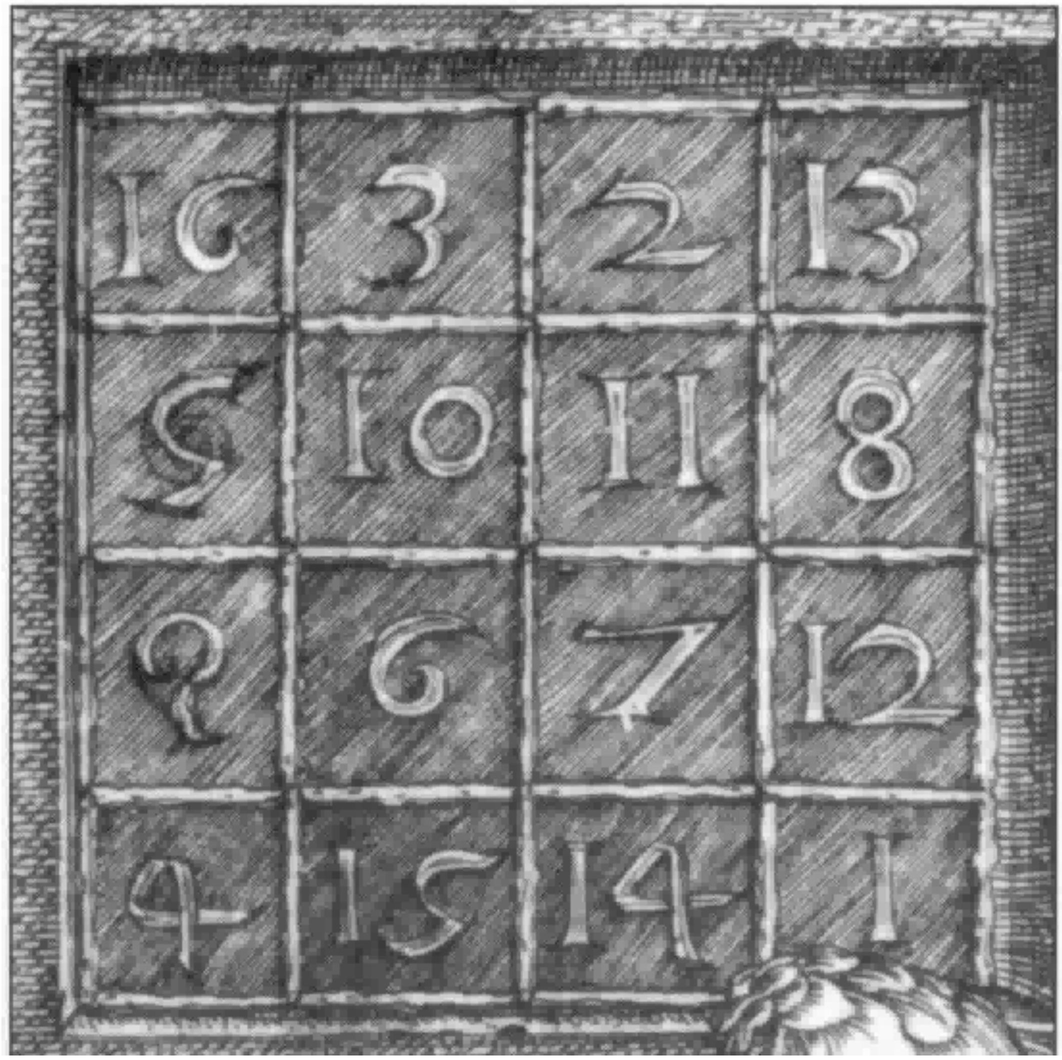
Out[431]=



taking inverse 3-level HT of thresholded data

In[432]:=
```
iHtrans[new1, 3]
```

Out[432]=



counting nonzero coefficients

In[433]:=
```
count[A_]:=Module[{AA},
AA=Map[If[#≠0,1,0]&,Flatten[A]];
Total[AA]
]
```

In[434]:=
```
count[ImageData[new1]]
```

Out[434]= 65 605

In[435]:=
```
count[ImageData[durerc]]
```

Out[435]= 304 704

In[436]:=
```
N[%% / %]
```

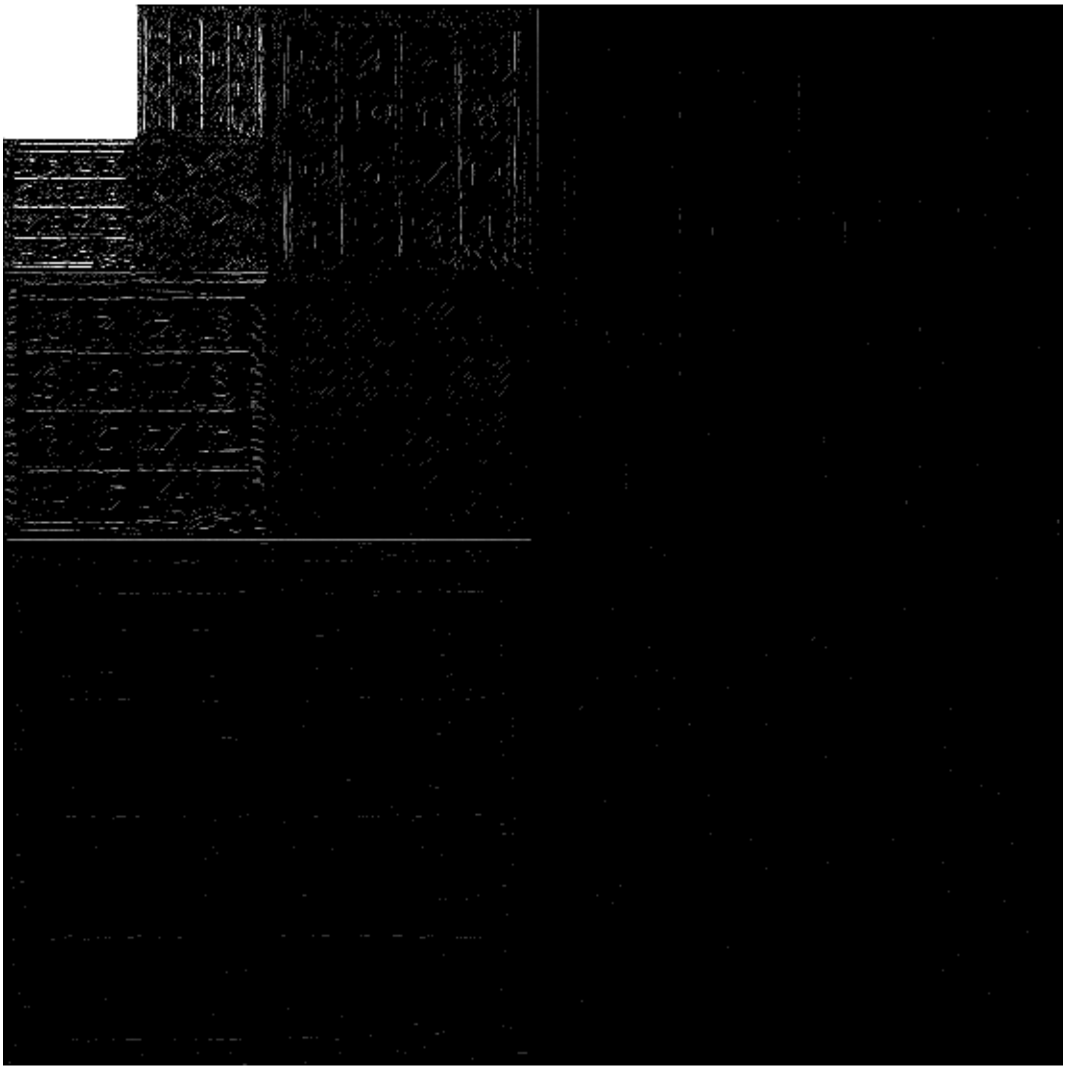
Out[436]= 0.215307

so the reconstruction with level 0.1 used only about 21% of the original amount of data!

- Now thresholding with level 0.2

In[437]:=
```
new2 = Threshold[H3durerc, 0.2]
```

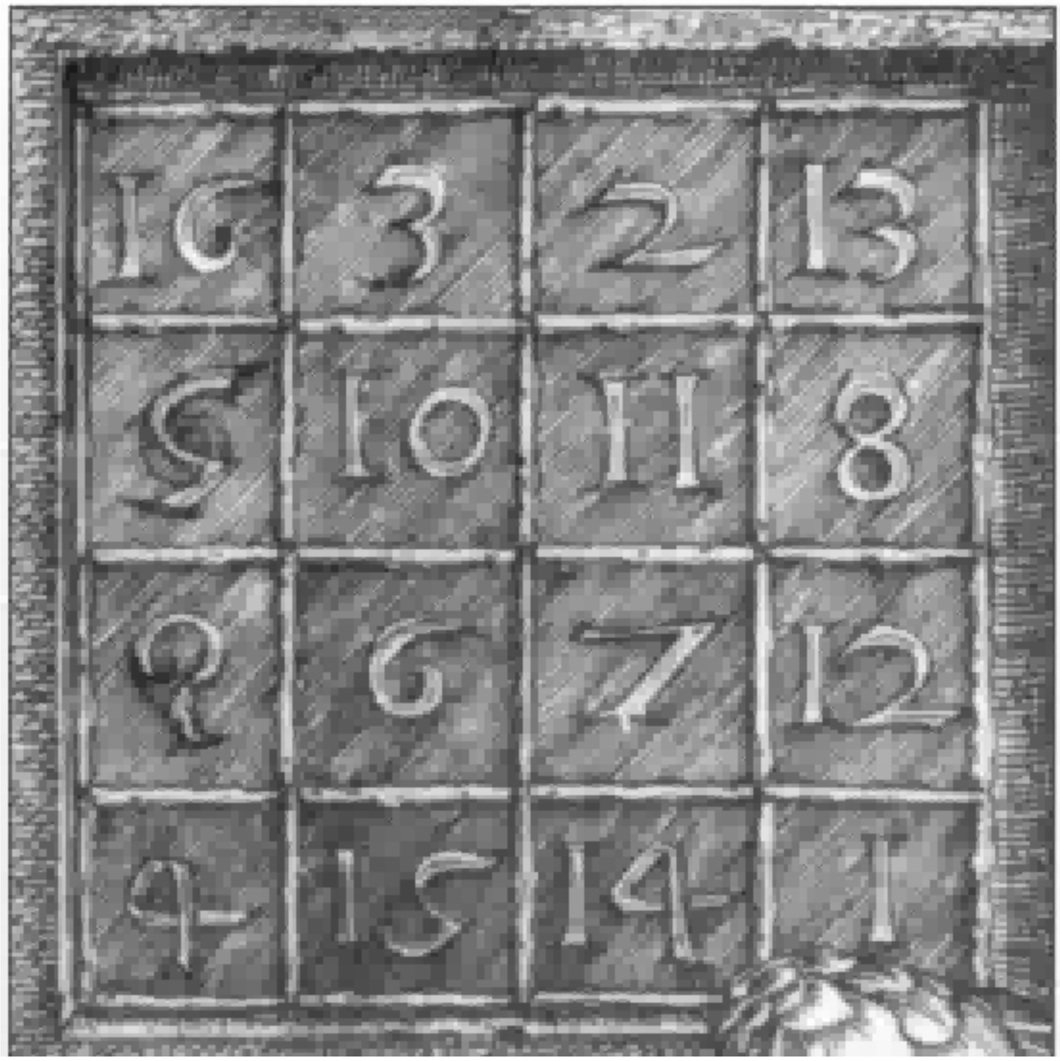
Out[437]=



taking inverse 3-level HT of thresholded data

In[438]:=
```
iHtrans[new2, 3]
```

Out[438]=



counting nonzero coefficients

In[439]:=
```
count[ImageData[new2]]
```

Out[439]= 21 242

In[440]:=
```
N[% / count[ImageData[durerc]]]
```

Out[440]= 0.0697136

reconstruction with threshold level 0.2 uses less than 7% of the amount of the original data

■ Using *Mathematica*'s built-in features

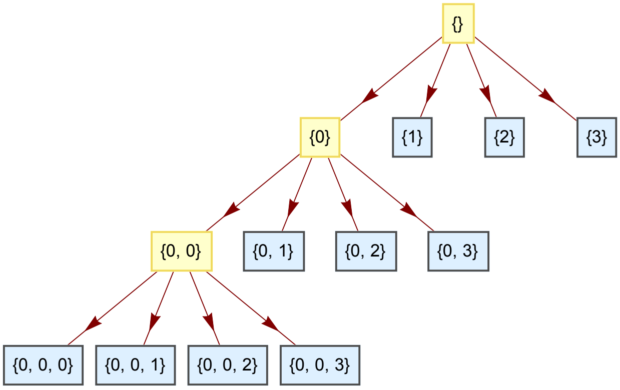
In[441]:=
```
dwd1 = DiscreteWaveletTransform[durerc, HaarWavelet[], 3]
```

Out[441]=
```
DiscreteWaveletData[ Data dimensions: {552, 552}
                     Refinements: 3 ]
```

visualizing the decomposition structure

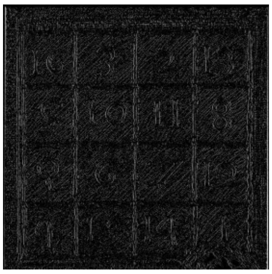
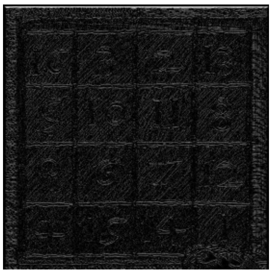
In[442]:=
```
dwd1["TreeView"]
```

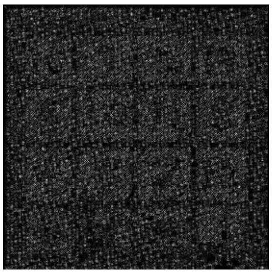
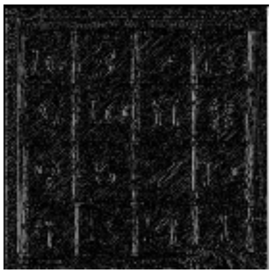
Out[442]=

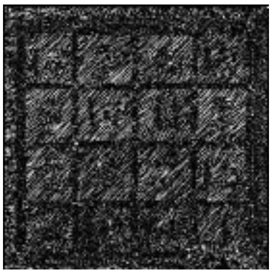


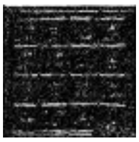
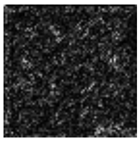
displaying all generated subimages

In[443]:= `dwd1[All, "Image"]`

Out[443]= {{0} → , {1} → , {2} → ,

{3} → , {0, 0} → , {0, 1} → ,

{0, 2} → , {0, 3} → , {0, 0, 0} → ,

{0, 0, 1} → , {0, 0, 2} → , {0, 0, 3} → }

energy distribution among subimages

In[444]:=
```
dwd1["EnergyFraction"] // MatrixForm
```

Out[444]//MatrixForm=

```
{1} → 0.00422299
{2} → 0.0054012
{3} → 0.00245287
{0, 1} → 0.0077127
{0, 2} → 0.00730357
{0, 3} → 0.00196534
{0, 0, 1} → 0.0124388
{0, 0, 2} → 0.0127398
{0, 0, 3} → 0.00206025
{0, 0, 0} → 0.943703
```

- Thresholding with level 0.15

In[445]:=
```
thr1 = WaveletThreshold[dwd1, {"Soft", 0.15}]
```

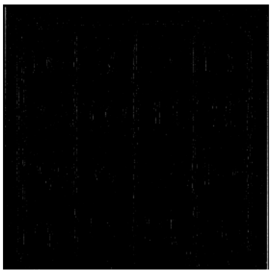
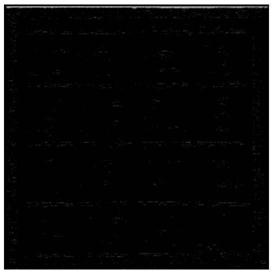
Out[445]= DiscreteWaveletData[Data dimensions: {552, 552} Refinements: 3]

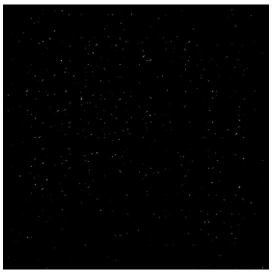
In[446]:=
```
thr1["ThresholdTable"]
```

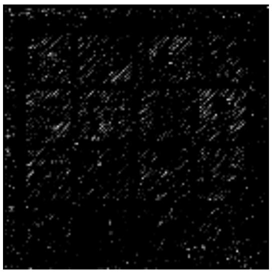
Out[446]=

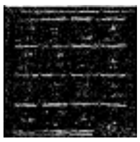
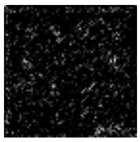
Wavelet Index	Threshold Value Channel 1
{1}	0.15
{2}	0.15
{3}	0.15
{0, 1}	0.15
{0, 2}	0.15
{0, 3}	0.15
{0, 0, 1}	0.15
{0, 0, 2}	0.15
{0, 0, 3}	0.15

In[447]:= `thr1[All, "Image"]`

Out[447]= {{0} → , {1} → , {2} → ,

{3} → , {0, 0} → , {0, 1} → ,

{0, 2} → , {0, 3} → , {0, 0, 0} → ,

{0, 0, 1} → , {0, 0, 2} → , {0, 0, 3} → }

counting nonzero coefficients

In[448]:= `Map[#[[1]] -> count[#[[2]]] &, thr1[Automatic]] // MatrixForm`

Out[448]//MatrixForm=

```
{1} → 2734
{2} → 4200
{3} → 809
{0, 1} → 5601
{0, 2} → 6027
{0, 3} → 2510
{0, 0, 1} → 2866
{0, 0, 2} → 2827
{0, 0, 3} → 1811
{0, 0, 0} → 4761
```

In[449]:= `Total[Map[#[[2]] &, %]]`

Out[449]= 34 146

In[450]:= `Map[#[[1]] -> count[#[[2]]] &, dwd1[Automatic]] // MatrixForm`

Out[450]//MatrixForm=

```
{1} → 75 379
{2} → 75 673
{3} → 75 468
{0, 1} → 19 000
{0, 2} → 19 020
{0, 3} → 19 004
{0, 0, 1} → 4754
{0, 0, 2} → 4761
{0, 0, 3} → 4761
{0, 0, 0} → 4761
```

In[451]:=
```
Total[Map[#[[2]] &, %]]
```

Out[451]= 302 581

In[452]:=
```
N[34 146 / 302 581]
```

Out[452]= 0.112849

reconstruction with threshold level 1.5 uses about 11% of the amount of the original data

reconstruction from thresholded data

In[453]:=
```
Image[InverseWaveletTransform[thr1, HaarWavelet[], 3]]
```

Out[453]=

Denoising

- Data with noise

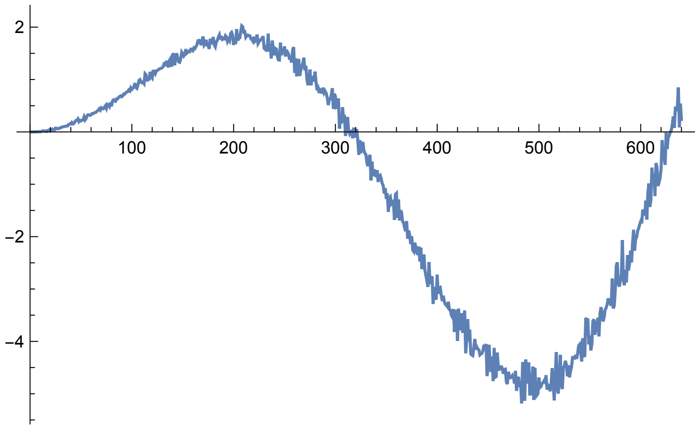
In[454]:=
```
data2 = Table[x * Sin[x] + x * Sin[10 x] RandomReal[{-0.1, 0.1}] +
    Exp[- (x - Pi) ^2] RandomReal[{-0.15, 0.15}], {x, 0, 6.39, 0.01}];
```

In[455]:=
```
Length[data2]
```

Out[455]= 640

In[456]:=
```
ListLinePlot[data2]
```

Out[456]=



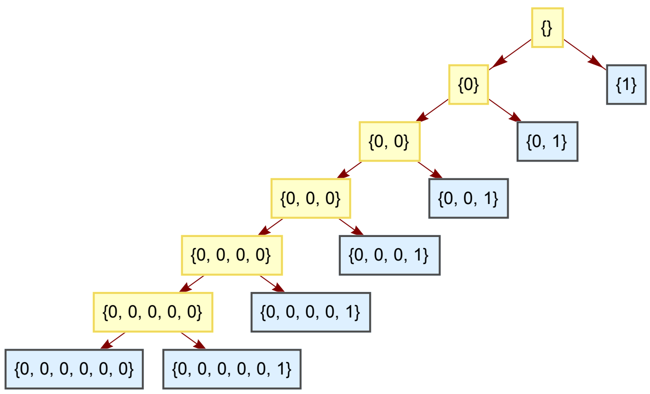
- Haar-DWT of noised data over 6 levels

In[457]:=
```
dwd2 = DiscreteWaveletTransform[data2, HaarWavelet[], 6];
```

schematic view of decomposition

In[458]:=
```
dwd2["TreeView"]
```

Out[458]=



energy distribution over subimages

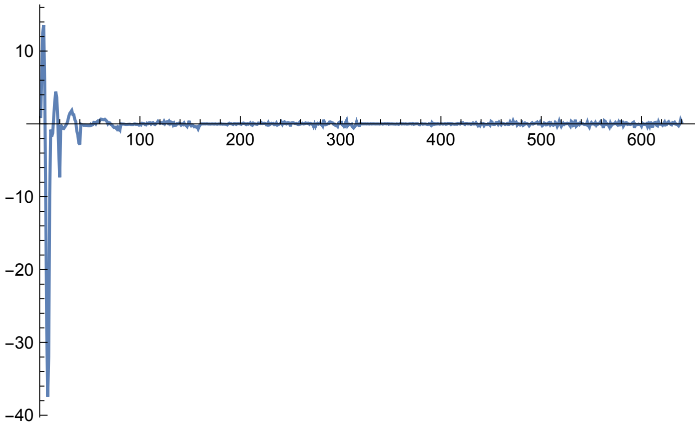
In[459]:=
```
efrac = dwd2["EnergyFraction"]; efrac // MatrixForm
```

Out[459]//MatrixForm=
```
{1} → 0.00186285
{0, 1} → 0.000927986
{0, 0, 1} → 0.000715752
{0, 0, 0, 1} → 0.00160494
{0, 0, 0, 0, 1} → 0.00811972
{0, 0, 0, 0, 0, 1} → 0.0293781
{0, 0, 0, 0, 0, 0} → 0.957391
```

■ The transformed data

In[460]:=
```
ListLinePlot[Flatten[Reverse[dwd2[Automatic, "Values"]]], PlotRange → All]
```

Out[460]=



■ Eliminating low-energy subimages

In[461]:=
```
eth[x_,ind_]:=
If[(ind/.efrac) < 0.005, x*0.,x]/;MemberQ[efrac[[All,1]],ind]

eth[x_,___]:=x
```

In[463]:=
```
dwd2a = WaveletMapIndexed[eth, dwd2];
```

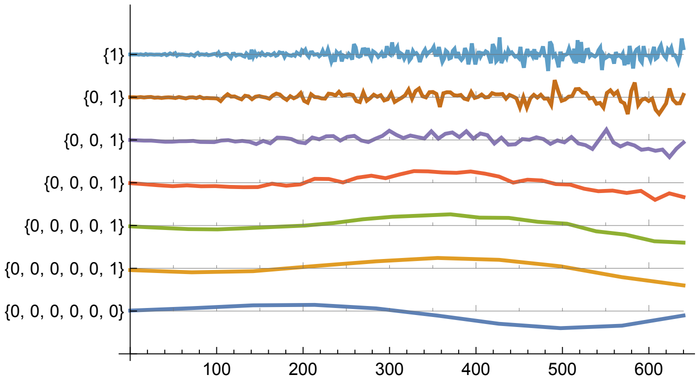
energy distribution for modified data

In[464]:=
```
dwd2a["EnergyFraction"] // MatrixForm
```

Out[464]//MatrixForm=
```
{1} → 0.
{0, 1} → 0.
{0, 0, 1} → 0.
{0, 0, 0, 1} → 0.
{0, 0, 0, 0, 1} → 0.00816143
{0, 0, 0, 0, 0, 1} → 0.0295291
{0, 0, 0, 0, 0, 0} → 0.96231
```

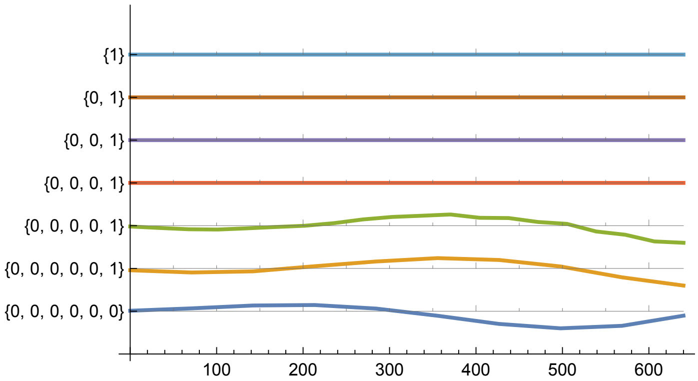
In[465]:=
```
WaveletListPlot[dwd2, Automatic, PlotLayout → "CommonXAxis",
  DataRange → {0, 640}, PlotStyle → Thick, Ticks → Full]
```

Out[465]=



In[466]:=
```
WaveletListPlot[dwd2a, Automatic, PlotLayout → "CommonXAxis",
  DataRange → {0, 640}, PlotStyle → Thick, Ticks → Full]
```

Out[466]=



- Inverting both data sets

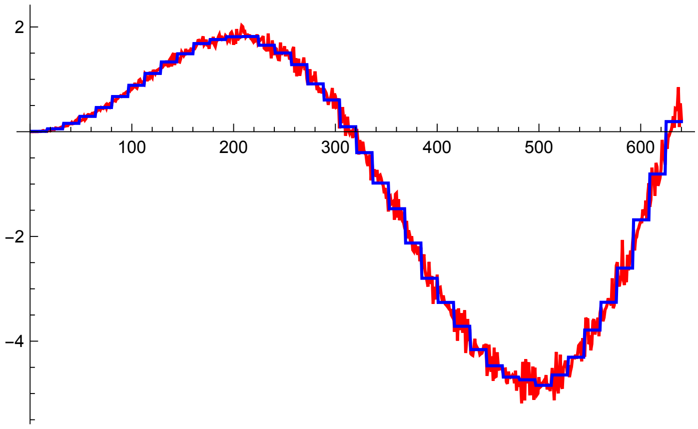
In[467]:=
```
iwt2 = InverseWaveletTransform[dwd2];
```

In[468]:=
```
iwt2a = InverseWaveletTransform[dwd2a];
```

comparison of reconstructions

In[469]:=
```
ListLinePlot[{iwt2, iwt2a }, PlotStyle → {Red, Blue}]
```

Out[469]=



- Same procedure with a more sophisticated wavelet transform

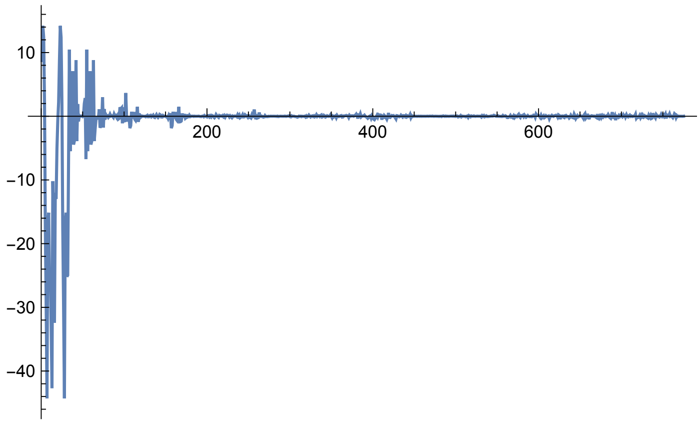
In[470]:=
```
dwd3 = DiscreteWaveletTransform[data2, DaubechiesWavelet[12], 6];
```

In[471]:=
```
efrac = dwd3["EnergyFraction"]; efrac // MatrixForm
```

Out[471]//MatrixForm=
```
{1} → 0.000465429
{0, 1} → 0.000230096
{0, 0, 1} → 0.000306106
{0, 0, 0, 1} → 0.00134137
{0, 0, 0, 0, 1} → 0.00308935
{0, 0, 0, 0, 0, 1} → 0.0476307
{0, 0, 0, 0, 0, 0} → 0.946937
```

In[472]:=
```
ListLinePlot[Flatten[Reverse[dwd3[Automatic, "Values"]]], PlotRange → All]
```

Out[472]=



In[473]:=
```
dwd3a = WaveletMapIndexed[eth, dwd3];
```

In[474]:=
```
dwd3a["EnergyFraction"] // MatrixForm
```

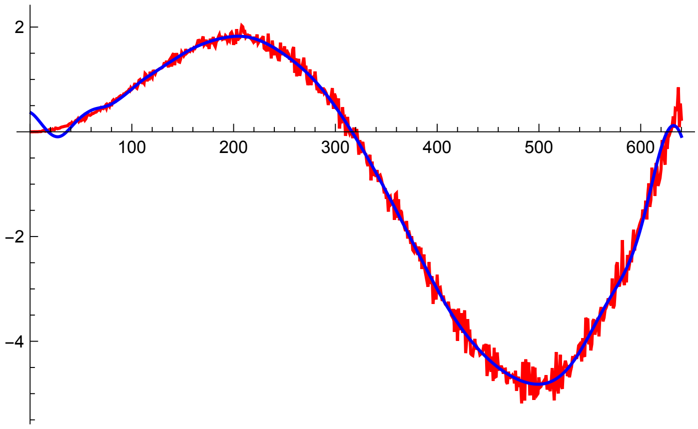
Out[474]//MatrixForm=
```
{1} → 0.
{0, 1} → 0.
{0, 0, 1} → 0.
{0, 0, 0, 1} → 0.
{0, 0, 0, 0, 1} → 0.
{0, 0, 0, 0, 0, 1} → 0.0478909
{0, 0, 0, 0, 0, 0} → 0.952109
```

In[475]:=
```
iwt3 = InverseWaveletTransform[dwd3];
```

In[476]:=
```
iwt3a = InverseWaveletTransform[dwd3a];
```

In[477]:=
```
ListLinePlot[{iwt3, iwt3a }, PlotStyle → {Red, Blue}]
```

Out[477]=



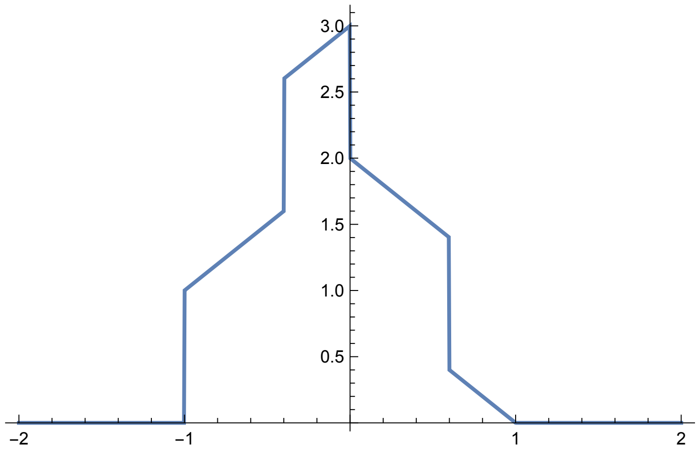
Edge detection

■ Data for edge detection

In[478]:=
```
data4 = Table[
   HeavisideLambda[x] + UnitBox[x - 0.1] + UnitBox[x + 0.5], {x, -2, 2, 4/1023}];
```

In[479]:=
```
ListLinePlot[data4, DataRange → {-2, 2}, PlotStyle → Thick]
```

Out[479]=

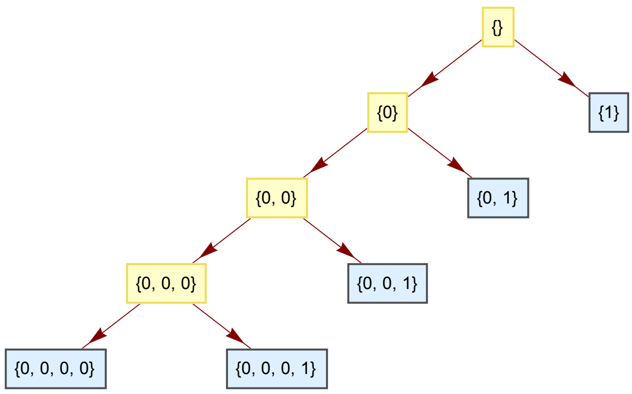


■ Wavelet transform over 4 levels

In[480]:=
```
dwd4 = DiscreteWaveletTransform[
   data4, DaubechiesWavelet[10], 4, Padding -> "Reflected"];
```

In[481]:=
```
dwd4[ "TreeView"]
```

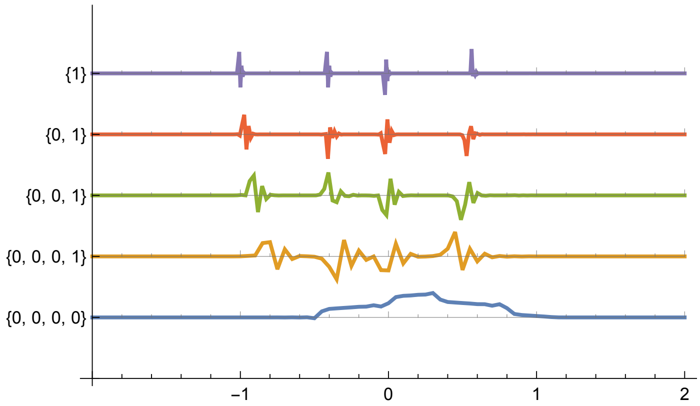
Out[481]=



comparison of the transformed data over a common axis

In[482]:=
```
WaveletListPlot[dwd4, Automatic,
 PlotLayout → "CommonXAxis",
 DataRange → {-2, 2},
 PlotStyle → Thick,
 Ticks → Full]
```

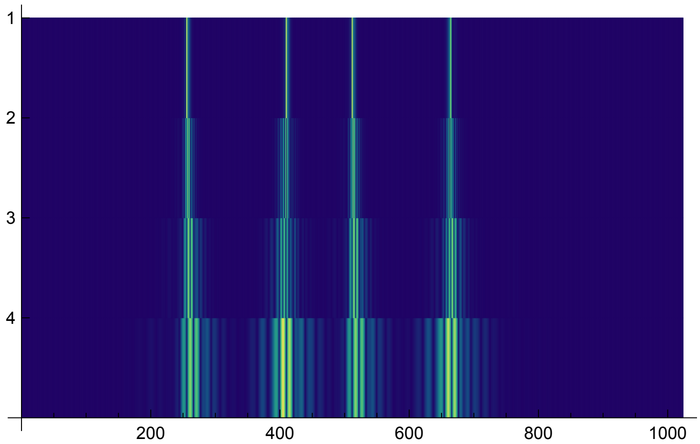
Out[482]=



■ The wavelet scalogram for visualization

In[483]:=
```
WaveletScalogram[dwd4, {___, 1},
 Method → "Inverse" → True,
 ColorFunction -> "BlueGreenYellow"]
```

Out[483]=

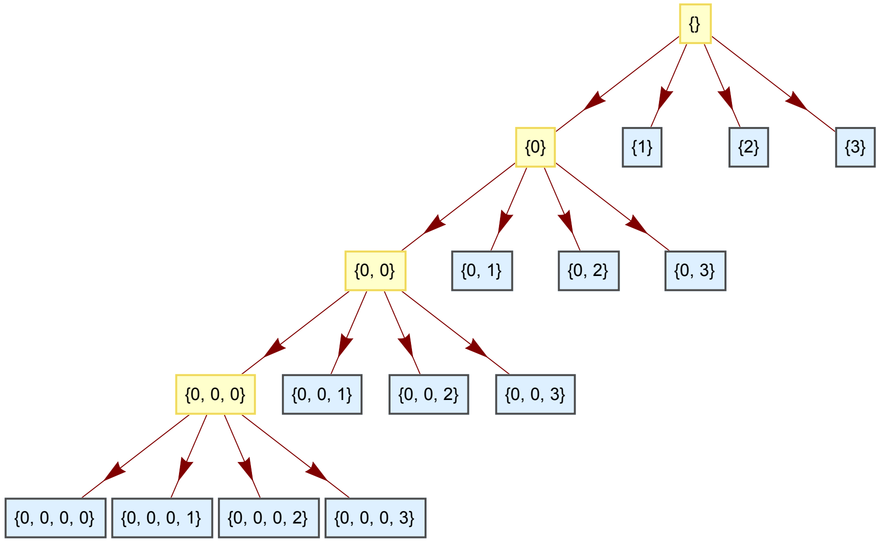


- Highlighting edges in the Durer image by eliminating the {0,0,0,0}-subimage

In[484]:=
```
dwd5 = DiscreteWaveletTransform[durer, SymletWavelet[2], 4];
```

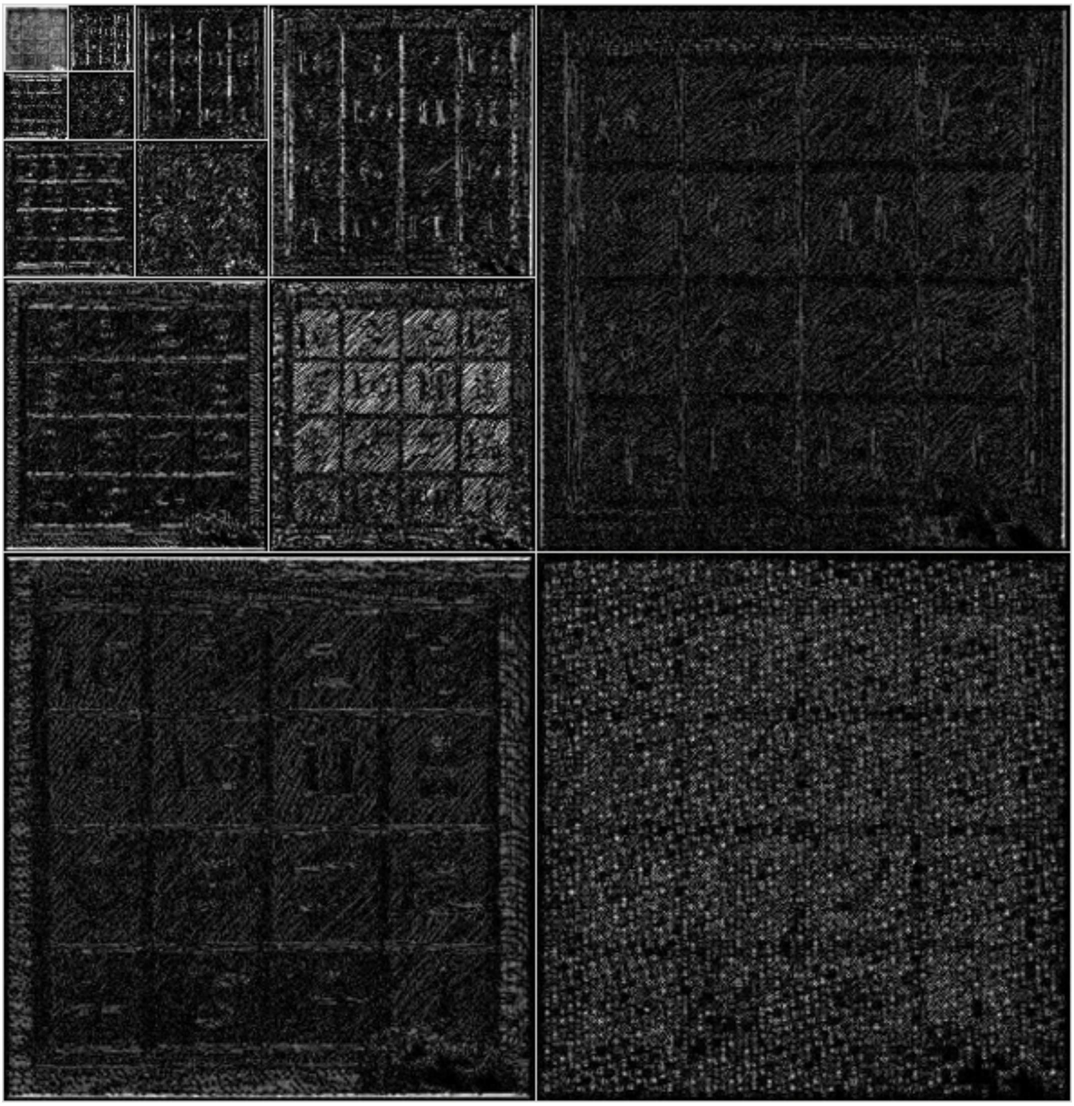
In[485]:=
```
dwd5[ "TreeView"]
```

Out[485]=



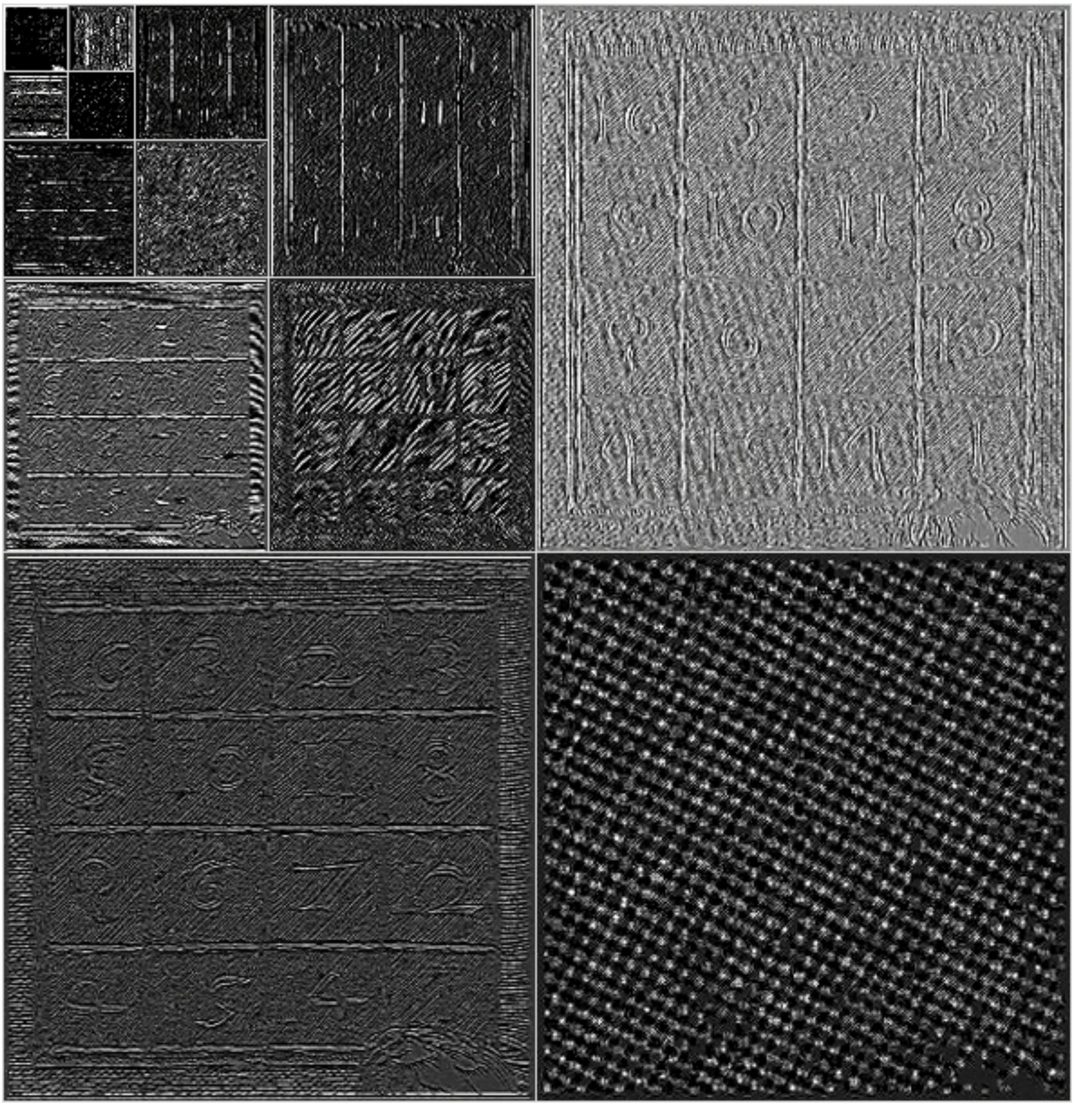
In[486]:= WaveletImagePlot[dwd5]

Out[486]=



In[487]:=
```
WaveletImagePlot[dwd5, Automatic, ImageAdjust[ImageAdjust[#], {1, 0.2, 1.9}] &]
```

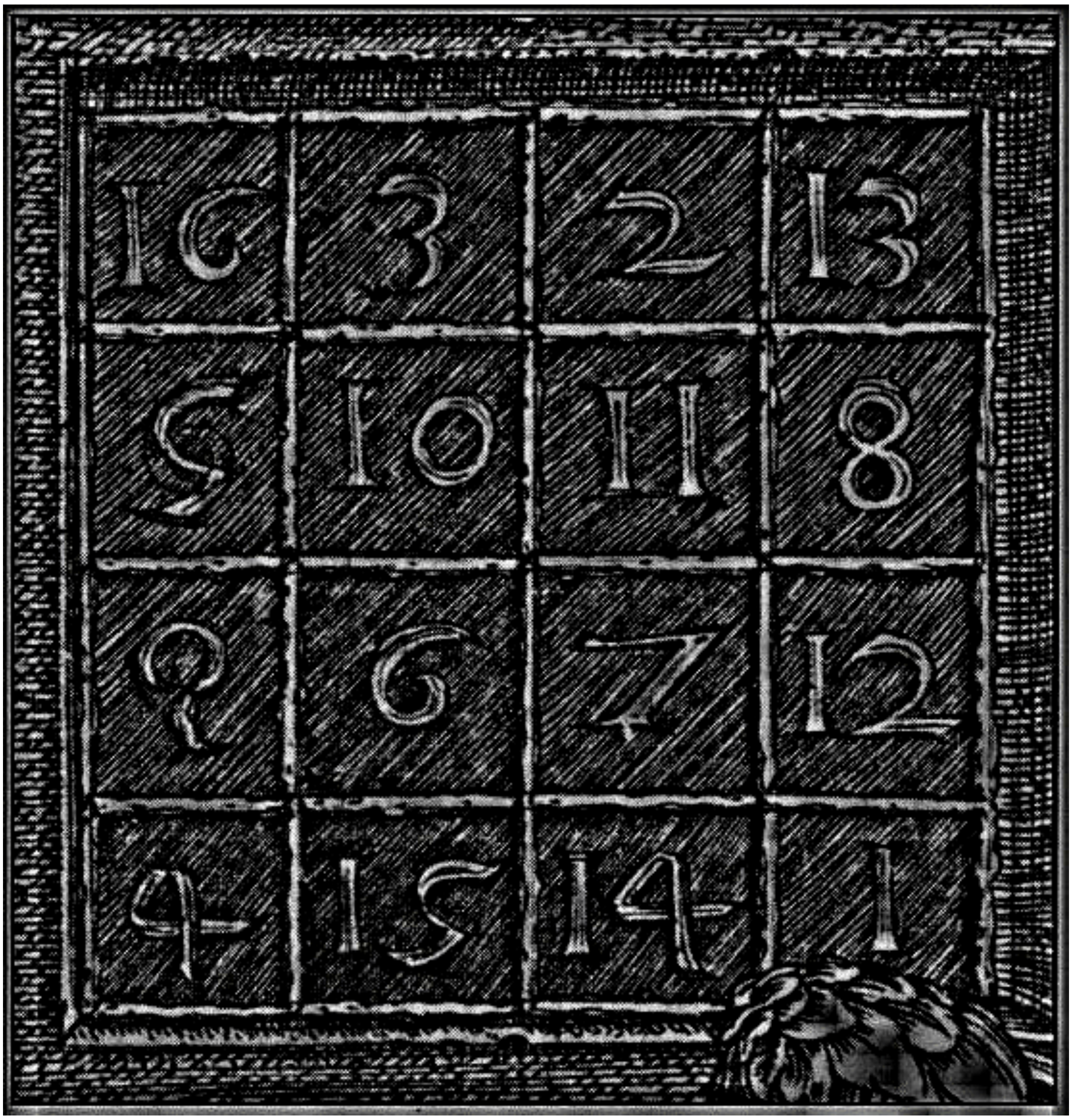
Out[487]=



In[488]:=
```
imgEdge[img_,{0,0,0,0}]:= ImageApply[# 0.0&,img]
imgEdge[img_,___]:= ImageApply[# 2&,img]
```

In[490]:=
```
Sharpen[InverseWaveletTransform[WaveletMapIndexed[imgEdge,dwd5]]]
```

Out[490]=



Frequency separation

■ Frequency data

In[491]:=
```
d1 = Table[Sin[x] + Cos[2 x], {x, -2 π, 2 π, 4 π/2047}];
```

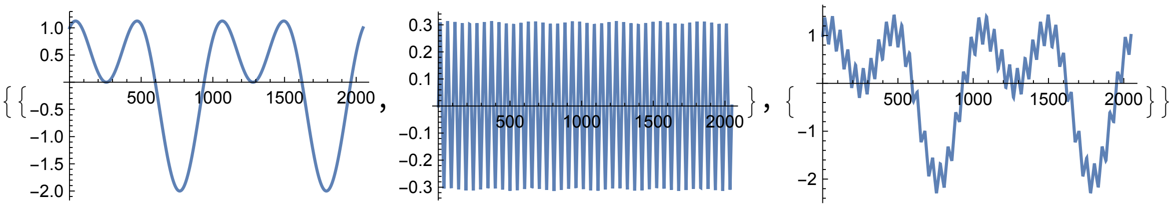
high-frequency data

In[492]:=
```
d2 = Table[1/5 ArcSin[Sin[20 x]], {x, -2 π, 2 π, 4 π/2047}];
```

superposition

In[493]:=
```
{{ListLinePlot[d1], ListLinePlot[d2]}, {ListLinePlot[d1 + d2]}}
```

Out[493]=



■ 6-level DWT of superposition

In[494]:=
```
dwd6 = DiscreteWaveletTransform[d1 + d2, SymletWavelet[4], 6];
```

energy distribution

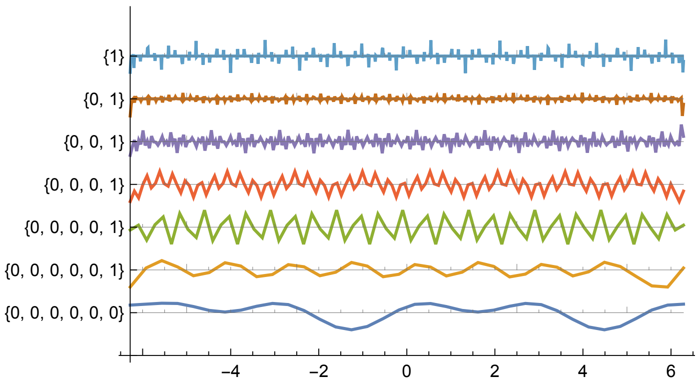
In[495]:=
```
dwd6["EnergyFraction"]
```

Out[495]=
```
{{1} → 1.06451 × 10^-6, {0, 1} → 3.0582 × 10^-6, {0, 0, 1} → 0.0000145693,
 {0, 0, 0, 1} → 0.0000864422, {0, 0, 0, 0, 1} → 0.00196998,
 {0, 0, 0, 0, 0, 1} → 0.0000866911, {0, 0, 0, 0, 0, 0} → 0.997838}
```

visualizing the transformed data

In[496]:=
```
WaveletListPlot[dwd6, PlotLayout → "CommonXAxis",
 Ticks → Full, DataRange → {-2 π, 2 π}]
```

Out[496]=



■ Filtering out the low-frequency data

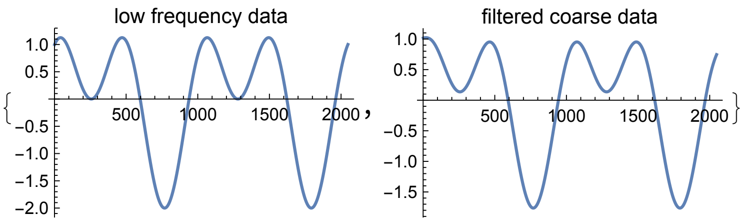
In[497]:=
```
rdwd6 = WaveletMapIndexed[(# * 0.0) &, dwd6, {___, 1}];
```

In[498]:=
```
rdwd6["EnergyFraction"]
```

Out[498]=
```
{{1} → 0., {0, 1} → 0., {0, 0, 1} → 0., {0, 0, 0, 1} → 0.,
 {0, 0, 0, 0, 1} → 0., {0, 0, 0, 0, 0, 1} → 0., {0, 0, 0, 0, 0, 0} → 1.}
```

In[499]:=
```
{ListLinePlot[d1, PlotLabel → "low frequency data"], ListLinePlot[
  InverseWaveletTransform[rdwd6], PlotLabel → "filtered coarse data"]}
```

Out[499]=



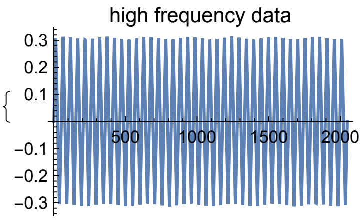
- Filtering out the high frequency data

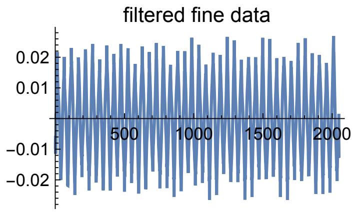
In[500]:=
```
rdwd7 = WaveletMapIndexed[(#*0.0) &, dwd6, {___, 0}];
```

In[501]:=
```
rdwd7["EnergyFraction"]
```

Out[501]=
```
{{1} → 0.000492419, {0, 1} → 0.00141465, {0, 0, 1} → 0.00673941,
 {0, 0, 0, 1} → 0.0399862, {0, 0, 0, 0, 1} → 0.911266,
 {0, 0, 0, 0, 0, 1} → 0.0401013, {0, 0, 0, 0, 0, 0} → 0.}
```

In[502]:=
```
{ListLinePlot[d2, PlotLabel → "high frequency data"],
 ListLinePlot[InverseWaveletTransform[rdwd7], PlotLabel → "filtered fine data"]}
```

Out[502]= { 

, 

}