Example for image denoising using wavelets

This Mathematica notebook uses a package for discrete wavelet transforms
by P. van Fleet with examples taken from his book *Discrete Wavelet Transforms*

In[48]:=
```
<< DiscreteWavelets`DiscreteWavelets`
```

SetDelayed::write : Tag Entropy in Entropy[v_] is Protected. >>

The images provided by the book

In[49]:=
```
ImageList[]
```

```
The base directory for the images is /Users/strehl/Library/Mathematica
  and the images courtesy of Dr. Radka Turcajova.

The naming convention for the thumbnail image is to add a _small to the name.
  For example, the thumbnail image for benches.png is benches_small.png.

The number in parentheses represents the
  dimensions and max iterations for the thumbnail image.
```

Out[49]//TableForm=

File Name	Type	Rows	Columns	Max Iterations
benches.png	Gray	512(160)	768(240)	8(4)
car.png	Gray	512(160)	768(240)	8(4)
chairs.png	Gray	512(160)	768(240)	8(4)
chess.png	Gray	768(240)	512(160)	8(4)
church.png	Gray	768(240)	512(160)	8(4)
clown.png	Gray	768(240)	512(160)	8(4)
dog.png	Gray	512(160)	768(240)	8(4)
dome.png	Gray	768(240)	512(160)	8(4)
heads.png	Gray	768(240)	512(160)	8(4)
landscape.png	Gray	512(160)	768(240)	8(4)
pigeon.png	Gray	512(160)	768(240)	8(4)
road.png	Gray	512(160)	768(240)	8(4)
sol.png	Gray	512(160)	768(240)	8(4)
surfer.png	Gray	512(160)	768(240)	8(4)
tower.png	Gray	768(240)	512(160)	8(4)
train.png	Gray	512(160)	768(240)	8(4)
tram.png	Gray	512(160)	768(240)	8(4)
yucca.png	Gray	768(240)	512(160)	8(4)
bridge.png	Color	512(160)	768(240)	8(4)
building.png	Color	786(240)	512(160)	8(4)
cups.png	Color	512(160)	768(240)	8(4)
facade.png	Color	512(160)	768(240)	8(4)
fish.png	Color	512(160)	768(240)	8(4)
goats.png	Color	512(160)	768(240)	8(4)
legos.png	Color	786(240)	512(160)	8(4)
telescope.png	Color	786(240)	512(160)	8(4)
waves.png	Color	786(240)	512(160)	8(4)

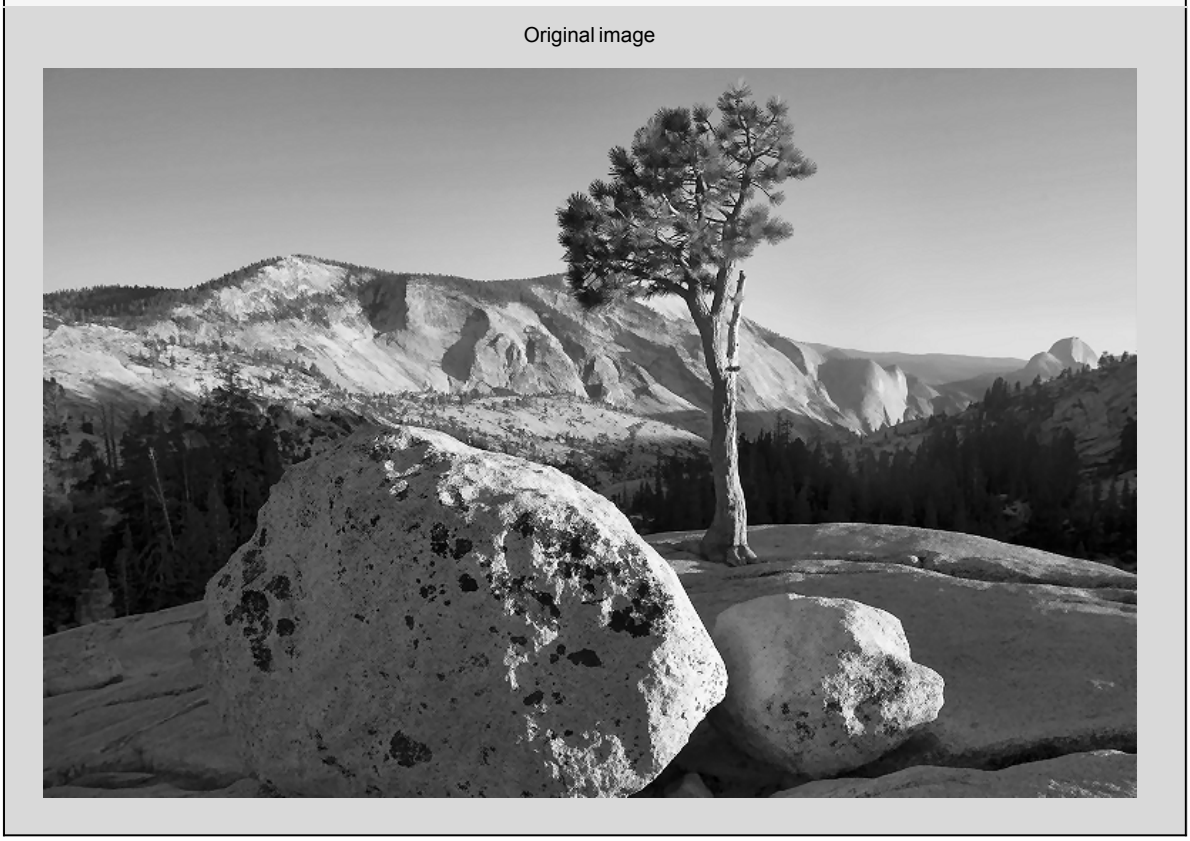
In[50]:=
```
path =
  "~/Library/Mathematica/Applications/DiscreteWavelets/Images/GrayScale/"
```

Out[50]=
```
~/Library/Mathematica/Applications/DiscreteWavelets/Images/GrayScale/
```

An example image taken from the book

In[51]:=
```
A = ImageRead[ToString[path <> "landscape.png"]];
p = ImagePlot[A, PlotLabel → "Original image"];
Show[p, ImageSize → Full]
```

Out[53]=



The image and its one- and two-level wavelet transforms using the D12 wavelet

In[54]:=
```
p0 = ImagePlot[A, PlotLabel -> "Original image"];
h = Daub[12];
B1 = WT2D1[N[A], N[h]];
p1 = WaveletDensityPlot[B1, PlotLabel -> "WT--one Phase"];
B2 = WT2D[N[A], N[h], NumIterations → 2];
p2 = WaveletDensityPlot[B2, PlotLabel -> "WT--two Phases"];
```

In[60]:=
```
Show[p0, ImageSize → Full]
```

Out[60]=

Original image

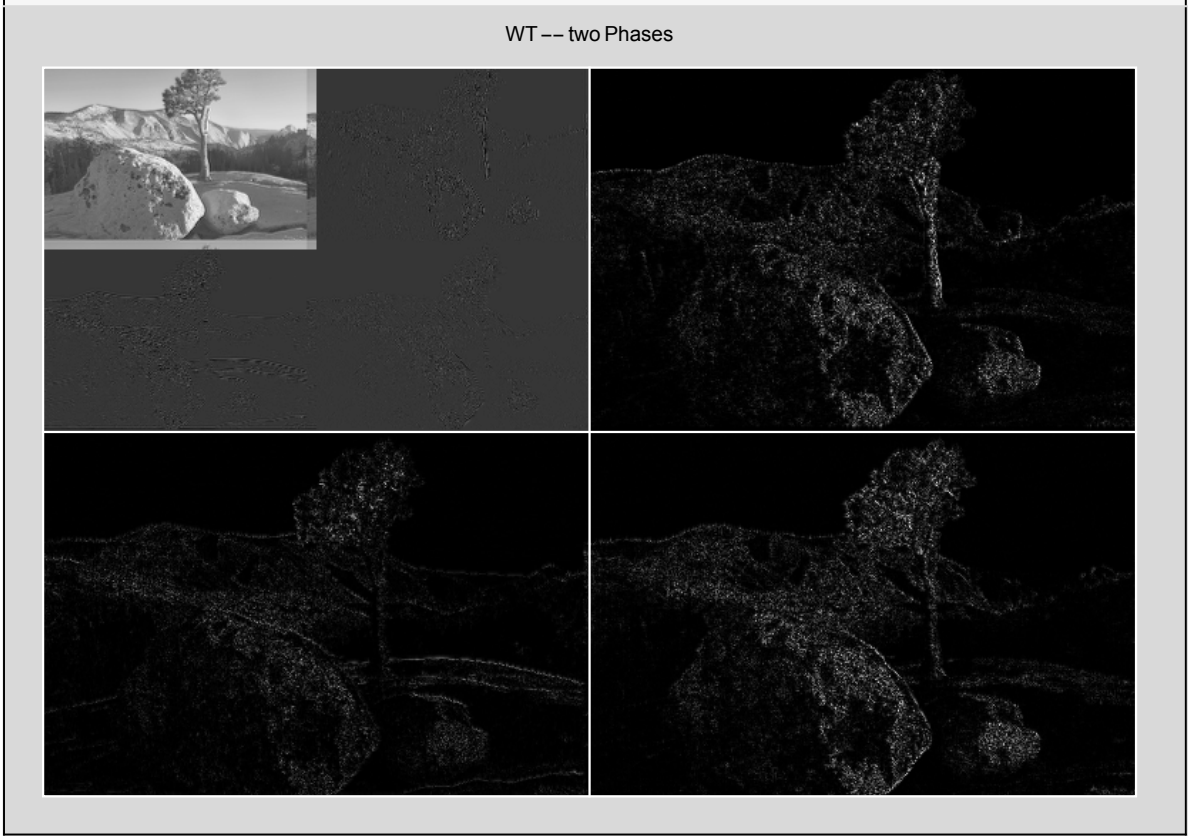
In[61]:=
```
Show[p1, ImageSize → Full]
```

Out[61]=

WT -- one Phase

In[62]:=
```
Show[p2, ImageSize → Full]
```

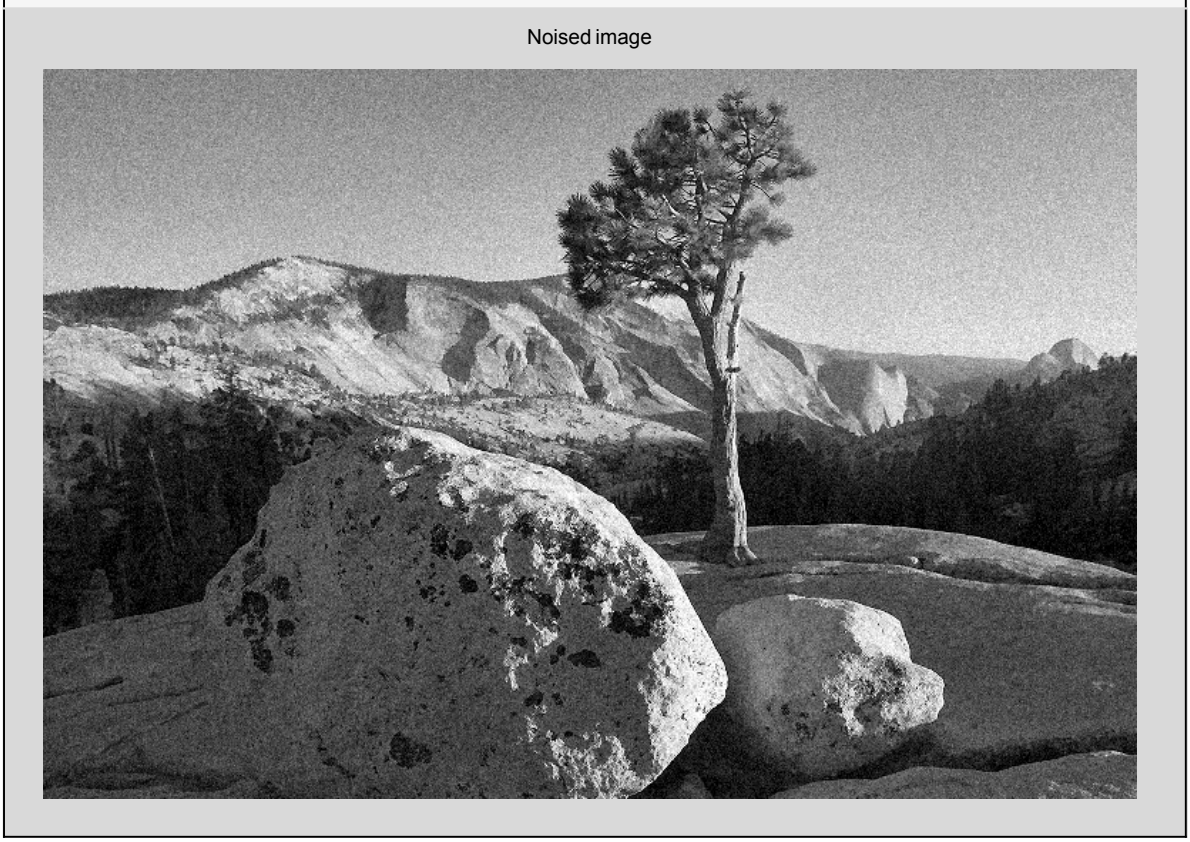
Out[62]=



A noised version of the image

In[63]:=
```
{rows, cols} = Dimensions[A];
nd = NormalDistribution[0., 1.];
SeedRandom[];
noise = Partition[Table[Random[nd], {k, 1, rows * cols}], cols];
sigma = 15.;
B = A + sigma * noise;
p3 = ImagePlot[B, PlotLabel → "Noised image"];
Show[p3, ImageSize → Full]
```

Out[70]=



Denoising using D12

Estimating the noise level using the D12 wavelet

In[71]:=
```
se = NoiseEstimate[B, Daub[12]]
```

Out[71]=
```
17.9266
```

Computing the universal threshold using the D12 wavelet

In[72]:=
```
UniversalThreshold[B, Daub[12], NumIterations → 2]
```

Out[72]=
```
90.7643
```

Computing Donoho's SURE value for the threshold parameter using D12

In[73]:=
```
wt = WT2D[B, Daub[12], NumIterations → 1];
wtlist = WaveletMatrixToList[wt, NumIterations → 1];
vertical = Flatten[wtlist[[2, 1]] / se];
horizontal = Flatten[wtlist[[2, 2]] / se];
diagonal = Flatten[wtlist[[2, 3]] / se];
λ = Map[DonohoSure, {horizontal, vertical, diagonal}]
```

Out[77]=
```
{1.40967, 1.48557, 1.44644}
```

Denoising using the universal Threhold

In[78]:=
```
its = 4;
univ = UniversalThreshold[B, N[Daub[12]], NumIterations → its];
Visu = WaveletShrinkage[B, N[Daub[12]], univ, NumIterations → its];
p4 = ImagePlot[Visu, PlotLabel → "Denoised image using VisuShrink"];
Show[p4, ImageSize → Full]
```

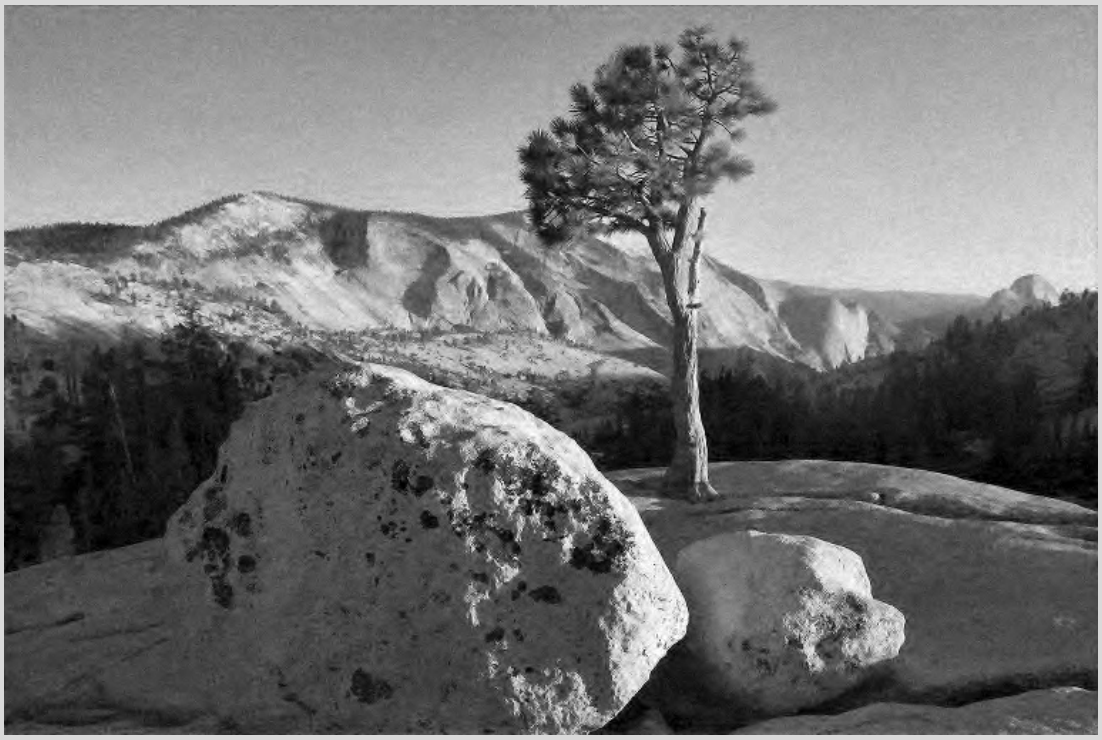
Out[82]=

Denoised image using VisuShrink

Denoising using the SURE method

In[83]:=
```
Sure = SureShrink[B, N[Daub[12]], NumIterations → its];
p5 = ImagePlot[Sure, PlotLabel → "Denoised image using SureShrink"];
Show[p5, ImageSize → Full]
```

Out[85]=



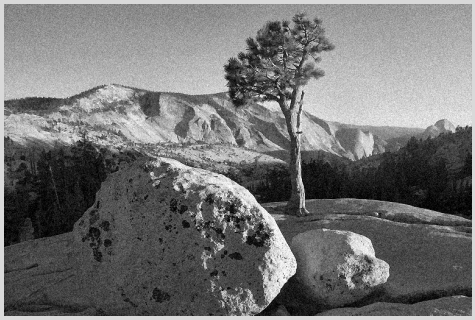
Comparing the results

In[86]:=
```
GraphicsGrid[{{p0, p3}, {p4, p5}}, Frame → All, ImageSize → Full]
```

Out[86]=

Original image

Noised image



Denoised image using VisuShrink

Denoised image using SureShrink