Compression of a Brain MRI Image with Wavelets

Introduction

This application reduces the size of a brain MRI image by removing low-level components by wavelet thresholding

```
> restart :
  with( LinearAlgebra) :
  with( ImageTools) :
  with( DiscreteTransforms) :
```

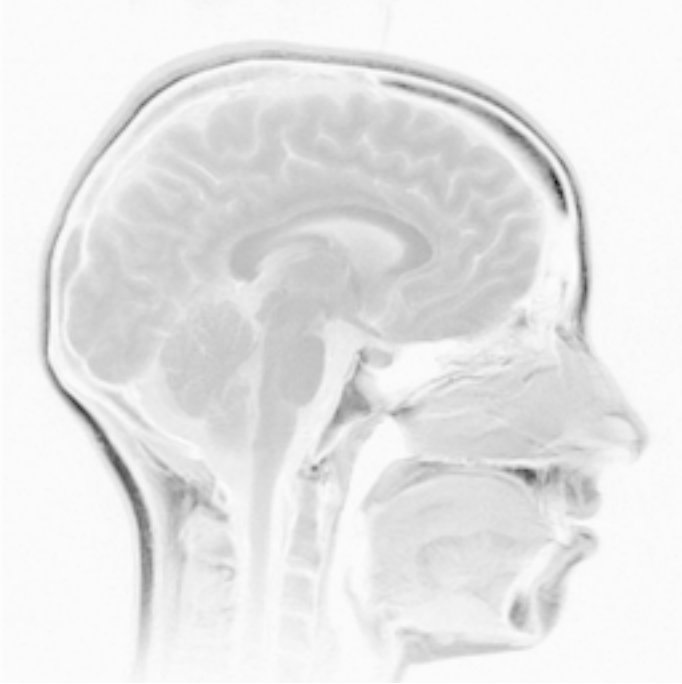
Utility Functions

```
> ImageDWT := proc( img::( Or( Matrix, Array) ), w1::Vector, w2::Vector, iters::posint)
```

```
    local rows, cols, n, final, current, temp;
  rows, cols := LinearAlgebra:-Dimensions( img );
  n := 1;
  final := Matrix( img );
  if modp( rows, 2^iters ) <> 0 or modp( cols, 2^iters ) <> 0 then
    error "img's dimensions are not divisible by %1", 2^iters
  end if;
  temp := Matrix( rtable_dims( final ), datatype = float[8] );
  for n to iters do
    DiscreteWaveletTransform( rows/2^(n-1), cols/2^(n-1), final, 1, w2, w1, temp, storagetype
      = singlearray );
    DiscreteWaveletTransform( cols/2^(n-1), rows/2^(n-1), temp, 2, w2, w1, final, storagetype
      = singlearray );
  end do;
  return final
end proc:
> InverseImageDWT := proc( img::( Or( Matrix, Array ) ), w1::Vector, w2::Vector, iters::posint )
    local rows, cols, n, final, current, temp, temp2, i, j;
  rows, cols := LinearAlgebra:-Dimensions( img );
  n := iters;
  final := Matrix( img );
  if modp( rows, 2^iters ) <> 0 or modp( cols, 2^iters ) <> 0 then
    error "img's dimensions are not divisible by %1", 2^iters
  end if;
  temp := Matrix( rtable_dims( final ), datatype = float[8] );
  for n from iters by -1 to 1 do
    InverseDiscreteWaveletTransform( cols/2^(n-1), rows/2^(n-1), final, 2, w2, w1, temp, storagetype
      = singlearray );
    InverseDiscreteWaveletTransform( rows/2^(n-1), cols/2^(n-1), temp, 1, w2, w1, final, storagetype
      = singlearray );
  end do;
  return final
end proc:
```

Import and Display Original Image

```
> img := Read("Brain_MRI.bmp") :
> Embed(img)
```

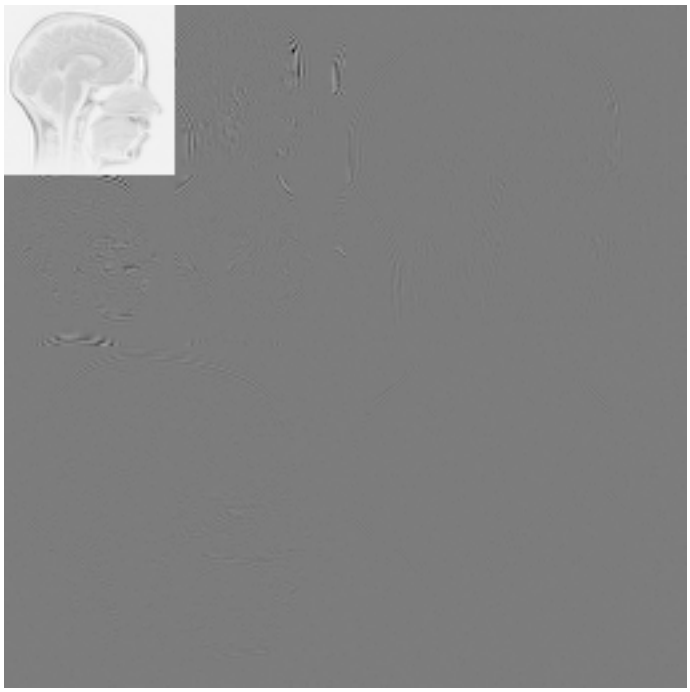


Threshold Image Via Wavelets

Now set the level of transform to be used, the percentage of zeros desired, and the wavelet to be used.

```
> imglevels := 2 :
  imgper := 90 :
  imgWLname := Daubechies :
  imgWLparam := 20 :
> imgT := ImageDWT(img, WaveletCoefficients(imgWLname, imgWLparam), imglevels) :
  imgthresh := Statistics[Percentile](ListTools:-Flatten(convert(map(abs, imgT), listlist)),
       imgper) :

  imgR := Matrix(map(z→`if`(|z| ≤ imgthresh, 0, z), imgT), datatype = float[8]) :
> Embed(FitIntensity(imgT))
```



View Compressed Image and Error

```
> imgNew := InverseImageDWT( imgR, WaveletCoefficients( imgWLname, imgWLparam ),
      imglevels ) :
> Side := ⟨img|imgNew⟩ :
> Embed( Create( Side ) )
```

```
> LinearAlgebra:-Norm( convert( imgNew − img, Matrix ), ∞ )
  ──────────────────────────────────────────────────────────;
                rtable_num_elems( img )
```

```
Er := Matrix( LinearAlgebra:-Dimensions( img ), ( i, j ) → `if`( 0.05 < |imgNew[ i, j ] − img[ i, j ]|, 1,
```

```
0) ) :
Quality( img, imgNew )
```

0.00007245819087

0.00036731225876403185 **(5.1)**

```
> Embed( Create( Er ) )
```