Background reading: <https://sinews.siam.org/Details-Page/from-global-to-local-getting-more-from-compressed-sensing>

This demo is for multi-level sampling. We will use the Shepp-Logan image (512\*512).



Step 1: Decide the sparisying system.

For this image, it is piecewise constant, so I use the haar wavelet (db1). For images of more details and textures, we could try Daubechies wavelet with higher vanishing moments. If the image could be approximated by cubic polynomials, db4 is enough.

Other choices include:

Curvelet: <http://www.curvelet.org/>

Contourlet: <https://www.mathworks.com/matlabcentral/fileexchange/8837-contourlet-toolbox>

Shearlet: <https://www3.math.tu-berlin.de/numerik/www.shearlab.org/>

Step 2: Decide the level of decomposition and the total sparsity.

Higher resolution images are usually better approximated with higher decomposition levels. Through trial and error, I find with 5 levels, the image could be approximated well by 3 percent of the total number of wavelet coefficients. The total sparsity is m = 0.03\*512\*512.

II = sparse\_approx(I,5,0.03) % change the ‘haar’ to ‘db4’, etc.



Step 3: Perform multi-level sampling with Hadamard measurements.

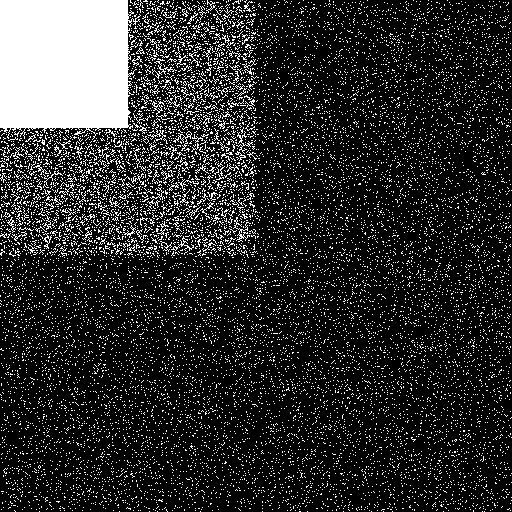
We need to decide the number of samples to achieve recovery. I just try multiples of m, M = i\*m, i = 1, …, 10. For a choice of M, the sampling scheme is then generated.

scheme = uniform\_rect\_samp\_scheme(512,512,M);

pos = random\_rect\_subsamp(scheme);

IND = sub2ind([512,512],pos(:,1),pos(:,2));

IND = sort(IND);



Step 4: Recovery from samples.

The recovery is by solving a basis pursuit denoising problem using SPGL1. I define the noise to be the difference on the sampled locations between the (re-ordered) Hadamard matrix applied on the original image and its sparse approximation.

b = fwht2(I);

b = b(IND)\*sqrt(n1\*n2);

bb = fwht2(II);

bb = bb(IND)\*sqrt(n1\*n2);

sigma = norm(bb-b);

Define the measurement matrix coupled with inverse wavelet transform to be A, together with its adjoint in the function handle.

opA = @(x,mode) partial\_Hadamard(x,S,IND,n1,n2,level,mode);

The wavelet coefficients are recovered.

C = spg\_bpdn(opA,b,sigma);

The final image is reconstructed. (with M = 6\*m)

III = waverec2(C,S,'haar');



With the same amount of samples, uniform sampling can only give us the following.

￼

Others

1. Fourier+wavelet is also implemented. However, line search error is usually encountered. I have tried another solver based on iterative hard thresholding, and it worked properly. So Fourier+wavelet is valid, it’s just SPGL1 solver is not a good choice in this case.
2. Hadamard transform can only be performed with signals of length to the power of 2. One can zero-padding to make the signal to the power of 2, I have not tried this yet. It would be helpful if someone could explore this.
3. The codes for Fourier and Hadamard sampling are available from

<https://bitbucket.org/vegarant/code-thesis>

1. Run spgsteup to compile SPGL1. A python version is also available at  
   <https://github.com/drrelyea/SPGL1_python_port>