Compressive Saliency Sensing: sampling near the edges

Scott Sievert, sieve121@umn.edu

Contents

---

1 One Dimension

In one dimension, we do not have a tree like we do in the two dimension case. Instead of having three branches like the 2D case, we have two branches at each node.

1.1 Approximating the Wavelet transform

Lets say we have an n dimensional signal: $x = (x_1, \dots, x_n)^T$. We can take the wavelet transform of this signal, w, using $x = hw$, where h is the wavelet matrix (in our case, the simple Haar matrix).

The terms in w correspond to different frequency components, and the latter terms correspond to higher frequencies. We only care about these terms if we're close to an edge, since that's where high frequency terms are. There's no need to approximate a DC or constant term with high frequency.

So let's say we're only interested in the top m terms of w, since what's specified when we approximate the wavelet. Since we only care about the upper portions of w, we can get rid of the corresponding rows for h.

$$\begin{bmatrix} h_{1,1} & h_{1,2} & h_{1,3} & h_{1,4} \\ h_{2,1} & h_{2,2} & h_{2,3} & h_{2,4} \\ h_{3,1} & h_{3,2} & h_{3,3} & h_{3,4} \\ h_{4,1} & h_{4,2} & h_{4,3} & h_{4,4} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix}$$

Since we only care about the first two entries of w ($m = 2$), we can write

$$\begin{bmatrix} h_{1,1} & h_{1,2} \\ h_{2,1} & h_{2,2} \\ h_{3,1} & h_{3,2} \\ h_{4,1} & h_{4,2} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$$

And, saying we didn't sample at index number 2, we can write

$$\begin{bmatrix} h_{1,1} & h_{1,2} \\ h_{3,1} & h_{3,2} \\ h_{4,1} & h_{4,2} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$$

Carrying out that matrix multiplication, or solving a linear system of equations, will approximate the wavelet coefficients.

1.2 Going between the wavelet and time indices

We know that $h^{-1}x = w$. Since we know that $h_{i,j} = 0$, we know that x_j won't matter for the ith element of w.

1.3 The actual reconstruction

We approximate the first $m = 2^{level}$ terms of the wavelet transform. If these coefficients are large enough ($|w_i| > \lambda$), we sample more at the indices corresponding to that wavelet location.

2 Two Dimensions

2.1 The wavelet transform with matrices

For two dimensions, we have to do the wavelet transform on each row and column recursively. In matrix language, it's $w_{level} = cxr$. To get it to higher dimensions, we can use the identity matrix and change the upper left corner.

$$w = I_c(cxr)I_r$$

We then want $mx = w$. Since we now know always know x and now know w, we can easily find m:

$$m = wx^{-1}$$

But, I just had an "aha!" moment. We don't have to find w with matrices. We can just use my handy-dandy function, `dwt2_full(x)`.

This isn't the best method. Since we're much of our image is the same, the matrix is singular (or at least close to it). We need to find another method to both approximate the transform and figure out which indices are important.

2.2 Approximating the wavelet transform

We can easily get m using `dwt2_full`. It's then the same problem as in one dimension: solve a linear system. Since you only care about the coarse wavelet coefficients, delete the rows m that give you those. Also delete the columns that you don't sample at.

2.3 Which indices are important?

We know the indices most important for the 1D Haar transform. It's essentially $w_{ind} = c_{ind}x_{ind}r_{ind}$

2.4 The reconstruction