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 Approximating the wavelet transform

We want to approximate the wavelet transform in two dimensions. How should we do that? Some ideas I went through were setting the non-sampled indices to 0 and deleting the proper (what's proper?) rows and columns in x, c and r.

Neither of those give you the right output. What you have to do is think of it at a more basic level. We're trying to approximate the 2D Haar wavelet transform. What if we just approximated the wavelet transforms of the rows then approximated the wavelet transform on {just the resulting $m \times m$ matrix, all the elements}?

Since we are *approximating* terms, that means that the last half of all of these signals are 0. So it doesn't matter whether we use all the indices or just the resulting $m \times m$ matrix – they'll give the same result.

The end goal: approximate wavelet transforms of the rows and columns. I'm guessing that we will approximate the columns then approximate the wavelet transform of that matrices rows.

2.2 Which indices are important?

Initially, I thought that the wavelet transform had to be recursive. That makes it really tricky for the indices – you would have to make `dwt_ind` functions, and keep doing it.

But after I talked to Ashkay, I learned that you can do the full wavelet transform on each row then each wavelet transform on each column. That makes this function trivial: it's just a matter of indexing.

2.3 The reconstruction

In the wavelet domain, we have a tree that corresponds to the image. An example is in Figure 1. The upper levels of the tree represent the lower frequency terms. Since there's no need to closely sample a low frequency term, we only sample where there are high frequencies.

We know that if any of these "branches" are close enough to zero ($|x| < \lambda$) that all of it's child branches are close enough to 0 as well. Therefore, we only look at where the branches are not close

enough to zero ($|x| > \lambda$). As we go further down the branch, we form a better approximation of the wavelet transform.

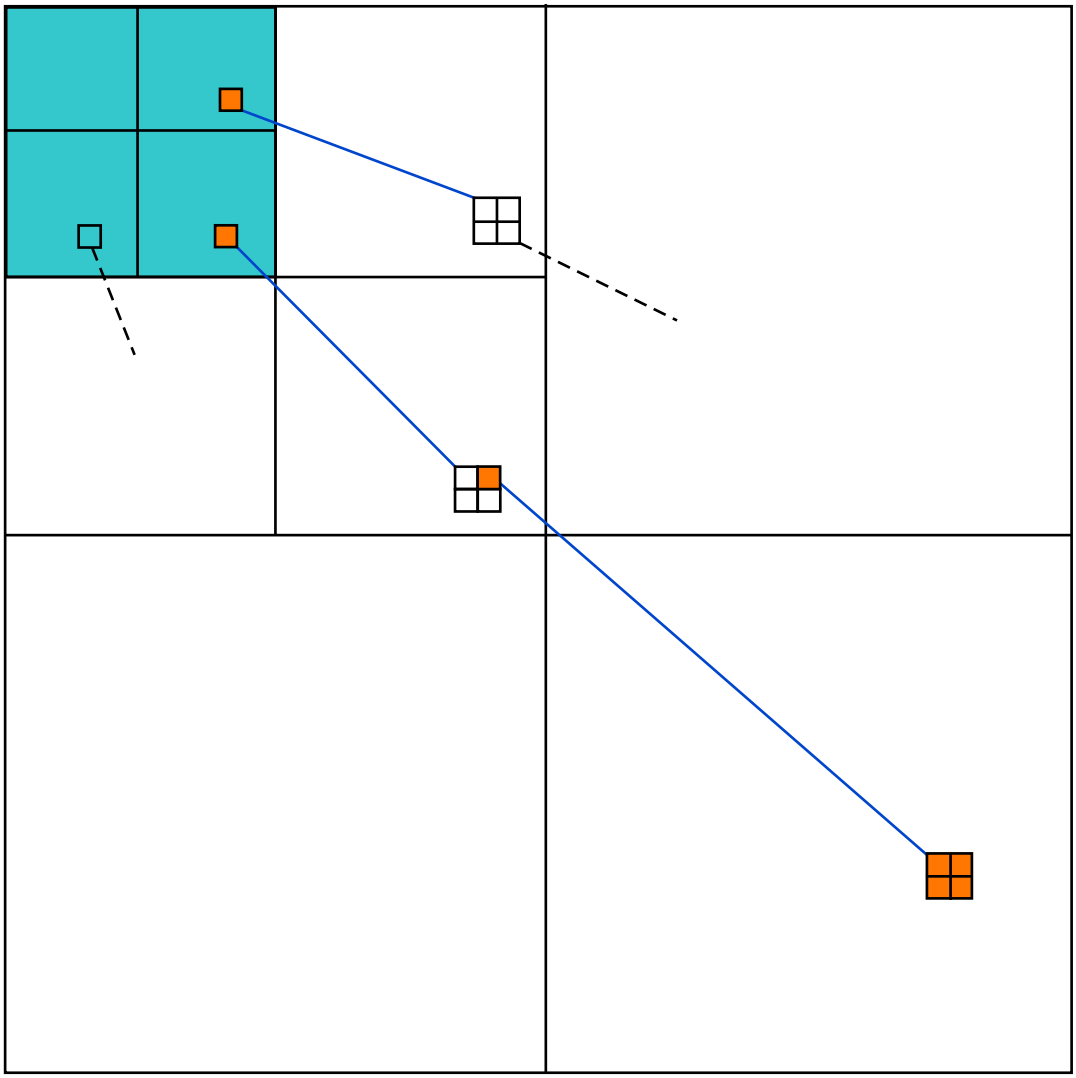

Figure 1: A visualization of our tree. The blue square is our current node. Only two of it's components are non-zero, meaning that there's more detail at lower levels we need to look at closer. We then build a better and better approximation as we need too.

Where we choose to look (or where we sample) is near the edges. The wavelet transform is zero for a constant: it has no high frequency terms.