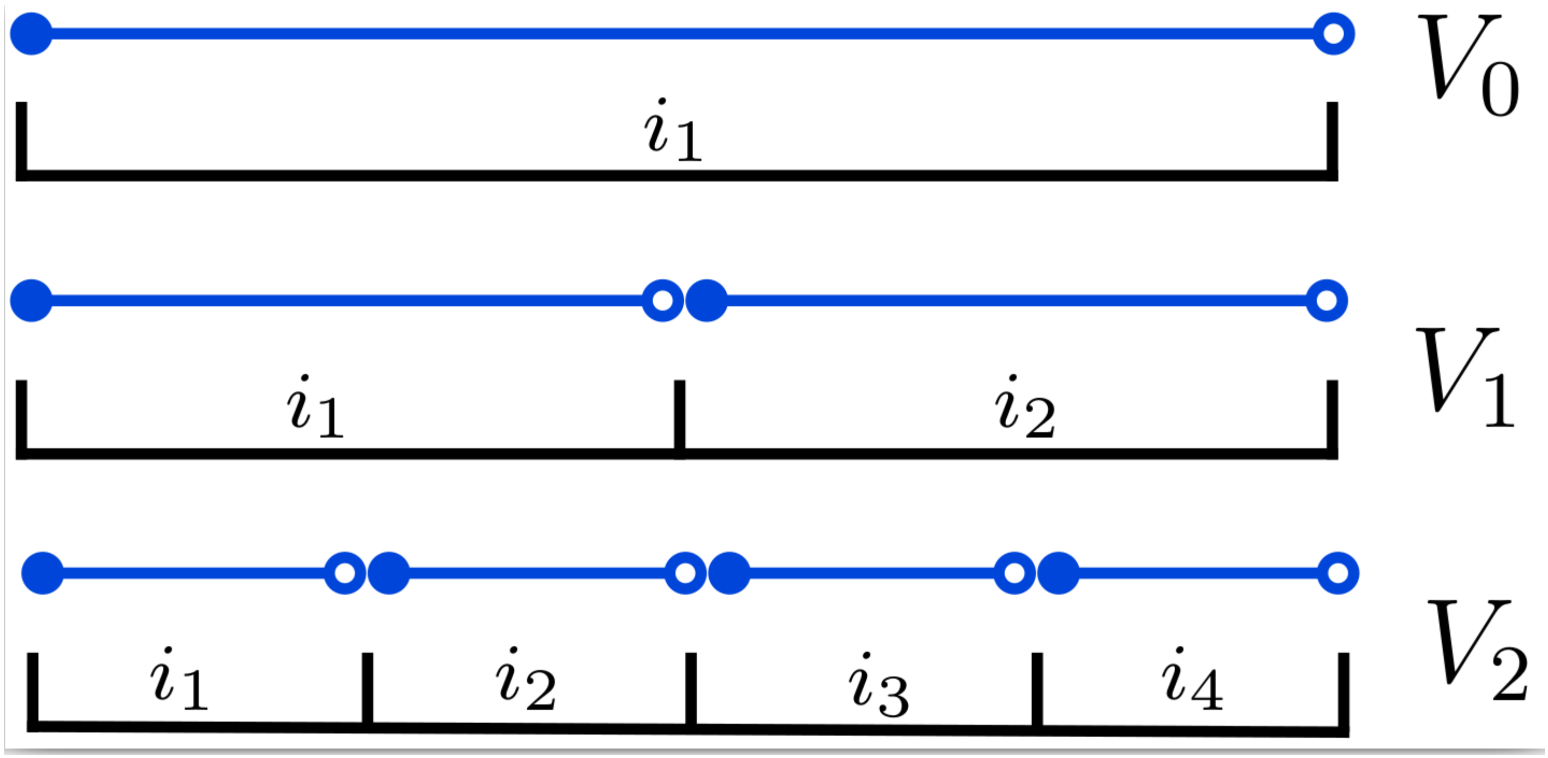
# Lecture 10 Part 1: Wavelets

- Wavelets offer a way to represent a function in a “multi-scale” way
- There are lots of different types of wavelets with different properties
- Used a lot in signal processing, audio/image/video analysis, audio/image/video compression
- The first wavelet type was created by Alfred Haar in 1909
- We are going to talk about Haar wavelets as they are the simplest type of wavelet
- There is a lot of theory behind why wavelets work, different types of wavelets, etc. Plenty to read if interested!

# Function representation

- Start with a representation of a function f on a 1D line
- f is represented by a combination of P0 scaling functions with compact support



# Haar wavelets

s, whose scaling functions are given by the constant function on each partition, or

$$\omega_{u,n}(x) = \begin{cases} 1, & \text{if } x \in i_n \\ 0, & \text{otherwise} \end{cases},$$

wavelet functions are defined as

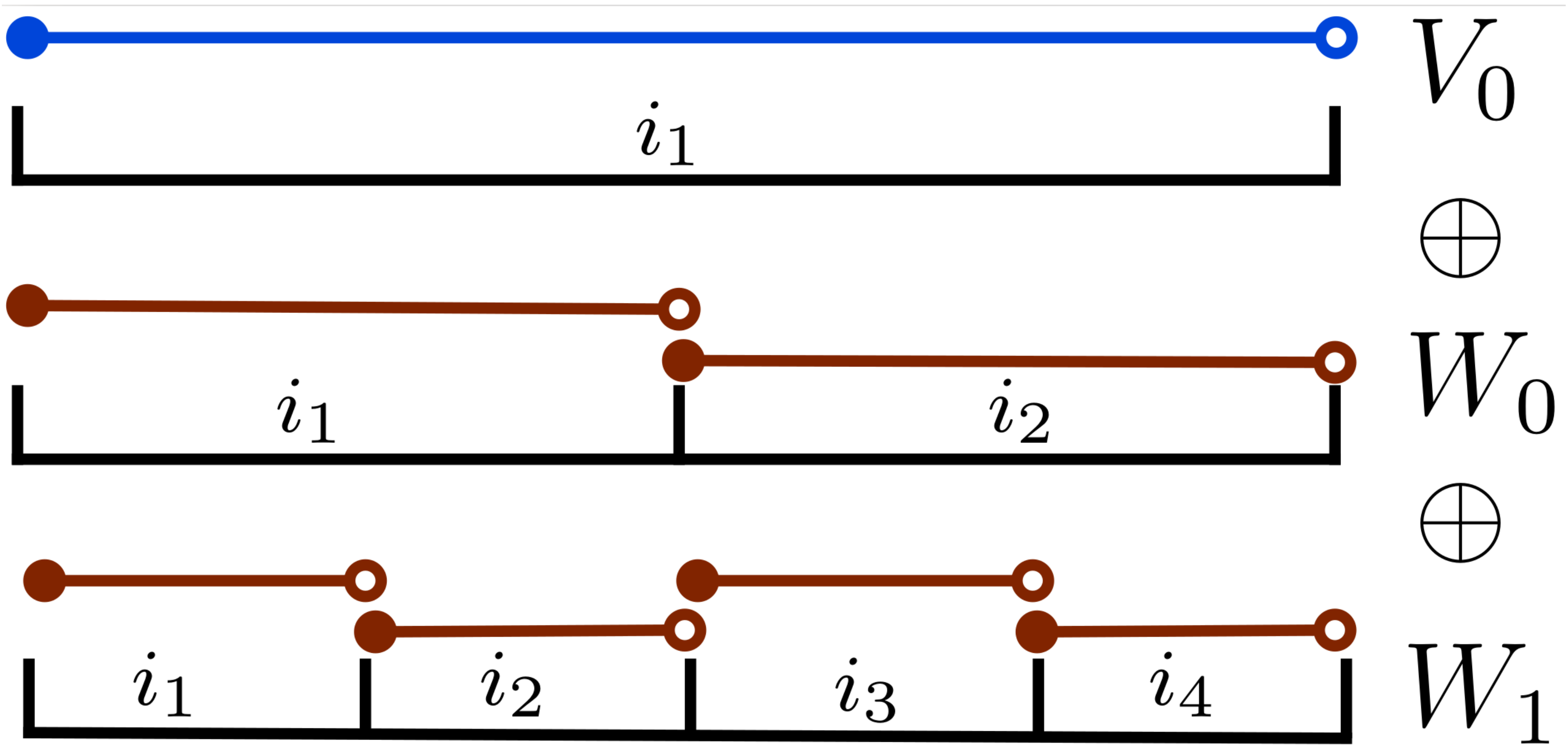
$$\tau_{m,k}(x) = \begin{cases} 1, & \text{if } x \in i_{2k-1} \\ -1, & \text{if } x \in i_{2k} \\ 0, & \text{otherwise} \end{cases}.$$

# Wavelet expansion

- A combination of scaling functions and wavelet functions can give us an expression for our original function
- Our representation of f on level j , with k wavelet functions on a level m, with n scaling functions on a base level V_u is given by:

$$f \approx f_j = \sum_{n} \alpha_{u,n} \omega_{u,n} + \sum_{m=u}^{j-1} \sum_{k} \beta_{m,k} \tau_{m,k},$$

# Haar wavelet function representation



# Why wavelets?

- Our original P0 representation and our wavelet representation are exactly equivalent
- Why chose a wavelet basis then?
- Wavelets are hierarchical
- As long as we have the scaling functions which cover the whole domain, we can include whichever wavelet functions we like and get a representation of our function with different amounts of “detail”
- No interpolation needed to move between the different levels of detail

# Performing our wavelet transform?

- How do we move from our P0 representation to a wavelet representation?
- Super simple with our Haar wavelets
- Pick the finest P0 representation you want
- Then create a “multi-scale” P0 representation
- You can compute wavelet coefficients directly on each level
- This has a tree-like structure
- Can be performed in O(n) time!
- See:

https://en.wikipedia.org/wiki/Fast_wavelet_transform

# Time to write some code!

- Create main_test_haar.cpp

# Wavelet compression

- How do wavelets lend themselves to compression?
- One feature is very important
- The size of the wavelet coefficient relates how “important” it is to representing our function
- So if you throw away all your “small” wavelet coefficients (up to some tolerance), you get the a good representation of your function with the fewest wavelet coefficients
- This is how wavelet compression schemes work for audio/video
- Very important on the internet for streaming services!

# Homework

- Rewrite main_test_haar.cpp to compute the Haar coefficients at the same time the P0 values on each level
- Rewrite main_test_haar.cpp to use contiguous memory across all the levels (ie the memory for each level is not split across different arrays, just use one array)
- Rewrite main_test_haar.cpp changing the ordering of how you access the level_data to be depth-first, not level first
- ADVANCED – Read:

http://www.ws.binghamton.edu/fowler/fowler%20personal%20page/EE523_files/Ch_15_4%20EZW%20(PPT)%20revised.pdf

# Homework

- SUPER ADVANCED – Implement an EZW-like scheme by picking a tolerance value, sort all the wavelet coefficients which are greater than this tolerance. Then halve the tolerance, sort the remaining wavelets, and continue this halving/sorting. This is an optimal encoding (with the most detail) given you could cut this bit stream at any point.
- ULTRA ADVANCED – Implement a 2D wavelet transform and apply it to the fractal image filters introduced in Lecture 4