Recap

Haar
- simple and fast wavelet transform

Limitations
- not smooth enough: blocky

How to improve?
- classical approach: basis functions
- Lifting: transforms

Erasing Haar Coefficients
Classical Constructions

Fourier analysis
- regular samples, infinite setting
- analysis of polynomials

Conditions:
- smoothness
- perfect reconstruction

But...
- Fourier analysis not always applicable

Lifting Scheme

Custom design construction
- entirely in spatial domain

Second generation wavelets
- boundaries
- irregular samples
- curves, surfaces, volumes
Haar Transform

Averages and differences

- two neighboring samples

\[ a = s - d/2 \]
\[ b = s + d/2 \]

\[ d = b - a \]
\[ s = (a + b)/2 \]

---

Haar Transform

In-place Version

- want to overwrite old values with new values
- rewrite

\[ d = b - a \]
\[ s = a + d/2 \]

\[ b -= a; \quad a += b/2; \]

- inverse: run code backwards!

\[ a -= b/2; \quad b += a; \]
Haar Transform

Forward

```
for( s = 2; s <= n; s *= 2 )
    for( k = 0; k < n; k += s ){
        c[k+s/2] -= c[k];
        c[k] += c[k+s/2] / 2;
    }
```
**Haar Transform**

**Lifting version**

- split into even and odd
  \[(\text{even}_{j-1}, \text{odd}_{j-1}) := \text{Split}(s_j)\]

- predict and store difference: detail coefficient
  \[d_{j-1} = \text{odd}_{j-1} - \text{even}_{j-1}\]

- update even with detail: smooth coefficient
  \[s_{j-1} = \text{even}_{j-1} + \frac{d_{j-1}}{2}\]

---

**Haar Transform**

\[d_{j-1} = \text{odd}_{j-1} - P(\text{even}_{j-1})\]

\[s_{j-1} = \text{even}_{j-1} + U(d_{j-1})\]
Haar Transform

Predict
- perfect if function is constant
- detail coefficients zero
- removes constant correlation

Update
- preserve averages of coarser versions
- avoid aliasing
- obtain frequency localization

Haar Transform

![Diagram of Haar Transform]

- $s_j$ split
- even $\rightarrow$ smooth $s_{j-1}$
- odd $\rightarrow$ detail $d_{j-1}$
- $P$, $U$
Lifting Scheme

Advantages
- in-place computation
- efficient, general
- parallelism exposed
- easy to invert

Lifting

Build more powerful versions
- higher order prediction
  - Haar has order 1
- higher order update
  - preserve more moments of coarser data

An example
- linear wavelet transform
Linear Prediction

Use even on either side
- keep difference with prediction
- exploit more coherence/smoothness/correlation

\[ d_k = a_{2k+1} - \frac{1}{2} (a_{2k} + a_{2k+2}) \]

Prediction
Update

Even values are subsampled

- aliasing!

DC components different
average different

---

Update

\[
s_k = a_{2k} + \frac{1}{4}(d_{k-1} + d_k)
\]

zero mean

add

even

detail

smooth
Inplace Wavelet Transform

Linear Wavelet Transform

Order
- linear accuracy: 2nd order
- linear moments preserved: 2nd order
- \((2,2)\) of Cohen-Daubechies-Feauveau

Extend
- build higher polynomial order predictors
Higher Order Prediction

Use more \((D)\) neighbors on left and right
- define interpolating polynomial of order \(N=2D\)
- sample at midpoint for prediction value
- example: \(D=2\)

effective weights:

\[-1/16\ 9/16\ 9/16\ -1/16\]
Summary

Lifting Scheme
- construction of transforms
- spatial, Fourier

Haar example
- rewriting Haar in place

Two steps
- Predict
- Update

Summary

Predict
- detail coefficient is failure of prediction

Update
- smooth coefficient to preserve moments, e.g., average

Higher order extensions
- increase order of prediction and update
Building Blocks

Transform
- forward
  \[ W\{s_{n,k}\} = \{d_{j,l}\} \]
- inverse
  \[ \{s_{n,k}\} = W^{-1}\{d_{j,l}\} \]
- superposition
  \[ \{s_{n,k}\} = \sum d_{j,l} W^{-1}\{\delta_{j,l}\} \]

Scaling Functions

Cascade/Subdivision
- single smooth coefficient

Diagram:
- Delta sequence
- U, P, Merge
- 0
Scaling Functions

Cascade/Subdivision

Scalae Functions

$N = 2$  $N = 4$

$N = 6$  $N = 8$
**Twoscale Relation**

**Scaling function**

\[ \phi(x) = \sum h_l \phi(2x - l) \]

**Duality**

**Function at 2 successive scales**

\[ \sum_{k} s_{j,k} \phi_{j,k}(x) = f(x) = \sum_{l} s_{j+1,l} \phi_{j+1,l}(x) \]

column vectors of coefficients

row vectors of bases
Interpolating Scaling Functions

Properties for order N=2D

- compact support:
  \[ \phi(x) = 0 \quad x \not\in [-N+1, N-1] \]

- interpolation:
  \[ \phi(k) = \delta_k \]

- polynomial reproduction:
  \[ \sum_k k^p \phi(x-k) = x^p \]

---

Interpolating Scaling Functions

Properties for order N=2D

- smoothness:
  \[ \phi_{j,k} \in C^{\alpha(N)} \]

- twoscale relation:
  \[ \phi(x) = \sum_{l=-N}^{N} h_l \phi(2x-l) \]

\[ s_{j+1,l} = \sum_k h_{l-2k} s_{j,k} \quad \phi_{j,k}(x) = \sum_l h_{l-2k} \phi_{j+1,l}(x) \]
Wavelets

Cascade/Subdivision

- single detail coefficient

Wavelets

\[ N = \text{merge} \]

\[ N = \text{delta sequence} \]

\[ N = 6 \]

\[ N = 8 \]
Twoscale Relation

Wavelet

\[ \psi(x) = \sum g_l \varphi(2x - l) \]

\[
\begin{bmatrix}
0 & 0 & 1 & 0 & 0
\end{bmatrix}
\]

\[ \{g_l\} \]

\[ 0 \ 0 \ -1/8 \ -1/4 \ -1/8 \ 0 \ 0 \]

cascade
Average Interpolation

constant  quadratic
**Average Interpolation**

**Idea**
- Assume observed samples are averages
- Which polynomial would have produced those averages?

\[
\text{observation: } \ldots S_{j,k-1} \quad S_{j,k} \quad S_{j,k+1} \ldots
\]

\[
\text{match: } p(x)
\]

\[
\text{finer averages: } S_{j+1,2k} \quad S_{j+1,2k+1}
\]
Scaling Functions

Average Interpolating Scaling Functions

Properties for order N=2D+1

- compact support:
  \[ \varphi(x) = 0 \quad x \notin [-N+1,N] \]

- average interpolation:
  \[ \int_{-k}^{k+1} \varphi(x) \, dx = \delta_k \]

- polynomial reproduction:
  \[ \sum_k \text{Ave}(x^p,k) \varphi(x-k) = x^p \]
Average Interpolating Scaling Functions

Properties for order $N=2D+1$

- smoothness:
  $$\varphi_{j,k} \in C^{\alpha(N)}$$

- twoscale relation:
  $$\varphi(x) = \sum_{l=-N+1}^{N} h_l \varphi(2x-l)$$
  $$\varphi_{j,k}(x) = \sum_{l} h_{l-2k} \varphi_{j+1,l}(x)$$
  $$s_{j+1,l} = \sum_{k} h_{l-2k} s_{j,k}$$

Wavelets

- $N = 1$
- $N = 3$
- $N = 5$
- $N = 7$
Differentiation

Interpolation and average interpolation

- Given interpolation sequence compute exact derivative

\[
\{s_{0,k}\} \quad \quad \quad \quad \quad \quad \quad \quad \quad N = 2D
\]
\[
\{s_{0,k}' = s_{0,k+1} - s_{0,k}\} \quad \quad \quad \quad \quad N' = 2D - 1
\]

\[
\frac{d}{dx}\phi_I(x) = \phi^{AI}(x+1) - \phi^{AI}(x)
\]
Cubic B-splines

Subdivision
1. generate \{1, 4, 6, 4, 1\}

\[
\begin{align*}
s_{j+1,2k+1} &= \left( s_{j,k} + s_{j,k+1} \right) / 2 \\
s_{j+1,2k} &= s_{j,k} + \left( s_{j+1,2k-1} + s_{j,2k+1} \right) / 2
\end{align*}
\]

Cubic B-spline Wavelet

Completing the space
1. put delta on detail wire: \{1, 4, 1\}

2. get vanishing moment with update stage: \{3/8, 3/8\}
Cubic B-spline

Scaling function

Wavelet