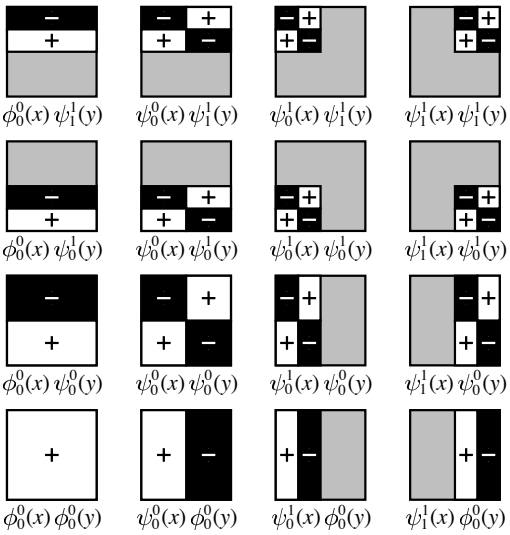

Figure 7 Standard construction of a two-dimensional Haar wavelet basis for V^2. In the unnormalized case, functions are +1 where plus signs appear, −1 where minus signs appear, and 0 in gray regions.

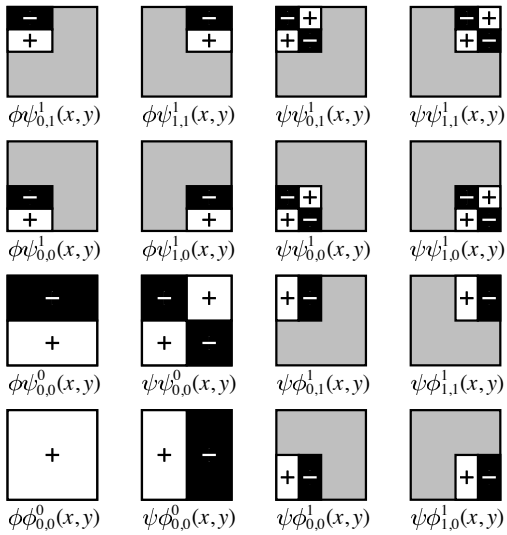

Figure 8 Nonstandard construction of a two-dimensional Haar wavelet basis for V^2.

while some standard basis functions have nonsquare supports. Depending upon the application, one of these choices may be preferable to the other.

3.3 Application II: Image compression

We defined compression in Section 2.3 as the representation of a function using fewer basis function coefficients than were originally given. The method we discussed for one-dimensional functions applies equally well to images, which we treat as the coefficients corresponding to a two-dimensional piecewise-constant basis. The approach presented here is only introductory; for a more complete treatment of wavelet image compression, see the article by DeVore *et al*. [6].

We can summarize wavelet image compression using the L^2 norm in three steps:

1. Compute coefficients $c_1, \ldots, c_m$ representing an image in a normalized two-dimensional Haar basis.
2. Sort the coefficients in order of decreasing magnitude to produce the sequence $c_{\sigma(1)}, \ldots, c_{\sigma(m)}$.
3. Starting with $\tilde{m} = m$, find the smallest $\tilde{m}$ for which $\sum_{i=\tilde{m}+1}^{m} (c_{\sigma(i)})^2 \leq \epsilon^2$, where ϵ is the allowable L^2 error.

The first step is accomplished by applying either of the two-dimensional Haar wavelet transforms described in Section 3.1, being sure to use normalized basis functions. Any standard sorting technique will work for the second step. However, for large images sorting becomes exceedingly slow.

The pseudocode below outlines a more efficient method that uses a binary search strategy to find a threshold below which coefficient sizes are deemed negligible. The procedure takes as input a one-dimensional array of coefficients C (with each coefficient corresponding to a two-dimensional basis function) and an error tolerance ϵ. For each guess at a threshold τ, the algorithm computes the square of the L^2 error that would result from discarding coefficients smaller in magnitude than τ. This squared error s is compared to ϵ^2 at each iteration to decide whether the binary search should continue in the upper or lower half of the current interval. The algorithm halts when the current interval is so narrow that the number of coefficients to be discarded no longer changes.

```
procedure Compress(C: array [1..m] of reals; ε: real)
    τmin ← min{ |C[i]| }
    τmax ← max{ |C[i]| }
    do
        τ ← (τmin + τmax)/2
        s ← 0
        for i ← 1 to m do
            if |C[i]| < τ then s ← s + (C[i])^2
        end for
        if s < ε^2 then τmin ← τ else τmax ← τ
    until τmin ≈ τmax
    for i ← 1 to m do
        if |C[i]| < τ then C[i] ← 0
    end for
end procedure
```

This binary search algorithm was used to produce the images in Figure 9. These images demonstrate the high compression ratios wavelets offer, as well as some of the artifacts they introduce.

DeVore *et al*. [6] suggest that the L^1 norm is best suited to the task of image compression. Here is a pseudocode fragment for a "greedy" L^1 compression scheme:

```
for each pixel (x, y) do
    δ[x, y] ← 0
end for
for i ← 1 to m do
    δ' ← δ + error from discarding C[i]
    if Σ_{x,y} |δ'[x, y]| < ε then
        discard coefficient C[i]
        δ ← δ'
    end if
end for
```

Note that this algorithm's results depend on the order in which coefficients are visited. Different images (and degrees of compression) may be obtained from varying this order—for example, by starting with the finest scale coefficients, rather than the smallest coefficients. You could also run a more sophisticated constrained optimization procedure to select the minimum number of coefficients subject to the error bound.