CS 522: Digital Image Processing
Homework 5 (Spring '07)

1 Theory

1. The n-th moment of Ψ is defined to be $M_n\{\Psi\} = \int_{-\infty}^{\infty} t^n \Psi(t) dt$. Let $f(t) = e^{-\pi t^2}$, $f'(t) = -2\pi t e^{-\pi t^2}$, and $f''(t) = 2\pi e^{-\pi t^2}(2\pi t^2 - 1)$. Prove the following:

   (a) $M_0\{f'\} = 0$.

   (b) $M_0\{f''\} = M_1\{f''\} = 0$.

2. The six vectors, $\mathbf{f}_1 = \begin{bmatrix} \cos(\pi/3) & \sin(\pi/3) \end{bmatrix}^{\mathrm{T}}$, $\mathbf{f}_2 = \begin{bmatrix} \cos(\pi/3) & -\sin(\pi/3) \end{bmatrix}^{\mathrm{T}}$, $\mathbf{f}_3 = \begin{bmatrix} -1 & 0 \end{bmatrix}^{\mathrm{T}}$, $\mathbf{f}_4 = \begin{bmatrix} -\cos(\pi/3) & -\sin(\pi/3) \end{bmatrix}^{\mathrm{T}}$, $\mathbf{f}_5 = \begin{bmatrix} -\cos(\pi/3) & \sin(\pi/3) \end{bmatrix}^{\mathrm{T}}$, and $\mathbf{f}_6 = \begin{bmatrix} 1 & 0 \end{bmatrix}^{\mathrm{T}}$ form a frame $\mathcal{F}$ for $\mathbb{R}^2$. Draw the frame.

   (a) Give two representations for the vector, $\mathbf{x} = \begin{bmatrix} 1 & 1 \end{bmatrix}^{\mathrm{T}}$, in $\mathcal{F}$.

   (b) Prove that $\mathbf{x}$ has an infinite number of representations in $\mathcal{F}$.

   (c) Give a matrix which transforms any representation of a vector in $\mathcal{F}$ into its representation in the standard basis for $\mathbb{R}^2$.

   (d) Give a matrix which transforms a representation of any vector in the standard basis for $\mathbb{R}^2$ into its representation in $\mathcal{F}$.

3. The continuous representation of the Haar highpass filter is $h_1(t) = \frac{1}{2}[\delta(t + \Delta t) - \delta(t - \Delta t)]$. The continuous representation of the Haar lowpass filter is $h_0(t) = \frac{1}{2}[\delta(t + \Delta t) + \delta(t - \Delta t)]$. Prove that $H_0(s)H_0^*(s) + H_1(s)H_1^*(s) = 1$ where $H_0(s)$ and $H_1(s)$ are the Fourier transforms of $h_0(t)$ and $h_1(t)$.

4. Compute the Haar transform of the vector $\begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix}^{\mathrm{T}}$.

2 Practice

1. Write a function *daubechies4* which takes a square image, *im*, of size 2^k for integer k as input, and returns a list of length four representing the two-dimensional $x - y$ separable Daubechies 4 wavelet transform of *im*. The

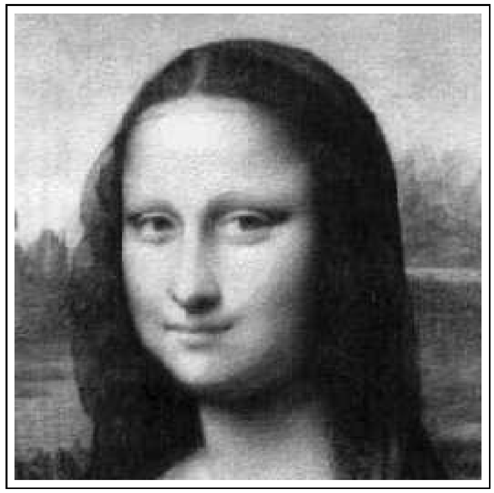

Figure 1: Mona Lisa.

last three elements of the list are the level 1 wavelet subbands and the first element is (itself) a list of length four (recursively) representing levels 2 through k of the wavelet transform.

2. Write a function *inverse-daubechies4* which takes a list of length four representing a two-dimensional $x-y$ separable Daubechies 4 wavelet transform of a square image, *im*, of size 2^k for integer k as input, and returns the reconstructed image. Demonstrate your function's ability to invert a wavelet transform you compute with *daubechies4* for an image of your choice.

3. Write a function *display-wavelet-transform* which takes a list of length four representing a two-dimensional $x-y$ separable Daubechies 4 wavelet transform of a square image, *im*, of size 2^k for integer k as input, and returns an image depicting the wavelet transform using the recursive scheme shown in Figure 2. Demonstrate your function on an image of your choice. Note: The images representing the wavelet subbands must each be normalized to the range [0-255] with grey level 0 mapped to grey level 128 prior to constructing the display.

4. Write a function *denoise-color-image* which takes a color image, *cim*, as input and returns a denoised color-image computed by:

    - Converting *cim* to HSI.

(a) (b)

Figure 2: (a) Bill Clinton. (b) Recursively displayed two-dimensional $x-y$ separable Daubechies 4 wavelet transform.

- Computing the Daubechies 4 wavelet transform of the intensity component.
- Soft-thresholding the S and I wavelet subbands.
- Computing the inverse Daubechies 4 wavelet transform.
- Converting the HSI representation back to RGB.

5. Find a noisy color image on the internet, *i.e.*, an image which has been degraded by aliasing from downsampling or contains visible JPEG blocking, film grain, or other additive noise. If you cannot find a suitable image, then start with a high quality color image and degrade it yourself, *e.g.*, using *xv*.

6. Use *denoise-color-image* to denoise your image. Use a threshold for shrinkage which you judge to be optimum and one which is too large. Show your results for both thresholds.