CS 787: Assignment 2, Linear Systems and Feature Detection
Due: 3:30pm, Mon. Feb. 26, 2007.

1. Fourier analysis and wavelets [5 marks]

"Gabor filters" are created by multiplying a sinusoidal grating times a Gaussian window:

$$\text{Gabor}(\mathbf{x}; \mathbf{k}, \sigma) = e^{i\mathbf{k}\cdot\mathbf{x}} e^{-(x^2+y^2)/(2\sigma^2)}$$

where $\mathbf{x} = [x, y]^T$ is the pixel ($(0, 0)$ at the center of the filter), $\mathbf{k} = [k_x, k_y]^T$ is the spatial frequency and orientation of the filter (in cycles/pixel), and σ is the filter standard deviation (in pixels).

The complex output of the filter is typically divided into the real and imaginary components, called the *sine gabor* and *cosine gabor*, respectively.

$$\Re\left[\text{Gabor}(\mathbf{x}; \mathbf{k}, \sigma\right] = cos(\mathbf{k}\cdot\mathbf{x}) e^{-(x^2+y^2)/(2\sigma^2)}$$

$$\Im\left[\text{Gabor}(\mathbf{x}; \mathbf{k}, \sigma\right] = sin(\mathbf{k}\cdot\mathbf{x}) e^{-(x^2+y^2)/(2\sigma^2)}$$

(a) Modulation theorem [3 marks]

The *modulation theorem* states that if $f(t)$ has Fourier transform $F(f)$, then $f(t)\cos 2\pi f_0 t$ has Fourier transform $\frac{1}{2}F(f - f_0) + \frac{1}{2}F(f + f_0)$.

Prove this theorem. (Hint: use the relation $\cos ax = \frac{e^{iax}+e^{-iax}}{2}$).

One application of the modulation theorem is to show that if $f(t) = e^{-\pi t^2}\cos 2\pi f_0 t$ then its Fourier transform is $F(f) = \frac{1}{2}\left[e^{-\pi(f-f_0)^2} + e^{-\pi(f+f_0)^2}\right]$ ie., the power spectrum of a Gabor filter has a Gaussian distribution.

(b) Space-frequency localization of "Gabor" filters [2 marks]

Consider the 2D Fourier transform:

$$F(u, v) = \int_{\infty}^{\infty}\int_{\infty}^{\infty} f(x, y)e^{-j2\pi(ux+vy)}dxdy$$

$$f(x, y) = \int_{\infty}^{\infty}\int_{\infty}^{\infty} F(u, v)e^{j2\pi(ux+vy)}dudv$$

where $f(x,y)$ is the image and $F(u,v)$ is its spectrum.

The *Similarity theorem* shows that $f(ax,by)$ has Fourier transform $\frac{1}{|ab|}F\left(\frac{u}{a},\frac{v}{b}\right)$. In words: if we compress a function in the spatial domain, we expand it in the frequency domain.

The similarity theorem can be generalized to show that $f(x\cos\theta+y\sin\theta, -x\sin\theta+y\cos\theta)$ has Fourier transform $F(u\cos\theta+v\sin\theta, -u\sin\theta+v\cos\theta)$. Ie., rotating the function in the spatial domain will rotate the function by the same amount in the frequency domain.

Use the demonstration program `gabor-demo.m` to show both of these effects. (Hand in printouts from your experiments.)

Note: To do this problem you need to download the CS787 code, decompress it into the `cs787/` directory, and start matlab from the `cs787/matlab/` directory. *You are welcome to experiment with the other code, to learn about linear systems, filtering, and image pyramids.*

2. Linear Systems and Edge Detection [8 marks]

Here you will implement the edge strength detector described in *Trucco and Verri*, Sec. 4.2.2.

The edge strength at pixel (i,j) is computed as

$$s(i,j) = \| \nabla(G \otimes I) \|$$

where $\nabla f \equiv [\partial f/\partial x, \partial f/\partial y]^T$ is the derivative operator, $I(x,y)$ is the image, and G is a Gaussian kernel

$$G(x,y;\sigma) = \frac{1}{2\pi\sigma} \exp\left(-\frac{1}{2\sigma^2}(x^2+y^2)\right)$$

Due to the associativity of linear operations (differentiation and convolution), we can rewrite the above as:

$$\nabla(G \otimes I) = (\nabla G) \otimes I$$

where ∇G is the derivative of the Gaussian kernel.

Note that $\nabla G = \left[\frac{\partial G}{\partial x}, \frac{\partial G}{\partial y}\right]^T$ denotes *two* filter kernels, one for each derivative. Let's denote these kernels, $G_x(x,y)$ and $G_y(x,y)$ to represent differentiation by x and y respectively.

(a) Calculating the filter masks [2 marks]

Plot the (2D) filters $G_x(x,y)$ and $G_y(x,y)$ for $\sigma = 2$ and for $-10 \le x \le 10$, $-10 \le y \le 10$. You can use a *mesh*, *contour*, or a *greyscale* plot to display your filters. **Note:** Use the following five point, central difference operator to compute the derivatives:

$$f_i' = \frac{-f_{i+2} + 8f_{i+1} - 8f_{i-1} + f_{i-2}}{12h} + O(h^4)$$

where $h = 1$.

(b) Computing edge strength $s(i, j)$ [4 marks]

Compute the edge strength $s(i, j)$ for the image `einstein.tif`. You may use the `conv2` command in Matlab for 2D convolution with $G_x(x, y)$ and $G_y(x, y)$.

(c) Linear filtering operations [2 marks]

Explain the advantage (if any) of computing $(\nabla G) \otimes I$ vs. $\nabla(G \otimes I)$.

(d) Separable filters [2 *bonus* marks]

The operation in **(b)** can be achieved by successive application of two 1D filters. Write (but don't program) an expression for these convolutions.

3. Corner Detection [10 marks]

As described in *Trucco and Verri*, Sec. 4.3, we can detect corners by looking at the following matrix:

$$C = \left[\begin{array}{cc} \sum E_x^2 & \sum E_x E_y \\ \sum E_x E_y & \sum E_y^2 \end{array} \right]$$

where E_x is the image derivative in the x direction, E_y is the derivative in the y direction, and the summation is taken over some small neighborhood, $-N/2 \ldots N/2$. Here we will use a *7-by-7* patch.

For a symmetric matrix, C, we can write $C = V \Sigma V^T$, where V is an orthonormal matrix and Σ is a diagonal matrix

$$\Sigma = \left[\begin{array}{cc} \sigma_1 & 0 \\ 0 & \sigma_2 \end{array} \right]$$

with $\sigma_1 \geq \sigma_2$.

There are three cases to consider:

1. $\sigma_1 \geq \sigma_2 \gg 0$ In this case there is texture in any direction of the patch.
2. $\sigma_1 \gg \sigma_2 \approx 0$ In this case there is texture along only one direction of the patch (eg., an object with bands or ridges)
3. $\sigma_1 \approx \sigma_2 \approx 0$ In this case there is no texture in the patch (eg., a smooth surface).

Let's define "corners" as any place in the image where there is sufficient structure to generate nonzero derivatives in both E_x and E_y (case 1 above).

A simple algorithm to find corners is as follows:

1. Apply the image derivative operators at every pixel in the image. You should use the five-point central-difference operator from Question #2 above. Note: you do not have to smooth the image as in Question #2, just compute the derivatives in the horizontal and vertical direction.
2. Collect sums of derivatives for an *N-by-N* image patch centered on every pixel. (Note: derivatives only need to be calculated once for each pixel.)
3. Take the singular value decomposition (`svd` in Matlab) for every patch.
4. Choose the matrix with the largest σ_2 and label this as a corner point.
5. Remove any points that are within a $2N$ neighborhood of this corner (to avoid near-duplicate corners).
6. Repeat until either σ_2 becomes too small, or enough corner points are gathered.

(a) Finding corners [10 marks]

Use the method described above to find the first 50 corners in the image `microserf.tif`. Use an *7-by-7* image patch for your computation. Show the position of the corners by overlaying markers on the input image.