CS 787: Assignment 1, Lighting, Fourier analysis, Feature detection
Due: 5pm, Wed. Oct. 13, 2004.

1. Lightness and shading [10 marks]

(a) Lambertian blocks [3 marks]

Suppose we observe a trihedral junction (ie., a vertex where three faces meet) of a rectangular block, with known orientation wrt the camera. If the block has a Lambertian surface of constant albedo, we can determine the direction of a single (distant) light source from the relative brightness of the three faces of the block.

If the orientation of each face in the camera's reference frame is given by unit normal vectors $\hat{\mathbf{n}}_1$, $\hat{\mathbf{n}}_2$, and $\hat{\mathbf{n}}_3$, respectively, show how to obtain the lighting direction and albedo of the surface.

Note: You can ignore the $\cos^4\theta$ term in the imaging equation and that we are using orthographic projection.

(b) Derivation of photometric stereo [2 marks]

(Adapted from Horn, Exercise 10-16) Suppose we have three images of a Lambertian surface under three different light source directions.

The three images E_1, E_2, and E_3 are given by the following equations,

$$E_1 = \rho(\hat{\mathbf{s}}_1 \cdot \hat{\mathbf{n}}), \quad E_2 = \rho(\hat{\mathbf{s}}_2 \cdot \hat{\mathbf{n}}), \quad E_3 = \rho(\hat{\mathbf{s}}_3 \cdot \hat{\mathbf{n}})$$

where ρ is the surface albedo (may vary with position (x, y)), $\hat{\mathbf{n}}$ is the surface normal (unit vector), and

$$\hat{\mathbf{s}}_i = \frac{(-p_i, -q_i, 1)^T}{\sqrt{1 + p_i^2 + q_i^2}}$$

for $i = 1, 2, 3$ give the directions of the light source.

By subtracting pairs of equations, show that:

$$\rho\hat{\mathbf{n}} \cdot (E_1\hat{\mathbf{s}}_2 - E_2\hat{\mathbf{s}}_1) = 0 \quad \text{and} \quad \rho\hat{\mathbf{n}} \cdot (E_2\hat{\mathbf{s}}_3 - E_3\hat{\mathbf{s}}_2) = 0.$$

The above equation shows that $\hat{\mathbf{n}}$ is perpendicular to both $(E_1\hat{\mathbf{s}}_2 - E_2\hat{\mathbf{s}}_1)$ and $(E_2\hat{\mathbf{s}}_3 - E_3\hat{\mathbf{s}}_2)$. Show that $\hat{\mathbf{n}}$ must be parallel to

$$(E_1\hat{\mathbf{s}}_2 - E_2\hat{\mathbf{s}}_1) \times (E_2\hat{\mathbf{s}}_3 - E_3\hat{\mathbf{s}}_2) = E_2(E_1(\hat{\mathbf{s}}_2 \times \hat{\mathbf{s}}_3) + E_2(\hat{\mathbf{s}}_3 \times \hat{\mathbf{s}}_1) + E_3(\hat{\mathbf{s}}_1 \times \hat{\mathbf{s}}_2))$$

Conclude that

$$\rho\hat{\mathbf{n}} = k(E_1(\hat{\mathbf{s}}_2 \times \hat{\mathbf{s}}_3) + E_2(\hat{\mathbf{s}}_3 \times \hat{\mathbf{s}}_1) + E_3(\hat{\mathbf{s}}_1 \times \hat{\mathbf{s}}_2))$$

for some constant k.

By taking the dot product of this relationship with $\hat{\mathbf{s}}_1$, and remembering that $E_1 = \rho(\hat{\mathbf{s}}_1 \cdot \hat{\mathbf{n}})$, show that

$$\rho(\hat{\mathbf{n}} \cdot \hat{\mathbf{s}}_1) = kE_1\,[\hat{\mathbf{s}}_1\hat{\mathbf{s}}_2\hat{\mathbf{s}}_3]$$

where $[\hat{\mathbf{s}}_1\hat{\mathbf{s}}_2\hat{\mathbf{s}}_3]$ is the triple product $\hat{\mathbf{s}}_1 \cdot (\hat{\mathbf{s}}_2 \times \hat{\mathbf{s}}_3)$.

Finally, show that $k = \frac{1}{[\hat{\mathbf{s}}_1\hat{\mathbf{s}}_2\hat{\mathbf{s}}_3]}$, so that

$$\rho\hat{\mathbf{n}} = \frac{(E_1(\hat{\mathbf{s}}_2 \times \hat{\mathbf{s}}_3) + E_2(\hat{\mathbf{s}}_3 \times \hat{\mathbf{s}}_1) + E_3(\hat{\mathbf{s}}_1 \times \hat{\mathbf{s}}_2))}{[\hat{\mathbf{s}}_1\hat{\mathbf{s}}_2\hat{\mathbf{s}}_3]}$$

(c) Photometric stereo experiment [5 marks]

I have given you three images of an unknown surface illuminated from three lightsource directions: $\mathbf{s}_1 = [0, 0, 1]^T, \mathbf{s}_2 = [-1, 1, 2]^T, \mathbf{s}_3 = [1, 0, 3]^T$. Assuming the light source has the same radiance at each position (ie., $E_1 = E_2 = E_3$). Use the expression in part (b) to compute the surface normal and albedo at each point. To accomplish this, complete the program `photometric.m`, and show plots of surface normals and albedo ρ. (Note: Since we don't know E, we can only determine ρ only up to a constant factor.)

2. Fourier analysis and wavelets [5 marks]

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

$$\text{Gabor}(x, y; u, v, \sigma) = e^{i2\pi(ux+vy)}e^{-(x^2+y^2)/(2\sigma^2)}$$

where (x, y) is the pixel ($(0, 0)$ at the center of the filter), (u, v) 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 *cosine gabor* and *sine gabor*, respectively.

$$\Re\,[\text{Gabor}(x, y; u, v, \sigma)] = cos(2\pi(ux + vy))e^{-(x^2+y^2)/(2\sigma^2)}$$

$$\Im\left[\text{Gabor}(x, y; u, v, \sigma)\right] = sin(2\pi(ux + vy))e^{-(x^2+y^2)/(2\sigma^2)}$$

(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 $\frac{1}{2}F(f - f_0) + \frac{1}{2}(f + f_0)$.

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. It should decompress it the directory `cs787-software/`. You should start matlab and do your course work in the `matlab/` subdirectory. (The file `startup.m` will be read on starting Matlab.)

3. Feature Detection [10 marks]

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

$$C = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}$$

where I_x is the image derivative in the x direction, I_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

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix}$$

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 I_x and I_y (case 1 above).

A simple algorithm to find corners is as follows:

1. Smooth the image using a Gaussian kernel.
2. Apply the image derivative operators at every pixel in the image.
3. 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.)
4. Take the singular value decomposition (`svd` in Matlab) for every patch.
5. Choose the matrix with the largest σ_2 and label this as a corner point.
6. Remove any points that are within a $2N$ neighborhood of this corner (to avoid near-duplicate corners).
7. 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. *Please print out your program and submit it (hardcopy) along with your images.*

To blur the image use the Gaussian kernel:

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

You should implement the Gaussian filter efficiently, using *separable* filters in x and y. You may try various values values of σ. A value of $\sigma = 2$ often works well.

To compute the derivatives, you should use the five-point central-difference operator:

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

where $h = 1$.

Comment: Since you have already computed image derivatives, you may wish to experiment with edge detection on this image. The formulas for edge strength and edge orientation are given in Trucco & Verri, §4.2.2.