Computer Vision CS 543 / ECE 549

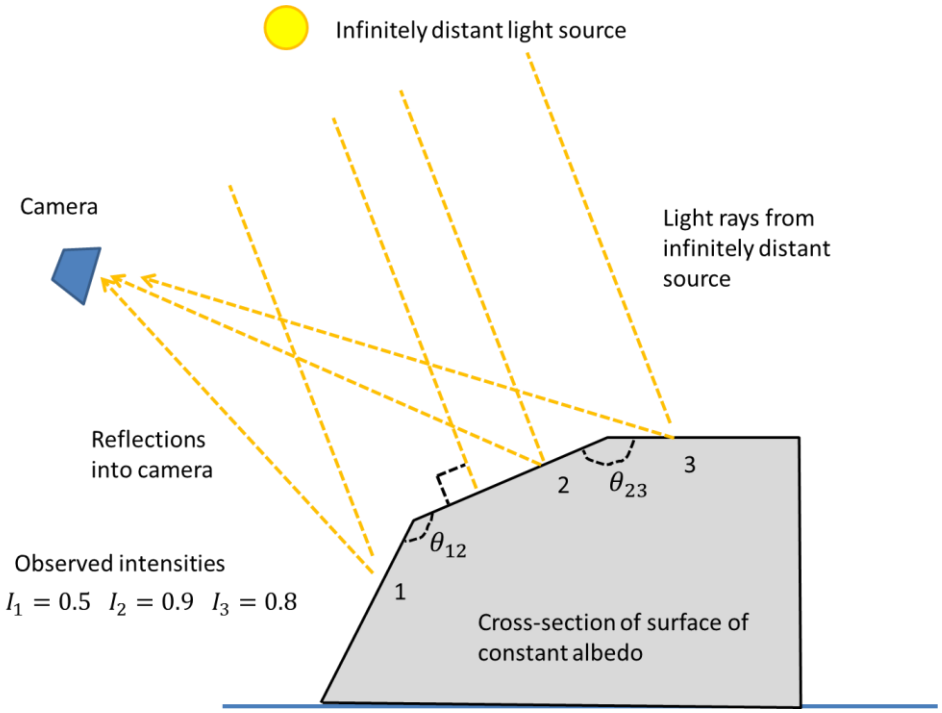
Homework 1 Due Feb 16, 2015

Answer the following questions and explain solutions. Numbers in parentheses give maximum credit value. You can discuss in small groups, but turn in individual solutions and indicate collaborators. Do not use code from the Internet or high-level functions from within Matlab (such as image pyramid or edge detection functions), unless specific permission is given.

Turn in assignments by Monday, Feb 16. Submit to Compass (1) a .pdf with answers, descriptions, and figures; and (2) a .zip file containing your code without any large image or result files. The pdf can be created using latex or converting a word document to pdf or any other method you prefer, as long as it is organized and easy to read.

About this homework: The overall goal of this homework is to review the basics of lighting and filtering and to establish your ability to work with and display images. The time to complete the homework varies widely with expertise and degree of debugging required. Overall, I expect this homework to take significantly less time than HW2. So I suggest trying to complete HW1 before the deadline (Feb 16) and starting on HW2 by Feb 10.

1. **Lighting (20%)**
   A. Answer the following regarding the above image (photo credit: ColinBrough from RGBStock.com). Short answers (several words) are sufficient (8%):
      1. In what direction is the dominant light source: left and above, directly above, or right and above?
      2. Why is one of the temple tips (the part that rests on the ear) so bright, considering that the other tip which has the same material is very dark?
      3. What causes the dark streaks in the wood (in terms of shape, albedo, reflectance, etc.)?
      4. If the table were completely specular, would the glasses cast a shadow on it (explain why or why not)?



B. Answer the following using the above illustration. Suppose you have observed the intensities of three points on an object (I_1, I_2, I_3), which are lit by an infinitely distant point source (the sun). The surface normal at point 2 is exactly perpendicular to the sun. The surface normals of points 1 and 3 differ in only one angle (θ), as shown in the cross-section.
   a. Suppose the surface has a specular component. Will the observed intensities change as the camera moves (if so why/how)? (4%)
   b. Suppose the surface material is Lambertian and has uniform (constant) albedo and that the camera response function is linear (and ignore effects due to interreflections in the scene). Express the intensities in terms of the between the surface normal and the lighting direction. Then, show (with equations for arbitrary observed intensities) how to compute the angles θ_{12}, θ_{23} between surfaces containing points 1 and 2 and points 2 and 3. Finally, compute the values of θ_{12}, θ_{23} for the observed intensities (0.5, 0.9, 0.8). (8%)

2. Image Pyramids (30%)

Choose an image that has an interesting variety of texture (from Flickr or your own images). The image should be at least 640x480 pixels and converted to grayscale. Write code for a Gaussian and Laplacian pyramid of level N (use `for` loops). In each level, the resolution should be reduced by a factor of 2. Show the pyramids for your chosen image and include the code in your write-up.

1) Display a Gaussian and Laplacian pyramid of level 5 (using your code). It should be formatted similar to the figure below. You may find `tight_subplot.m`, included in hw1.zip, to be helpful. (15%)
2) Display the FFT amplitudes of your Gaussian/Laplacian pyramids. Appropriate display ranges (using imagesc) should be chosen so that the changes in frequency in different levels of the pyramid are clearly visible. Explain what the Laplacian and Gaussian pyramids are doing in terms of frequency. (15%)

Useful functions include: `imfilter, fft2, cell, figure, subplot, imagesc(im,[minval maxval]), colormap, mat2gray, help/doc`

Your displayed images for the Gaussian and Laplacian pyramids should look something like this:

3. Edge Detection (50%)

Overview: The main steps of edge detection are: (1) assign a score to each pixel; (2) find local maxima along the direction perpendicular to the edge. Sometimes a third step is performed where local evidence is propagated so that long contours are more confident or strong edges boost the confidence of nearby weak edges. Optionally, a thresholding step can then convert from soft boundaries to hard binary boundaries.

I have provided 50 test images with ground truth from BSDS, along with some code for evaluation. Your job is to build a simple gradient-based edge detector, extend it using multiple oriented filters, and then describe other possible improvements. Details of your write-up are towards the end.

a) Build a simple gradient-based edge detector that includes the following functions (25%)

`function [mag, theta] = gradientMagnitude(im, sigma)`
This function should take an RGB image as input, smooth the image with Gaussian std=sigma, compute the x and y gradient values of the smoothed image, and output image maps of the gradient magnitude and orientation at each pixel. You can compute the gradient magnitude of an RGB image by taking the L2-norm of the R, G, and B gradients. The orientation can be computed from the channel corresponding to the largest gradient magnitude. The overall gradient magnitude is the L2-norm of the x and y gradients. `mag` and `theta` should be the same size as `im`.

`function bmap=edgeGradient(im)`
This function should use gradientMagnitude to compute a soft boundary map and then perform non-maxima suppression. For this assignment, it is acceptable to perform non-maxima suppression by retaining only the magnitudes along the binary edges produce by the Canny edge detector: `edge(im, 'canny')`. Alternatively, you could use the provided `nonmax.m` (be careful about the way orientation is defined if you do). You may obtain better results by writing a non-maxima suppression algorithm that uses your own estimates of the magnitude and orientation. If desired, the boundary scores can be rescaled, e.g., by raising to an exponent: `mag2 = mag.^0.7`, which is primarily useful for visualization.

Evaluate using `evaluateSegmentation.m` and record the overall and average F-scores.

b) Try to improve your results using a set of oriented filters, rather than the simple derivative of Gaussian approach above, including the following functions: (15%)

`function [mag,theta] = orientedFilterMagnitude(im)`
Computes the boundary magnitude and orientation using a set of oriented filters, such as elongated Gaussian derivative filters. Explain your choice of filters. Use at least four orientations. One way to combine filter responses is to compute a boundary score for each filter (simply by filtering with it) and then use the max and argmax over filter responses to compute the magnitude and orientation for each pixel.

`function bmap= edgeOrientedFilters(im)`

Similar to part (a), this should call `orientedFilterMagnitude`, perform the non-maxima suppression, and output the final soft edge map.

Evaluation: The provided `evaluateSegmentation.m` will evaluate your boundary detectors against the ground truth segmentations and summarize the performance. You will need to edit to put in your own directories and edge detection functions. Note that I modified the evaluation function from the original BSDS criteria, so the numbers are not comparable to the BSDS web page. The overall and average F-score for my implementation were (0.57, 0.62) for part (a) and (0.58, 0.63) for part (b). You might do better (or slightly worse).

Write-up: Include your code with your electronic submission. In your write-up, include:

- Description of any design choices and parameters
- The bank of filters used for part (b) (imagesc or mat2gray may help with visualization)
- Qualitative results: choose two example images; show input images and outputs of each edge detector
- Quantitative results: precision-recall plots (see "pr_full.jpg" after running evaluation) and tables showing the overall F-score (the number shown on the plot) and the average F-score (which is outputted as text after running the evaluation script).

c) Ideas for improvement (10%): Describe at least one idea for improving the results. Sketch an algorithm for the proposed idea and explain why it might yield improvement. Improvements could provide, for example, a better boundary pixel score or a better suppression technique. Your idea could come from a paper you read, but cite any sources of ideas.

d) Extra credit (up to 10%): Try to improve the results that you get in (b), e.g., using your idea from (c). Include code, explain what you did, and show any resulting improvement. Note: this is not an easy way to get points; 10 pts would be reserved for big improvements.

Related papers:
Reading these will aid understanding and may help with the assignment.

The design and use of steerable filters:
http://people.csail.mit.edu/billf/papers/steerpaper91FreemanAdelson.pdf

Berkeley Pb Detector:
http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/papers/mfm-pami-boundary.pdf

Multi-scale edge detection:
http://www.cs.washington.edu/homes/xren/publication/xren_eccv08_multipb.pdf