Assignment 8

CS283, Computer Vision
Harvard University

Due Tuesday Nov. 7 at 5:00pm

This assignment deals with color and texture. Remember to adhere to the formatting guidelines posted on the course website, and submit via Canvas.

1. (*30 points*) A reasonable reflectance model for dielectric (non-conducting) surfaces is the so-called dichromatic model, according to which the spectral BRDF is written as a linear combination of a Lambertian diffuse component f_d and a wavelength-independent specular component f_s:

$$f(\lambda, \hat{\omega}_i, \hat{\omega}_o) = f_d(\lambda) + f_s(\hat{\omega}_i, \hat{\omega}_o).$$

When one images a scene comprised of such surfaces under a directional light source with spectral power distribution $I(\lambda)$ and direction $\hat{l} = (l_x, l_y, l_z)$, the RGB values $\vec{C}(\vec{u}) = (C_R(\vec{u}), C_G(\vec{u}), C_B(\vec{u}))$ recorded at pixel $\vec{u} = (u, v)$ can be expressed as

$$\vec{C}(\vec{u}) = \langle \hat{n}(\vec{u}), \hat{l} \rangle \vec{d}(\vec{u}) + g_s(\vec{u})\vec{s},$$

where $\langle \cdot, \cdot \rangle$ is the inner product operation, $\hat{n}(\vec{u})$ is the surface normal at the scene point imaged by pixel $\vec{u}$, and $g_s(\vec{u})$ is a function that depends non-linearly on $\hat{n}(\vec{u})$ (as well as view and lighting directions) through the specular component of the BRDF.

   (a) Assuming that the spectral sensitivities of a camera's three filters are $(c_R(\lambda), c_G(\lambda), c_B(\lambda))$ and that the BRDF at the surface point imaged at pixel $\vec{u}$ is $f(\lambda, \hat{\omega}_i(\vec{u}), \hat{\omega}_o(\vec{u})) = f_d(\lambda, \vec{u}) + f_s(\hat{\omega}_i(\vec{u}), \hat{\omega}_o(\vec{u}))$, write expressions for the elements of the *diffuse color vectors* $\vec{d}(\vec{u})$ and the *source color vector* $\vec{s}$.

   (b) Suppose you are given two unit-length three-vectors $\hat{r}_1$ and $\hat{r}_2$ that are orthogonal to $\vec{s}$. Show that the two-channel image[1] given by the per-pixel inner products $\vec{J}(\vec{u}) = (\langle \hat{r}_1, \vec{C}(\vec{u}) \rangle, \langle \hat{r}_2, \vec{C}(\vec{u}) \rangle)$:

      i. does not depend on the specular components of the BRDFs, $f_s(\hat{\omega}_i(\vec{u}), \hat{\omega}_o(\vec{u}))$.

      ii. depends linearly on the surface normals, $\hat{n}(\vec{u})$.

   (c) Show that the two properties from part (b) are also satisfied by the single-channel (grayscale) image $J(\vec{u}) = ||\vec{J}(\vec{u})||$.

   (d) Write a function `imout=makeLambertian(im,s)` that takes an RGB image `im` and a source color vector `s` and computes the grayscale image $J(\vec{u})$ from part (c). Run this function on the image `fruitbowl.png` from the data folder using `s=[0.6257 0.5678 0.5349]` and submit the results. Explain in three sentences or less why `imout` might be more useful than `im` to a computer vision system. (See hints below for help with displaying these images.)

2. (*25 points*; inspired by Bill Freeman) Two distinct spectral distributions are *metameric* if they induce the same responses in the cones of a human retina. Said another way, *metamers* are distinct spectral distributions that map to the same tristimulus vector (in CIE XYZ or any other linear color space). If you own a grocery store that sells bananas, it is in your interest to choose your light source so that overripe bananas look that same as those in their prime. That is, you want the light reflected from ripe and overripe bananas to be metameric.

   Let $\vec{f}$ and $\vec{g}$ be $N \times 1$ vectors that come from discretizing the spectral reflectance functions $f(\lambda)$ and $g(\lambda)$ of ripe and overripe bananas, respectively. (For example, if we sampled from 400nm to 700nm in increments of 10nm, we'd have $N = 31$.) Let R be the $3 \times N$ matrix whose rows are obtained by discretizing the CIE XYZ color matching functions (FP Fig. 4.7, right).

[1] This image can have pixels with negative values, and these negative values are just as useful as the positive ones.

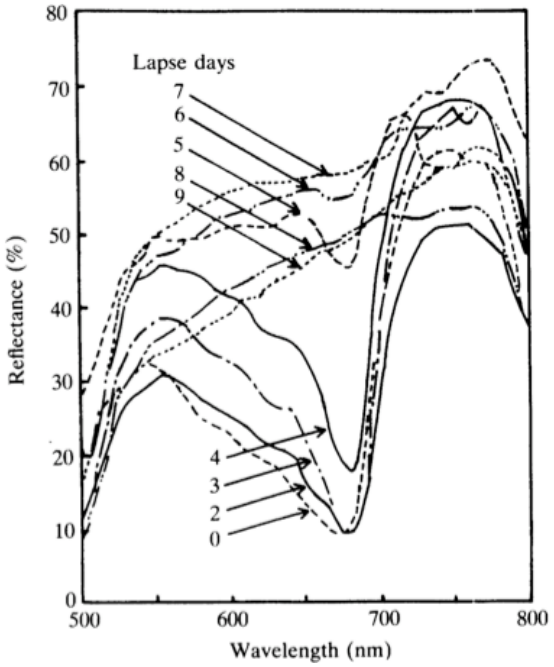

Figure 1: Relative spectral reflectance curves for different colors of bananas during ripening. Day five is the first day of shelf life. (Taken without permission from Morita et al., " Evaluation of Change in Quality of Ripening Bananas Using Light Reflectance Technique", in *Memoirs of the Faculty of Agriculture, Kagoshima University*, 1992.)

(a) Show that the tristimulus vector $\vec{C}_f = (X_f, Y_f, Z_f)$ for a ripe banana $f(\lambda)$ under illuminant spectrum $l(\lambda)$ can be written as $\vec{C}_f = L_f \vec{l}$ where L_f is a $3 \times N$ matrix and $\vec{l}$ is the discretization of $l(\lambda)$.

(b) Given $\vec{f}$ and $\vec{g}$, a good way to choose $\vec{l}$ is to minimize the distance, in a least-squares sense, of the two resulting tristimulus vectors. Write an expression for the Euclidean (L2) distance between the two tristimulus vectors in terms of L_f, its counterpart L_g, and $\vec{l}$.

(c) Write a Matlab function `T=metamericLight(f,g)` that finds the temperature T in the closed interval $T \in [2500\text{K},\ 10,000\text{K}]$, with precision ± 50K, of the blackbody radiator that minimizes the distance derived in part (b). (See hints and information below.) The file `bananas.mat` from the data folder contains two spectral reflectance functions, `ripe` and `overripe`, which are sampled in 10nm increments from 400nm to 700nm and correspond roughly to the data from Day 5 and Day 7 of Fig. 1. Use your function to find the optimal metamer-inducing blackbody temperature for these materials, and plot the distance as a function of temperature, and the spectral power distribution of the best illuminant. You will need to use the file `ciexyz64_1.csv` from the data folder, which contains tabulated data for the CIE XYZ color matching functions. Remember to normalize the spectral power distributions of blackbody radiators in a way that is meaningful for the minimization criterion of part (b).

3. (*30 points*) In this problem you will implement the Efros and Leung algorithm for texture synthesis. It is depicted in Figure 10.50 of Szeliski, described briefly in Section 9.3 of Forsyth and Ponce, and described in detail on Alyosha Efros's webpage. As discussed in class, the Efros and Leung algorithm synthesizes a new texture one pixel at a time. For each unsynthesized pixel in the target image, a weighted sum-of-squared differences (wSSD) measure is used to identify pixels of a source texture image that are good candidates for filling the target pixel, and the pixel is filled by randomly selecting from among these candidates. The target is initialized by randomly selecting a 3×3 patch from the

source image and placing it in the center of the target. The boundaries of this seed patch are then recursively filled until all pixels in the target image have been assigned values.

(a) The first step is to write a function that searches a source texture image for regions that match a given neighborhood of the target. This function will have the syntax

```
[pixvalues, matcherrors] = FindMatches(targetwindow, validmask, sourceimage, G)
```

where `targetwindow` is a $w \times w$ window extracted from a neighborhood of the target pixel to be filled; `validmask` is $w \times w$ binary matrix whose elements indicate which pixels of that window contain already-synthesized values; `sourceimage` is the grayscale source texture image; and G is a $w \times w$ Gaussian kernel to be explained below. The output is a list `pixvalues` of candidate gray-values to insert into the target pixel, and a corresponding list `matcherrors` that reports the wSSD associated with each candidate.

When matching the target window to the source image, Efros and Leung suggest down-weighting pixels that are further from the center of the target window, with the intuition that these are less likely to be related to the target pixel being filled. This is accomplished by computing the wSSD with weights that are the unit-sum-normalized, element-wise product of `validmask` and `G`:

```
weights = validmask.*G; weights=weights/sum(weights(:));
```

Efros and Leung suggest using a Gaussian with standard deviation $\sigma = w/6.4$.

Your function `FindMatches` cannot contain any loops. This can be accomplished by exploiting the relationship between wSSD and convolution. See Hints and Information below. Validate your code using the provided script in the .MLX file.

(b) The next step is to write the main function

```
synthim = SynthTexture(sourceimage, w, synthdims)
```

that synthesizes a large texture image `synthim` of size `synthdims=[height width]` from the source texture image `sourceimage` using window size `w`. It will be helpful to look at the pseudo-code on Efros' webpage.

This function should initialize by placing a 3×3 seed patch from the source at the center of the target. It will then grow this seed patch recursively by synthesizing values around the border of the already-filled pixels. A useful technique for recovering the locations of boundary pixels in Matlab is *dilation*, a morphological operation that expands regions of a binary image. See Section 3.3.2 of Szeliski and the Matlab `imdilate` and `find` commands.

For each target pixel to be filled, use a call to your function `FindMatches` to obtain a list of candidate gray-values and their corresponding wSSD errors. Then, randomly select a gray-value from the subset of `pixvalues` whose wSSD error is less than $(1+\varepsilon)$ times the minimum wSSD error. To avoid randomly selecting a gray-value with unusually large error, also check that the wSSD error of the randomly selected gray-value is below threshold δ. Efros and Leung use threshold values of $\varepsilon = 0.1$ and $\delta = 0.3$.

(c) Test and run your implementation using the grayscale source texture image `rings.jpg`, with window widths w = 5,7, and 13 and `synthdim`= [100 100]. Instead of using a random seed, deterministically use a 3×3 window of the source whose top-left corner is the fifth row and fourth column in a Matlab indexing system (i.e., `sourceimage(5:7,4:6)`).

Explain the algorithm's performance with respect to window size. For a given window size, if you re-run the algorithm with the same starting seed do you get the same result? Why or why not? Is this true for all window sizes?

In your submission, include plots of the synthesized textures that correspond to each window size along with answers to the above questions and your code.

Hints and Information

- A simple way to numerically obtain a set of $N-1$ unit-length vectors that are orthogonal to N-vector $\vec{v}$ is to use the `null` command in Matlab.

- The image `fruitbowl.png` from the assignment's data folder is a 16-bit image with relatively high dynamic range, and it does not display well using standard image viewers. One way to get a reasonable depiction of it is to multiply by a large gain factor and then clip the bright highlights:

```
im=im2double(imread('fruitbowl.png'));
imshow(min(1,im./0.1));
```

- The spectral power distribution for a blackbody radiator (see FP Sect. 6.1.2) with temperature T is

$$L_T(\lambda) \propto \frac{1}{\lambda^5} \frac{1}{e^{\frac{a}{\lambda T}} - 1}$$

  where $a = 1.4388 \times 10^{-2} (\mathrm{m} \cdot \mathrm{K})$.

- In Matlab, you can read data from a CSV file using the `csvread` command.

- To avoid loops in Matlab, the weighted sum-of-squared-differences (wSSD) between a sliding template $I_0(x, y)$ and an image $I_1(x, y)$ can be written as a sum of three convolutions. To see this, let the spatial indices of the template be such that the center of the template is $(x, y) = (0, 0)$. Then the wSSD output map (same size as the image) is given by the expression

$$wSSD(u, v) = \sum_{xy} w_0(x, y) w_1(x + u, y + v) \left(I_0(x, y) - I_1(x + u, y + v) \right)^2 .$$

  Expanding gives

$$\begin{aligned} wSSD(u, v) &= \sum_{xy} w_0(x, y) w_1(x + u, y + v) I_0^2(x, y) \\ &\quad -2 \sum_{xy} w_0(x, y) w_1(x + u, y + v) I_0(x, y) I_1(x + u, y + v) \\ &\quad + \sum_{xy} w_0(x, y) w_1(x + u, y + v) I_1^2(x + u, y + v) \\ &= (w_0 I_0^2) \otimes w_1 - 2(w_0 I_0) \otimes (w_1 I_1) + w_0 \otimes (w_1 I_1^2), \end{aligned}$$

  and the result follows from the fact that convolution is equivalent to correlation with a flipped template. (This is all discussed in Section 8.1.2 of Szeliski. He also notes that Fourier transforms can thus be used to accelerate wSSD for large images, but since our images are small in this assignment, there is no such benefit here.)

  In this assignment, I_0 is a $w \times w$ region from the target texture image, w_0 is the associated $w \times w$ unit-sum mask of weights (e.g., zero-valued at places where the target has not yet been synthesized), I_1 is the source texture image, and w_1 is a matrix the same size as the source with elements equal to one.