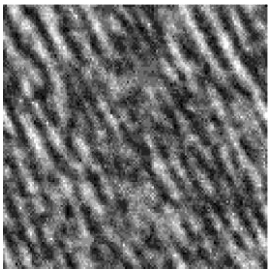
Computer Vision I : Project #2 (total 10 points)

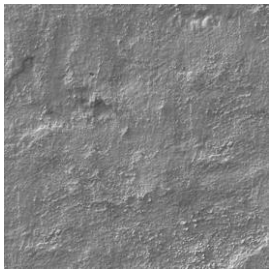
Sampling Julesz Ensemble and FRAME Model

Due Oct. 20 11:59pm

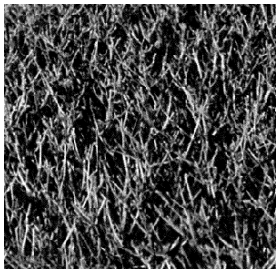
Objective: This project explores the minimax entropy learning procedure for texture modeling and synthesis. We use three texture examples below which you can find in the zip file.



(a) fur



(b) stucco



(c) grass

Part 1: Julesz Ensemble

Experiment: For each input image, select a set of filters and extract the histograms of filter responses. We define the **Julesz ensemble** as the set of images that reproduce the observed histograms over those selected filters. Given a selected set of filters and their histograms, you use the **Gibbs sampler** to draw samples from the Julesz ensemble. You need an **annealing scheme**. The synthesis starts with a uniform noise image drawn from a uniform distribution, and the sampling process stops when it matches all the selected histograms. E.g., it continues to select more filters until all the filtered histograms are matched within an epsilon error.
To reduce computational complexity, you may synthesize the image in 8 gray levels [0,7] with size 256x256 pixels using torus boundary condition. Note it is more important to match histograms closely at the tail bins of the histograms. E.g. you may use these weights for the 15 bins: [8, 7, 6, 5, 4, 3, 2, 1, 2, 3, 4, 5, 6, 7, 8].

Instructions:

1. "filters.py" contains a Python function "get_filters()", which generates a set of filters in the format of matrices.

Modification: you need to add more filters. For instance, you should at least add the Dirac delta function whose response is the intensity of a given pixel.

2. "julesz.c" contains a C function "getHistogram()", which computes the normalized filter response histogram on an image.

Modification: on top of this function, you should also write another one by yourself, e.g., "getHistogram2()", where you can specify the bin range for computing the histogram. Note that the output histogram is normalized by 256 * 256. You need to multiply the normalized histograms by 256 * 256 for computing the difference (error) between two histograms.

3. “julesz.c” contains a C function “Gibbs()”, which implements a Gibbs sampling process for the image synthesis.

Modifications:

a) When computing the histograms of input image and the synthesized image, you must use the same bin range. For this, you may compute the maximum and minimum responses of the response of a filter on the input image, and then divide the range equally to obtain the interval of each bin. Use these bin ranges for both input and synthesized image.

b) Assign different weights to the bins when you calculate the difference between two histograms, e.g.:

```
for (x = 0; x < num_bins; x++) {
        probs[k+gray] = probs[k+gray] + abs(difference[x]) * w[x];
}
```

c) Tune the coefficient of the simulated annealing process to improve the synthesis.

4. “julesz.py” is an example of the filter selection process in Python.

Note that this naïve script does not work, since the filters should be selected based on the information gain, not in the order of “1,2,3,4,5” shown in the script, and the bin ranges should be computed based on the filters’ responses on the input image.

5. Write your own main script to replace “julesz.py”, which should include the following main steps of the learning algorithm:

a) Generate a filter bank by “filters.py”.

b) The set of chosen filters S is initially empty.

c) Randomly sample a synthesized image I (sample the intensity of each pixel uniformly from [0,7]). You may also use other types of initial synthesized image, e.g., an all black image.

d) For each unselected filter, compute its response histogram on both input image and current synthesized image I. Calculate the weighted difference (error) between two histograms.

e) If the highest error is smaller than a threshold, then go to step 8. Otherwise, choose the filter with the highest error and add it to the chosen set S.

f) Use the modified Gibbs function to sample a new image I', and replace I with I'. (Hint: the input of the Gibbs function includes the synthesized image I from the last iteration, chosen filters set S, histograms of chosen filters' responses on the input image, and the corresponding bin ranges of the histograms.)

g) Compute the total error between the new I and the input image based on the new set S. This is the error that you need to show for the report. Then go to step 4.

h) Output the synthesized image I as the final result.

Submission:

1. Show the set of selected filters (enlarge them to make them visible) and print the sequence of images that you synthesize by increasing the filter number.
2. Plot the errors of the histograms of filtered responses over the iterations (after choosing a new filter).

Hints for getting nice results:

i) Match the histogram closely, especially at the tails; and

ii) Try other filters not in the file.

Part 2: FRAME Model

Objective: This part explores the procedure for learning the filters, random field, maximum entropy (FRAME) model, a well-defined mathematical model for textures combining filtering and random field models. Maximum entropy is used when constructing the probability distribution on the image space. Minimum entropy is used when selection filters from a filter bank. Together this scheme is called the min-max entropy principle.

Recall the principle of maximum entropy to construct a probability distribution p on some random variables X.

Let $\phi_n(x)$ be some known function with $E_p[\phi_n(x)] = \mu_n$.

Let $\Omega = \{p(x) | E_p[\phi_n(x)] = \mu_n, n = 1, \dots, N\}$.

The principle of maximum entropy dictates that $p \in \Omega$ which maximizes entropy $H(p)$ is a good choice. This leads to a solution in the form of

$$p(x;\Lambda) = \frac{1}{Z(\Lambda)} \exp\{-\sum_{n=1}^{N} \lambda_n \phi_n(x)\}$$

with Lagrange multipliers $\Lambda = \{\lambda_1, \lambda_2, \dots, \lambda_N\}$ and
partition function $Z(\Lambda) = \int \exp\{-\sum_{n=1}^{N} \lambda_n \phi_n(x)\}\ dx$.

From this principle you can derive the FRAME model in the textbook Chapter 3.4.

Instructions:

To learn the FRAME model, you will have to select filters and learn Λ:

1. Read the relevant chapter in the textbook.
2. Create copy "frame.c" of "julesz.c". Adjust the Gibbs sampling code to compute the term $-\sum_n \lambda_n \phi_n(x)$.
3. Create a copy "frame.py" of "julesz.py". Adjust the code to define Λ and update $\{\lambda_1, \lambda_2, \dots, \lambda_N\}$ by gradient ascend on $\log p(x;\Lambda)$, i.e.,

$$\frac{d\lambda_n}{dt} = \frac{d \log p(x;\Lambda)}{d\ \lambda_n} = E_{p(x;\Lambda)}[\phi_n(x)] - \mu_n\ , \qquad n = 1,2, \dots, N.$$

Submission:

1. Plot the estimated $\Lambda = \{\lambda_1, \lambda_2, \dots, \lambda_N\}$ over learning iterations.
2. Show the set of selected filters and print the sequence of images that you synthesize by increasing the filter number.