IITB, EE 610 Image Processing, July-Nov 2021, Assignment 2

The objective of this assignment is to use machine learning to train a non-linear filter to restore images. You will learn the effect of various window sizes and model complexities. The entire work has to be submitted as a single ipython notebook with various cells for comments, observations, and motivations for next steps in lieu of the sections of a PDF report. If you still want to use MATLAB, then submit a PDF report in IEEE paper style.

1. Download color images from the BSD dataset: https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/
2. Prepare the training dataset: [2]
    a. Select the largest odd window size W, e.g. 13 or 27
    b. Prepare a few blur kernels and noise models
    c. For each training image
        i. Degrade multiple times using different blur kernels and noise models
        ii. Display a few images to check if the degradation is realistic looking instead of too much or too little
        iii. For each degraded image version
            1. Mine and store degraded patches of size WxW and central pixel of original patch
3. Train a regression model:
    a. Select a window size w less than or equal to the largest window size W
    b. Select a machine learning model (nonlinear regression), e.g. support vector regression, random forest regression, neural network regression, or convolutional neural network
    c. Write a function to read only the w×w central pixels as input, and (optionally) pre-process them (e.g. make it zero mean and unit variance, or work in HSI space) [1]
    d. In python, train a regression model to predict the clean (optionally, normalized) central pixel [1]
    e. Monitor the normalized mean square error or mean absolute error for validation data [1]
    f. Observe if models over-fits. If so, then implement early stopping
    g. Experiment with different choices, e.g. window size, machine learning models, capacity of models (e.g. tree depth, SVR penalty, number of hidden nodes in NN, number of layers and kernels in CNN, etc.) to find a reasonable model with small normalized RMSE, e.g. less than 1% or 2%. [2]
4. Apply the model on held-out or private images (couple of them synthetically degraded, couple of them real images with slightly poor quality): [2]
    a. For each degraded testing image
        i. Initialize blank clean image
        ii. For each overlapping patch of $w_{opt} \times w_{opt}$
            1. Predict clean central pixel
            2. Optionally denormalize (multiply the std deviation, add the mean, or inverse HSI)
        iii. Contrast enhance using a function call (no need to implement from scratch) and put the pixels in the right range (0-255, integer)
        iv. Display the cleaned image
5. Your report or ipython comment cells should contain the following: [2]
    a. Your design choices: which window size range and ML models you will try
    b. Your initial guess of what trends you will observe about what will work better
    c. Your observation, and any surprises in what worked better. E.g., was there a maximum window size above which you did not get any advantage when you increased the window size further? Did you expect CNN to be better, but something else turned out to be better? Did you expect a more complex model to work better?
    d. Some thoughts on why you think you had any surprising observations.
    e. References, including internet sources of code, blogs, or friends with whom you discussed.