Assessment and Restoration of noisy Images

Report File

Submitted on
18/08/2017

ISI Kolkata, CVPR Unit
Project Mentor: Sarbani Palit

Dibakar Sil,
National Institute of Technology, Durgapur.
Dept. ECE

Sujoy Kansabanik,
Jadavpur University,Kolkata
Dept. CSE

2nd Year Summer vacation project.
Duration: May-July, 2017

We are trying to reconstruct an image having no reference given. We tried our best to get the image as close as 35 PSNR value from a noisy image.

Introduction:

We were trying to automatically detect what kind of noise is implemented in an image and were trying to enhance the quality of the image (greater than 30PSNR).

To achieve that goal, we took following algorithms.

***Algorithms* for detecting noise:**

We used Machine Learning to detect it! We trained the machine implementing Gaussian, Rayleigh and Uniform Noise in a set of images individually in Octave through Image Processing tool. We used command 'imnoise' and 'imnoise2'. We took the histogram as a feature having number of bins 256 i.e. there is 256 features and one single labelling for each Noise Identification i.e. for gaussian it's labelling is 1, for rayleigh it's 2 and for uniform it's 3. Then we used Gaussian Naive Bayes as a classifier in Python 3 to detect the corresponding noises in a test image.

Comment: We trained the machine in such a way that it can detect the noise in the range of 0.0001< S.D.(sigma) <0.01. i.e. Here we can detect the form of noise in multimodal format as well. It can detect on an average of 35 PSNR value.

Software used: Octave, Python 3
Modules used: scikit-learn for machine learning in python 3 and octave image processing tools.

Algorithms for enhancing the test image:

After identifying the noise, we used image enhancement processes to improve the quality of the image. These processes are used:

1.*Blurring:* For the gaussian noise images, we are blurring the image by averaging it with appropriate mask.

For rayleigh noise, we were doing median filtering using 'medfilt' command with appropriate masking.

And result is storing in an another variable let's say it's B.

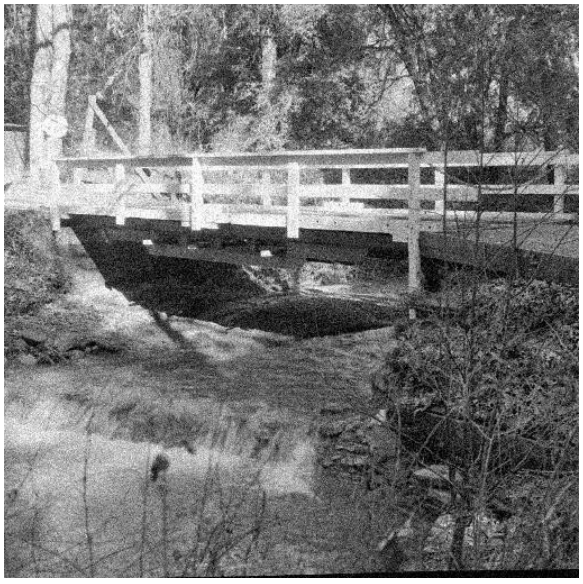
Bridge-blurring for gaussian

Leena-Blurring for gaussian

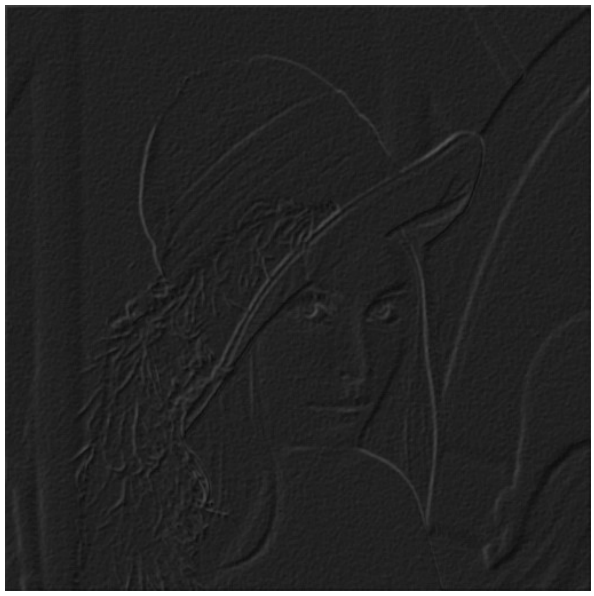
Rayleigh noise cancellation using median filter

2. *Edge Detection*: We detected the edge by using 'Laplace filter' or 'stdfilt' then stored this convolved matrix in a variable (let's say it's E).

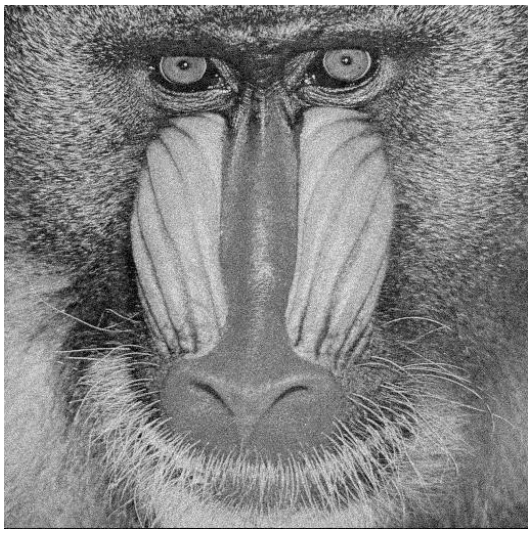
To get more clear information, in each and every steps of the algorithms, we are using 'mat2gray' for better result.



'Laplace' filtering using 3x3 mask having weightage [0 -1 0;-1 4 -1;0 -1 0]



‘Laplace’ filtering using 3x3 mask having weightage [0 -1 0;-1 4 -1;0 -1 0]



using ‘stdfilt’

3. *Sharpening:* After doing that, we are multiplying E with an appropriate value. Lets say its V. And then adding it with B.

i.e. at the end we get something like,

S=B+(V*E)

this S variable is sharpen result.

Result of Sharpening

4. *Gamma Correction*: Gamma encoding of images is used to optimize the usage of bits when encoding an image, or bandwidth used to transport an image, by taking advantage of the non-linear manner in which humans perceive (OQA) light and color.

For any non-linear image, gamma can be applied. As the **power relationship**

Vout =**V**in .^ gamma;

The curve in log-log plot is linear with the slope equal to gamma.

where, **gamma=(d(log(Vout))) / (d(log(Vin)))**

PSNR 19.7 PSNR 27

Gamma Corrected Image
(following the above steps once)

5. *Histogram Equalization*: Here we are trying to enhance the image G by finding more details. It lead us good result sometimes and not so good in some cases.

Here we are listing it because sometimes it gave us a nice result.

6. *Homomorphic Filtering*: It gave us a good result in case of edge detailing in lower or higher contrust.

We applied it in G for getting some more details. It lead us a good result.

Sometimes we were trying to find details using DWT. It helps us getting an approximate result.

C*onclusion:*

After following the above steps one time, we got the final image having PSNR value 27 where the noisy image have the PSNR approx. 20. Using the above steps multiple times (2-3) with appropriate values and mask, we can achieve PSNR value on an average of 35.

Reference:
https://drive.google.com/file/d/0B0lq-mwCZDchXzBHWGxSN05CNWM/view?ts=5996dfac

For making training set -

https://drive.google.com/file/d/0B0_uP_xj2v-AdGR3RnJyTmRuakU/view?ts=5996ef45