Column Noise Reduction in Motion Pictures

Vivek Singh Bhadouria, E-mail: vivek@hotmail.de

Introduction

This report details the process of simulating the column noise and its reduction process.

Tools Used:

Python version-3, and its various standard libraries e.g. OpenCV, Scikit image, Numpy to name a few.

Methodology

This experiment has been performed on RGB color space on a per plane basis however, a vector color processing based approach is also feasible and probably more effective in terms of color restoration.

Synthetic Data Generation

Noise model: It is assumed that such column noise sets the pixel value to zero. In terms of density modeling, it is assumed that in any frame 1% of image columns are corrupted i.e. [19,21]. It is important to note that noise can appear in any column therefore, this fact has been considered while generating the synthetic data.

Above mentioned assumptions were utilized and synthetic data was generated by using a test FHD video from the https://www.demolandia.net/. The sequence is of 25 FPS rate. FHD with 25FPS was considered to limit the computational resource requirements. However, data generation script is scalable and will work for any video resolution and FPS.

Synthetically generated data can be seen at https://youtu.be/aHTusaLSn1E.

Noise reduction algorithm

Stage1: It is essential to determine the noisy columns before applying any filtering operation because unwanted filtering operation will result in blurring of the image. A simple mechanism is utilised here to determine the noisy columns i.e. algorithm marks a column as a noisy column if sum of all the pixel values along any given column is zero.

For other noise models, a dedicated detection mechanism has to be developed.

Stage2: Modified order statistics filter was utilized to remove the column noise. It is being referred here as “modified” because only non-zero values were considered to reconstruct the replacement pixel for the noisy pixel.

Note that noisy density is considered to be 1% and in such conditions, smaller kernels yield good results however, at higher noisy density there is a significant probability that many consecutive columns are affected by noise. In such scenarios, small kernels will not be useful and a more image impaining type of approach will be useful.

Stage3: Increasing sharpness: After the noise removal, reconstructed image has been sharpened using unsharp masking approach. However, intention is to keep the original content intact therefore, columns (from reconstructed and sharpened image) belonging to noisy locations were copied to the noisy image. Effectively, noisy pixels have been replaced by reconstructed and sharpened pixels. The final output can be seen here: https://youtu.be/KwOZzRCgVg4

Limitations

1. Code is not optimized for performance therefore, one can observe frequent interchanging of data formats.
2. Code is not memory-access optimized therefore, it will have performance limitations with respect to system I/O.
3. Code has been coarsely parallized to utilize all the cores available on the computer so as to chive higher throughput.
4. FFMPEG was used to extract frames from the video and additionally for collating the frames to video.

Folder Layout and Source Layout

Folder Layout

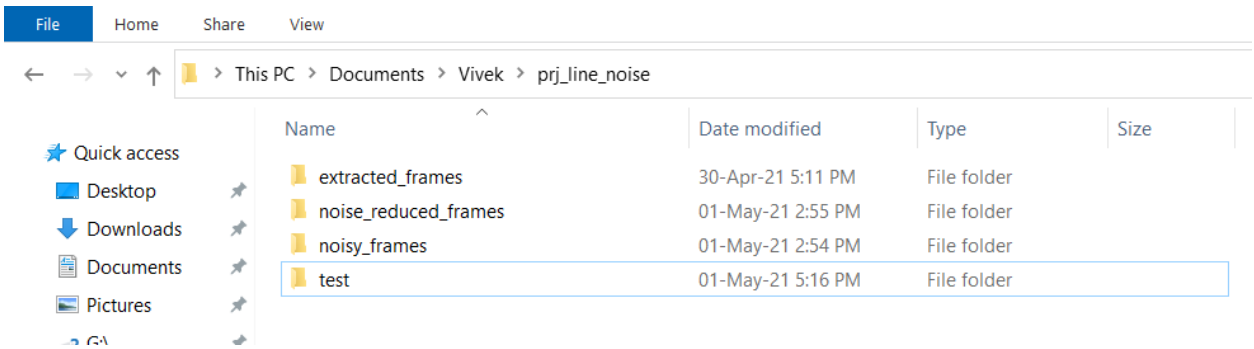

Fig. 1 Image illustrating folder structure

Source code

Source code is located at https://github.com/vivekalig/column_noise_removal.git

END OF REPORT