Noise Measurement in DevWare

Formulas for the Diagnostics Noise Measurement dialog

Introduction

This document details the calculations performed by the Sensor Control – Diagnostics – Noise Measurement GUI in DevWare/DevWareX.

Region of Interest

The Noise Measurement calculations are performed on the region of interest (ROI) defined by the Rectangle Mouse Selection. Note that row and column indexes in DevWare start from zero. The raw data will be separated by color plane, and the ROI defined on the GUI will be projected into each plane. The ROI may be adjusted slightly if needed so that it is aligned to the color filter array (CFA) pattern.

For Bayer data, or any other 2×2 CFA there will be four planes, and the ROI will be the same size in each plane. For the RGB-IR 4×4 CFA there will be five planes, and the red and blue planes in the ROI will have exactly half as many pixels as the green and IR planes.

Definitions

For the following definitions the data has *I* rows, *J* columns and *K* frames. Each color channel is analyzed individually. Variances in the time domain are using the unbiased estimation; *K* – 1 in the denominator. Noise is the square root of this variance.

1. Indexes and pixel values:

$$row\ number\ i \in [1 \ldots I]$$

$$column\ number\ j \in [1 \ldots J]$$

$$frame\ number\ k \in [1 \ldots K]$$

$$pixel\ value\ P(i,j,k)$$

2. Mean signal value of each pixel:

$$\mu(i,j) = \frac{1}{K}\sum_{k=1}^{K} P(i,j,k)$$

3. Mean signal value of each row:

$$\mu_{row}(i) = \frac{1}{J}\sum_{j=1}^{J} \mu(i,j)$$

4. Mean signal value of each column:

$$\mu_{col}(j) = \frac{1}{I}\sum_{i=1}^{I} \mu(i,j)$$

5. Mean signal value of each row of each frame:

$$\mu_{row}(i,k) = \frac{1}{J}\sum_{j=1}^{J} P(i,j,k)$$

6. Mean signal value of each column of each frame:

$$\mu_{col}(j,k) = \frac{1}{I}\sum_{i=1}^{I} P(i,j,k)$$

7. Mean signal value of all pixels local to a row:

$$\mu_{row\_local}(i) = \frac{1}{10}\sum_{u=i-5}^{i+4} \mu_{row}(u)$$

8. Mean signal value of all pixels local to a column:

$$\mu_{col\_local}(j) = \frac{1}{10}\sum_{v=j-5}^{j+4} \mu_{col}(v)$$

9. Mean signal value of all pixels:

$$\mu = \frac{1}{I \times J}\sum_{i=1}^{I}\left(\sum_{j=1}^{J} \mu(i,j)\right)$$

10. Variance of each pixel:

$$\sigma^2(i,j) = \frac{1}{K-1}\sum_{k=1}^{K}[P(i,j,k) - \mu(i,j)]^2$$

11. Variance of each row:

$$\sigma_{row\_temp}{}^2(i) = \frac{1}{K-1}\sum_{k=1}^{K}[\mu_{row}(i,k) - \mu_{row}(i)]^2$$

12. Variance of each column:

$$\sigma_{col\_temp}{}^2(j) = \frac{1}{K-1}\sum_{k=1}^{K}[\mu_{col}(j,k) - \mu_{col}(j)]^2$$

13. Mean variance of all pixels:

$$\sigma_{tot\_temp}{}^2 = \frac{1}{I \times J}\sum_{i=1}^{I}\left(\sum_{j=1}^{J}\sigma^2(i,j)\right)$$

14. Mean variance of all rows:

$$\sigma_{row\_temp}{}^2 = \frac{1}{I}\sum_{i=1}^{I}\sigma_{row\_temp}{}^2(i)$$

15. Mean variance of all columns:

$$\sigma_{col\_temp}{}^2 = \frac{1}{J}\sum_{j=1}^{J}\sigma_{col\_temp}{}^2(j)$$

Noise Measurement Results

The noise measurement results are formatted as tab-separated text data. There is a row of data for each color plane. The columns are as follows:

Signal: The average pixel value of the entire data set. The black level is subtracted.

$$Signal = \mu$$

RMS_Dyn: Temporal noise of all pixels. This includes pixel-wise, row-wise and column-wise dynamic noise, but not fixed pattern noise.

$$RMS_{Dyn} = \sigma_{tot\_temp}$$

Pix_Dyn: Isolate the pixel-wise temporal noise by removing the row-wise and column-wise noise.

$$PIX_{Dyn} = \sqrt{\sigma_{tot\_temp}{}^2 - \sigma_{row\_temp}{}^2 - \sigma_{col\_temp}{}^2}$$

FPN: All fixed pattern noise, including row-wise and column-wise.

$$FPN = \sqrt{\frac{1}{I \times J}\sum_{i=1}^{I}\left(\sum_{j=1}^{J}[\mu(i,j) - \mu]^2\right)}$$

Col_FPN: Column-wise fixed pattern noise only.

$$Col_{FPN} = \sqrt{\frac{1}{J}\sum_{j=1}^{J}[\mu_{col}(j) - \mu]^2}$$

ColLFPN: Also column-wise fixed pattern noise, but using only nearby columns for the reference signal level so that uneven illumination or lens shading does not get included as false noise level.

$$Col_{local\_FPN} = \sqrt{\frac{1}{J}\sum_{j=1}^{J}\left[\mu_{col}(j) - \mu_{col\_local}(j)\right]^2}$$

Row_FPN: Row-wise fixed pattern noise only.

$$Row_{FPN} = \sqrt{\frac{1}{I}\sum_{i=1}^{I}[\mu_{row}(i) - \mu]^2}$$

RowLFPN: Also row-wise fixed pattern noise, but using only nearby rows for the reference signal level so that uneven illumination or lens shading does not get included as false noise level.

$$Row_{local\_FPN} = \sqrt{\frac{1}{I}\sum_{i=1}^{I}\left[\mu_{row}(i) - \mu_{row\_local}(i)\right]^2}$$

Col_Dyn: Column-wise temporal noise.

$$Col_{Dyn} = \sigma_{col\_temp}$$

Row_Dyn: Row-wise temporal noise.

$$Row_{Dyn} = \sigma_{row\_temp}$$

Total: Combine all temporal noise and all fixed pattern noise to get total noise.

$$total = \sqrt{\sigma_{tot\_temp}{}^2 + FPN^2}$$

Signal to noise ratios (SNR) in dB are also calculated for each of the above noise values.

$$SNR_{\{noise\}} = 20\log_{10}\frac{\mu}{\{noise\}}$$

The EMVA 1288 SNR is calculated as the linear ratio of mean signal over total noise.

$$SNR_{EMVA1288} = \frac{\mu}{total}$$

Algorithms in DevWare

DevWare uses a ‘single-pass’ algorithm for calculating variances in the time domain. This means that it doesn’t need to store the frames, so any number of frames can be sampled and the amount of memory needed for the calculations is only proportional to the size of the ROI. However the algorithms are mathematically equivalent to the above formulas.

See: https://en.wikipedia.org/wiki/Algorithms_for_calculating_variance

Scripting Noise Measurement

The function of the Noise Measurement dialog can be scripted with DevWare Python. The ROI can be set with apbase.set_mouse_selection, and the other parameters can be set with apbase.setstate. Below is a minimal example Python code to do a noise measurement on raw data. The ROI is set to a 100x100 square in the center of the image. The results are printed to the Python Console window:

```
[Basic Noise Measurement]
import threading
import time
def measure():
    x1 = apbase.Camera().sensor.width // 2
    y1 = apbase.Camera().sensor.height // 2
    apbase.set_mouse_selection('rectangle', x1-50, y1-50, x1+49, y1+49)
    apbase.setstate("Noise Image Type", 0)
    apbase.setstate("Noise Frames", 50)
    apbase.setstate("Noise Defects", 0)
    apbase.setstate("Delay Noise Measurement", 0)

    apbase.setstate("Begin Noise Measurement", 1)
    while apbase.getstate("Begin Noise Measurement") > 0:
        time.sleep(1)
    results = apbase.getstate('Noise Measurement Results')
    print(results)

threading.Thread(target=measure).start()
```

The "Noise Image Type" parameter chooses between analyzing the raw data from the camera, or the processed RGB data output of the DevWare colorpipe. The possible values are: 0 = raw data as it came from the camera; 2 = RGB data after processing by the DevWare colorpipe; 3 = both raw and processed data.

The while loop has to run in a background thread because DevWare suspends processing images while scripts are running in the main thread.

Revision History

2017 October 17	Bill Dirks	First draft.