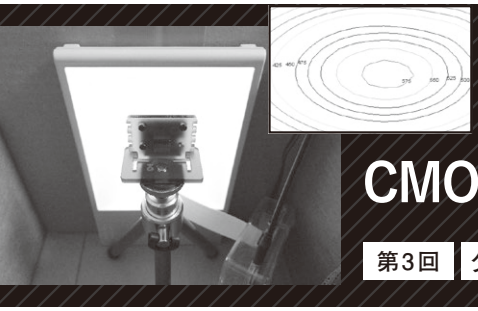

ダウンロード・データあります

ラズパイで体験！ CMOSイメージセンサ性能の測定評価

第3回 ダーク・ランダム・ノイズその3…ノイズの実測

米本 和也

いよいよRawデータをキャプチャしてダーク・ランダム・ノイズを算出できる段階になりました．ここまで用意周到にラズベリー・パイとカメラ・モジュールの設定ができていますので，後はpicameraモジュールを駆使してPythonスクリプトを作成するだけです．

大きな流れとしては，まずプレビューにより光が入らないような撮像条件になっていることを確かめ，ダーク・ランダム・ノイズの測定評価に適した条件である最大ISO感度，最短蓄積時間に設定し，Rawデータをキャプチャします．本連載の第2回で解説したようにフレーム間差分法を用いますので，キャプチャした2フレームのRawデータからダーク・ランダム・ノイズを算出し表示します．

ダーク・ランダム・ノイズを求めるPythonスクリプトの作成

リスト1にダーク・ランダム・ノイズを求めるPythonスクリプトを示します．

● プレビューから設定の固定まで

まず，モジュールのセクションでは入出力を扱うio，配列の計算を容易にするnumpyを取り入れます．スクリプトの中で複数回使われるカメラ設定の値として，ラズベリー・パイCamera V2の最大ISO感度値のISO，フレーム・レートfps，最小の蓄積時間Texp，画像のフォーマットに関係する値として画像サイズimsize，ダミーなし/ありのRawデータ行列サイズ，これらをそれぞれcrop，shapeとして値を記憶しておきます．

プレビューの部分はそのまま使い，プレビューを抜けた後，ISO感度，フレーム・レートの設定をします．カメラ・モジュール起動後の初期状態ではAE（自動露出）が働いているので，蓄積時間が明るさに応じて自動的に変化してしまいます．従って，蓄積時間を固定するためにexposure_modeをoff設定とします．

この場合，蓄積時間を決めるのはshutter_speedなので，あらかじめ決めたTexpの値で指定します．これらの設定が完了していることの確認を表示のセクションで行います．ここで蓄積時間が1μsになっていないことに気がつきますが，それがラズベリー・パイCamera V2の設定ができる実際の最小値です．ここまででカメラ・モジュールがダーク・ランダム・ノイズを求めるための状態になりました．次はキャプチャしてデータ処理します．

● RawデータからRaw画像への変換を関数化する

前回，Rawデータのフォーマットを詳しく解説しましたが，そのフォーマットに従ってRawデータをRaw画像の配列に変換する操作は，どの性能項目についても同じように利用されます．従って，Strm2Imgという関数として定義しておきます．

この関数は，キャプチャ・コマンドで得られたJPEGのバイナリ・ストリームのヘッダを除いたメタデータ部分について，いったんは符号なし8ビットの1次元配列dataとした後，shapeとcropの値に従って2次元配列に整えます．

この状態では，4画素が5バイト構成のまま整列していません．そこで新たに作った2次元NumPy配列imgに，4画素分の5バイト単位で，2進数を2ビット上位にシフトすることで4倍してから，5バイト中の先頭から4バイトを4画素として代入してます．続いて5バイト目に詰め込まれている4画素分の下位2ビットを，それぞれ空いている下位2ビットへ埋める操作をします．これで1画素10ビットの値がimgに整列して格納されます．後は配列imgを関数から返します．

● 画像のキャプチャからノイズを求める処理まで

ダーク・ランダム・ノイズを求めるに当たって，フレーム間差分を用いることを前回解説しました．そこで，2フレームをキャプチャしてそれぞれをraw1，raw2としておきます．最後に固定パターン・ノイズ（FPN：Fixed Pattern Noise）を取り除く操作であるraw1とraw2の差分をとり，$1/\sqrt{2}$倍した画像をrawdrnとします．

これの画角全体にわたるゆらぎ，つまり標準偏差を計算すればよいのですが，その際ディジタル値[DN]

注1：イメージセンサの分野ではディジタル値と電子数の単位をそれぞれ[DN]と[e]で表記する．

本記事を試す際には次のOSを推奨します．
Raspberry Pi OS（レガシー），2023年12月5日，32ビット，カーネルバージョン：6.1，Debian バージョン：11（ブルズアイ）