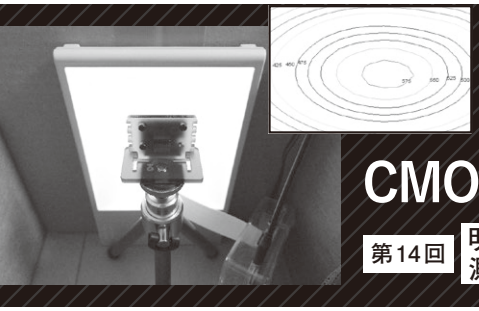

ラズパイで体験！

CMOSイメージセンサ性能の測定評価

第14回 明るい被写体の周りに生じる水平スミア（後編）測定のためのPythonスクリプト

米本 和也

水平スミアは，被写体中の画素信号が飽和する明るい部分に対して，その横にだけうっすらとした帯状のコントラストの異なる部分が発生してしまう現象です．ストリーキングとも呼ばれます．特に明るい被写体の周りが，ほぼ真っ暗な状態で目立つ性質があります．

前回は水平スミアの特徴や性能基準などを解説したあと，ラズベリー・パイのカメラ・モジュールPiCamera V1，V2（以降，V1，V2）を用いて実際に測定を行い，結果を分析しました．今回は測定に利用したPythonスクリプトについて，解説します．

測定手順

● ステップ1：カメラ・モジュールの位置を調整

ウィンドウ・チャートの窓が水平画角の約10％を占めるものを選択し，カメラ・モジュールの位置を調整して窓が画角中央に来るようにします．

● ステップ2：信号が飽和している状況を作る

ウィンドウ・チャートの窓を撮像した画素からの信号が飽和している状況を作ります．それにはイメージセンサの内部ゲインを最小にして飽和の2倍以上が得られる露光時間に調整する必要があります．この露光時間は，プレビュー時の窓の信号量とAE露光時間から算出します．従って，プレビューの終了後，水平スミアを測定するための画像をキャプチャする前に，適切な露光時間を見積もる操作をします．

● ステップ3：ランダム・ノイズを抑圧するため多数のフレームで平均化

キャプチャでは微小な水平スミアの信号を取るのに，イメージセンサのゲインを最大にし，かつランダム・ノイズの影響を抑えるのに多フレーム平均化処理を用います．

● ステップ4：適切な列数の列信号を水平方向に加算（平均）する

さらに各列の信号を水平方向に加算平均します．ただし，発生原因の一例で示したメカニズム以外の原因もあり，水平スミアの大きさは水平方向に一様ではなく，分布を持つ場合も少なくありません．従って，水平方向の加算平均は列数を幾つかのブロックに分割して個別の水平スミア抑圧比を算出することになります．

● ステップ5：レンズ・フレアを補正する

窓の信号は画素が飽和状態でゲインが最大の設定だとレンズ・フレアなどによって水平スミアが現れている部分の信号がレンズ・フレア成分で浮き上がります．水平スミアをこの浮き上がりから分離する（補正する）処理も必要になります．

Pythonスクリプト

上記手順を満たすようにPythonスクリプトを作成していきます．ウィンドウ・チャートの窓のサイズと撮像面内位置は都度変わるのと，窓部の信号を飽和に合わせるために蓄積時間を設定しなければなりません．従って，プレビューでウィンドウ・チャートに画角を合わせたら，すぐに「キャプチャ」→「処理」というわけにはいかず，**図1**のような測定手順を踏むようにスクリプトを作成します．

この測定手順に従い作成したPythonスクリプトの一例を**リスト1**に示します．また今回の水平スミアはPiCamera V1とV2を比較することにしましたが，スクリプトは共通です．このため，前回でカメラ・モジュールを動作させるモジュールがPicameraから

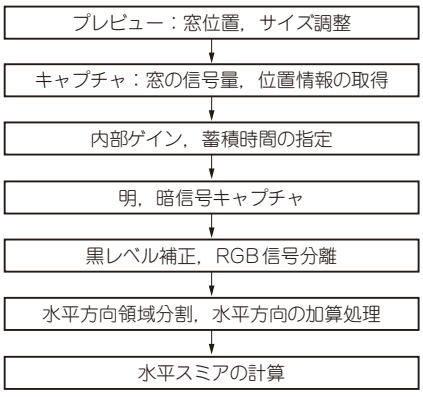

図1　水平スミアを数値として得るまでの流れ

本記事は次のOSで動作を確認しています．
Raspberry Pi OS（64-bit），2024年11月19日，Debian Bookworm