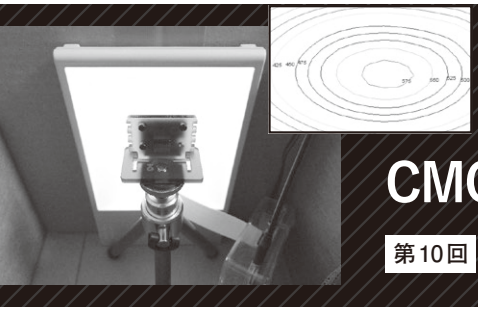

ダウンロード・データあります

ラズパイで体験！ CMOSイメージセンサ性能の測定評価

第10回 PRNUその2…実測

米本 和也

前回は，感度の不均一性でもっとも重要なPRNU（Photo Response Non-Uniformity）について，発生原因，測定の環境と方法からデータの扱い方まで詳しく解説しました．今回は実際にラズベリー・パイとPiCamera V1（以降，V1），PiCamera V2（以降，V2）を使って画像をキャプチャし，データ処理をしてPRNUを求めてみます．

リスト1　PRNUを測定するPythonスクリプト

```
##### モジュール #####
import io
import time
import numpy as np
import gc
import picamera
camera = picamera.PiCamera()

##### カメラ設定の値 #####
ISOpr = 54   # プレビュー用ISO感度 カメラ・モジュールV2
ISO = 54     # キャプチャ用ISO感度 （最小値） カメラ・モジュールV2
Fps = 10     # フレーム・レート [fps]

##### キャプチャの条件設定 #####
Nexp = 500    # 明信号のキャプチャフレーム枚数
Ndrk = 100    # 暗信号のキャプチャフレーム枚数
Rexp = 2      # 蓄積時間の対AE蓄積時間比率
ROIsize = 20 # 信号を取得する画角中央の正方画素ブロックの縦横サイズ
### キャプチャ・フレーム数（Nexp，Ndrk）と黒レベル補正（ShftDrk）を，
          平均化，メディアン・フィルタ計算が整数で実行できるように修正
Nexp = int(2**np.ceil(np.log2(Nexp)))
                        # 明信号のキャプチャフレーム枚数 （2^N化）
Ndrk = int(2**np.ceil(np.log2(Ndrk)))
                        # 暗信号のキャプチャフレーム枚数 （2^N化）
ShftDrk = int(np.log2(Nexp/Ndrk))  # 暗信号のビットシフト数
### メディアン・フィルタのカーネル・サイズ
KerSize = 5

##### 画像のフォーマット関係の値 #####
ImSize = (2464, 3280)  # 画像のサイズ （行数，列数） V2
Dummy = (16, 28)       # Dummy （行数，列Byte数） V2
ByPx = (5, 4)          # Byte per Pixel （Byte，Pixel）
Crop = (ImSize[0], int(ImSize[1]/4*5))
                        # Rawデータの行列サイズ，ダミーなし
Shape = (Crop[0]+Dummy[0], Crop[1]+Dummy[1])
                        # Rawデータの行列サイズ，ダミーあり
Shft = int(8/ByPx[1])  # ビットシフト幅
### 画素の色配置
Col = [["B", "Gb"], ["Gr", "R"]]                    # V2
ByrSize = (int(ImSize[0]/len(Col)),
           int(ImSize[1]/len(Col[0]))) # 色別の画像サイズ

##### 開始 #####
print("[ PRNU ]")
print("  イメージセンサ =", camera.exif_tags[
              'IFD0.Model'])  # イメージセンサのタイプを表示

##### プレビューのカメラ設定（フレーム・レート，ISO感度） #####
camera.framerate = Fps
                        # フレーム・レートは標準設定値30fpsのまま
camera.iso = ISOpr    # ISO感度設定

##### プレビュー #####
camera.resolution = (int(ImSize[1]/2), int(
       ImSize[0]/2)) # HVとも半分（GPUメモリ・サイズ制限のため）
camera.preview_fullscreen = False     # プレビューの画角制限
camera.preview_window = (0, 0, 640, 480)
                                      # プレビュー画角の大きさ
camera.start_preview()            # プレビュースタート
time.sleep(3)                     # 蓄積時間の安定化時間
print("\nプレビュー")
print("  フレームレート =", camera.framerate, " [fps]")
print("  ISO感度 =", camera.iso)
print("  AE蓄積時間 =", camera.exposure_speed, " [us]")
input('  改行を押してキャプチャ開始 >')
camera.stop_preview()             # プレビューストップ

##### 蓄積時間の自動調整（AE）解除とISO感度の設定およびその表示 #####
camera.shutter_speed = camera.exposure_speed
                                    # AE状態蓄積時間のコピー
camera.exposure_mode = 'off'         # 蓄積時間の固定化
AutoExpSpeed = camera.shutter_speed # AE状態蓄積時間の記憶
camera.iso = ISO                     # ISO感度
print("\nキャプチャ開始")
print("  フレームレート =", camera.framerate, " [fps]")
print("  ISO感度 =", camera.iso)
print("  AE蓄積時間 =", camera.shutter_speed, " [us]")

##### 関数定義：キャプチャのバイナリストリームを
                                    2次元配列Rawデータに変換 #####
def Strm2Img(strm):
    data = strm.getvalue()[-int(Shape[0]*Shape[1]):]
    data = np.frombuffer(data, dtype=np.uint8)
    data = data.reshape((Shape[0], Shape[1]))
                                      [:Crop[0], :Crop[1]]
    data = data.astype(np.uint16) << Shft
    img = np.zeros(ImSize, dtype=np.uint16)
    for byte in range(ByPx[1]):
        img[:,byte::ByPx[1]] = data[:,byte::ByPx[0]]
          | \((data[:,ByPx[1]::ByPx[0]]>>((byte+1)*2))
                                          & (2**Shft-1))
    return img

##### メディアン・フィルタ関数 （Ksize: 3以上の奇数）#####
### メモリが限られている場合
                    （行単位のメディアン・フィルタ処理でメモリを節約）
def ImgMedFilterLbL(Img, Isize, Ksize):
    ImgMd = np.zeros(Isize, dtype=np.int32)
    LineMd = np.zeros((Ksize**2, Isize[1]),
                                          dtype=np.int32)
    for i in range(Isize[0]):
        for j in range(Ksize):
```

本記事は次のOSで動作を確認しています．
Raspberry Pi OS（Legacy），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（bullseye）