ADVACAM
Imaging the Unseen

Noisy pixels masking

Online version: https://wiki.advacam.cz/wiki/Noisy_pixels_masking

Contents

The detector response can be changed due to the environment conditions. One of the results might be appearing of so-called noisy pixels. These pixels produces artificial signal which is not generated by radiation interacting in the sensor volume. Generally for most of the measurements, these pixels should be removed before the measurement. Fro these purposes, the method below and its implementation into Pixet C++ API was done.

Method description

Two types of noisy pixels are defined: saturated and oscillating.

The saturated pixels produces signals constantly and is always above the threshold. These pixels can be easily observed in the ToT frame measurement when they have usually value equal to the amount of counts in that given acquisition time (for example if t_acq = 0.1 ms with 40 MHz clock, then their ToT value is close to 40 000 000 * 0.1/1000 = 4000 ToT counts).

The oscillating pixels are very close to threshold and produces their signals randomly but steadily based on very small changes in the conditions. These pixels can be observed in the count measurements when they produces large values of counts.

In the proposed methodology, the saturation pixels are detected in ToT measurement. An additional scaling factor is introduced to defined the minimal limit for a pixel to be considered as saturated one. This limits defines portion of the expected ToT values for the given acquisition time of the measurement. The default value of this parameter is 50 which means that every pixels in 0.1 ms measurement whose ToT value is more that (50/100) * (40 000 000 * 0.1/1000) = 2000 are considered to be saturated.

The oscillating pixels are detected in count measurement and minimal value to marked pixels as oscillating is 50 counts. In other words, all pixels with count values more or equal to 50 are oscillating pixels.

All oscillating and saturated pixels are considered as noisy pixels and this methods would mask them.

This method itself is very suitable for silicon sensors. The CdTe sensors introduces new difficulties and are discussed in the section below.

CdTe Sensors

In the case of CdTe sensors, the technique proposed above could be accompanied with lowering of threshold or with complete verification of equalization. This part is to be tested and it is NOT implemented in the software solution below.

Implementation in Pixet C API

The function definition can be found below:

```
// Finds noisy pixels and it can optionally masked them. Designed for TPX3 devices in
environment WITHOUT radiation background.
// [in] deviceIndex - index of the device (indexing starting from 0)
// [in] limitNoisy - limit to mask noisy pixels. It goes from 1 to 1022, where 1 is the
most strongest masking and 1022 is the weakest.
```

```
// [in] limitSatur - limit to mask saturated pixels. It goes from 0 to 100, where 0 is the
most strongest masking and 100 is the weakest.
// [in] doMaskNoisyPixels - control to mask noisy pixels within the process. Default is
false alias do not mask the found pixels.
// [in] matrixSize - size of the matrix.
// [in/out] noisyPixelsMatrix - mask/positions of noisy pixels.
PXCAPI int pxcFindNoisyPixels(unsigned deviceIndex, double limitNoisy = 50, double
limitSatur = 50, bool doMaskNoisyPixels = false,
                                unsigned* noisyPixelsMatrix = nullptr, unsigned matrixSize
= 65536);
```

More detailed explanation of the individual parameters influencing the masking performance can be found in the section above.

Example of usage:

```
#include "pxcapi.h"
#include <string>

int main()
{
    int rc=  0;

    rc = pxcInitialize();

    double limitNoisy = 10;
    double limitSatur = 50;
    unsigned matrixSize = 65536;
    unsigned int noisyPixelsMatrix[65536];
    rc = pxcFindNoisyPixels(0, limitNoisy, limitSatur, true, noisyPixelsMatrix,
matrixSize);
    printf("------------------------------\n");
    printf("Found noisy pixels\n");
    for (int i = 0; i < matrixSize; ++i)
    {
        if(noisyPixelsMatrix[i] == 1)
            printf("%d\t%d\n", i, noisyPixelsMatrix[i] );
    }

    printf("------------------------------\n");
    printf("Additional frame measurement to check hit pixels\n");

    double dataIToT[65536];
    unsigned short dataCount[65536];
```

```
    rc = pxcMeasureSingleFrameTpx3(0, 0.1, dataIToT, dataCount, &matrixSize, PXC_TRG_NO);
    for (int i = 0; i < matrixSize; ++i)
    {
        if(dataIToT[i] >= 1 or dataCount[i] >= 1)
            printf("%d\t%.0f\t%d\n",i, dataIToT[i], dataCount[i]);
    }
    printf("------------------------------\n");
    printf("Additional measurement in data driven\n");

    rc = pxcSetTimepix3Mode(0, PXC_TPX3_OPM_TOATOT);
    rc = pxcMeasureTpx3DataDrivenMode(0, 1, "data.t3pa");

    printf("------------------------------\n");

    rc = pxcExit();
}
```