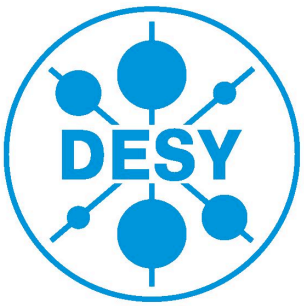

Testing the Medipix3 photon-counting detector

Thomas Dobbelaere, Ghent University, Belgium

September 8, 2011

Abstract

The performance of the Medipix3 hybrid silicon pixel detector is characterised at different temperatures and settings. Images are taken using the built-in test pulse feature to simulate impinging energetic photons. The result is globally evaluated by counting the number of (in)correctly working pixels. Additionally, a per-pixel analysis is done in order to study the behaviour of each pixel and find parameters such as the cut-off threshold and the noise edge. Measurements of the voltages of the built-in DACs show an important temperature dependency, illustrating the need to equalise the pixel array after major changes. Some demonstration X-ray images are provided as well.

Contents

1 Introduction

High-energy photon detection is an important aspect of several experimental physics setups, e.g. imaging, scattering, spectroscopy and tomography, using either an X-ray tube or a synchrotron beamline. The function of the detector is to retrieve as much information as possible about the impinging photons. This would imply measuring the intensity (number of photons per second), the energy (wavelength), the position, the angle, the arrival time, and the polarisation[1]. Unfortunately, the ideal detector does not exist, so one has to compromise. The CCD[1] is a popular type of semiconductor detector which can offer a high spatial resolution, but only integrates the total intensity and offers no spectral information. Noise and leakage current are integrated along with the signal during exposure, and for long exposures (low intensity) this can be problematic.

Hybrid pixel detectors aim to improve on some of these limitations. They were originally developed as detectors for particle physics at CERN, but proved to be very useful for X-ray imaging as well. They feature two separate layers: the top layer is photosensitive, while the bottom layer contains per-pixel processing circuitry. An array of bump bonds provides electrical connections between both layers. This approach has several advantages: the whole area is available for photon capture, while there is also space for per-pixel readout and processing electronics. The end result is a photon-counting, energy-discriminating design with good spatial resolution. The measurements in this report were carried out using the Medipix3, a state-of-the-art chip designed by the Medipix collaboration with lots of advanced features [2].

2 Hardware

2.1 Detector structure

In Figure 1 the Medipix3 sensor arrangement is shown schematically. The bulk of the top layer consists of n-type silicon, of which the upper side is metallised with aluminium. The opposite side is implanted with p-type material at regular intervals, forming an array of pn-junctions (one for every pixel). In normal operation, a positive voltage of about 100V is applied to the n-type material with respect to the p-type. In other words, the junctions are reverse biased; this creates depletion regions in which no current (aside from leakage) normally flows. For each pixel, this region fulfills the role of a photon-detecting medium. When an energetic photon impinges, it can cause a high-energy electron to be released in accordance with the photoelectric effect. In turn, this photoelectron excites local electron-hole pairs. The existing electric field causes drift of the mobile charges, and a current flows in the corresponding pn-junction. The produced charge is proportional to the energy of the impinging photon.

The top layer is connected to the Medipix3 integrated circuit using an array of solder (or indium) bump bonds, one for each pixel. Thus, any current produced in a pn-junction in

[1] Charge-coupled device.

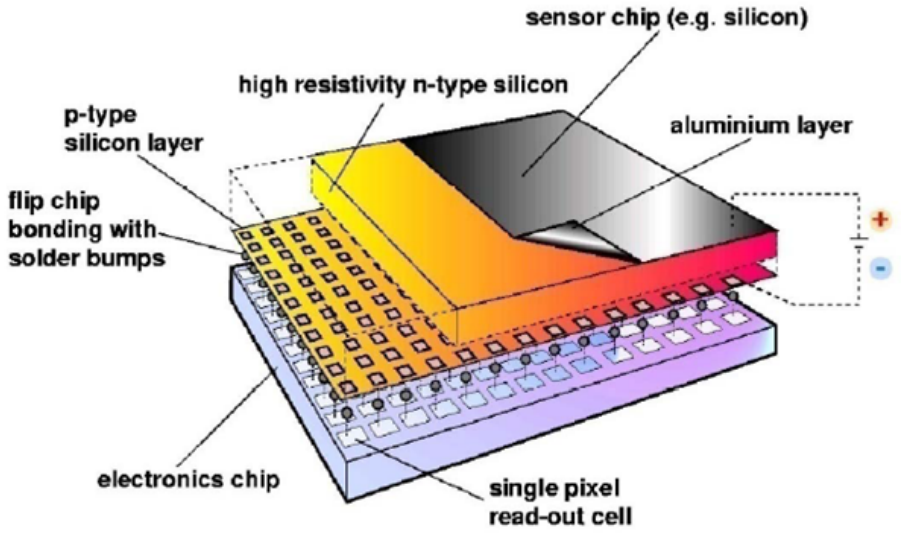

Figure 1: Illustration of the Medipix sensor, showing the layered design.

the top layer will be conducted to the corresponding electrode on top of the electronics chip. Figure 2 shows this arrangement of electrodes on a bare Medipix3 chip. Every pixel contains CMOS circuitry that will amplify, shape, discriminate, and count the electrical pulses it receives. The next paragraph will elaborate on this.

2.2 Signal processing

The function of each pixel is to analyse the electrical impulse it receives from the pn-junction in order to reconstruct information about the photon. A block diagram of a single pixel readout cell is shown in figure 3. The charge received on the input pad first goes through amplification and shaping. The resulting signal is a voltage that varies in time according to a peak shape; the height of the peak is proportional to the collected charge, which is in itself proportional to the energy of the captured photon. It is then simply a matter of comparing the signal to a threshold voltage to get a (digital) photon count. By using both an upper and lower threshold, one can even count photons inside a certain energy window. This idea is illustrated in figure 4.

2.3 Configuration

2.3.1 Test pulses

In addition to receiving charge from the input pad, one can configure each pixel readout cell to be injected with a charge from the 4.8 fF capacitance C_{TEST} (figure 3). This feature is very useful for test purposes: it simulates charge from incoming photons without actually having to expose the sensor to a source. Several parameters, including the injected charge (energy) and the number of pulses, can be configured by the user.

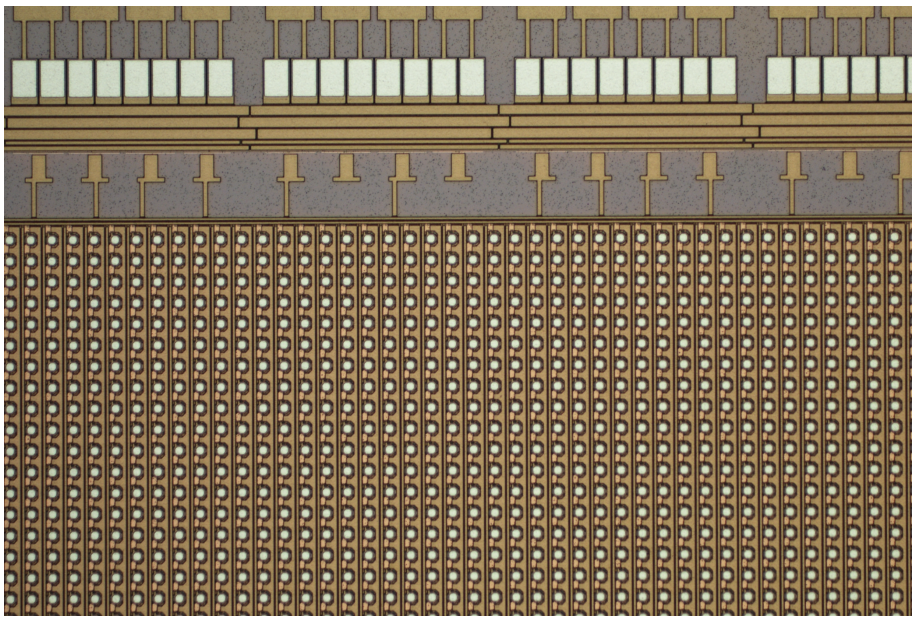

Figure 2: Microscope view of the top layer of a bare Medipix3 chip. One pixel measures 55×55 μm; the whole chip contains 256×256 pixels.

2.3.2 Equalisation

Not all pixels of the Medipix3 behave in exactly the same way. Due to limitations within the production process, transistor gains are often mismatched. This causes certain pixels to be more sensitive than others. In other words, the threshold would be different for every pixel. Without correcting for this, the image would be very noisy. Luckily, each pixel in the Medipix3 has its own adjustment circuitry to compensate. The idea is that one can perform an algorithm which determines – for each pixel – a correction value which brings it as close as possible to a global behaviour. This procedure is called an *equalisation*. It can be easily performed by the Pixelman software (see paragraph 3.1) [5]. Once the equalisation file for a certain chip is calculated, it can be saved and re-loaded the next time, unless the chip has been exposed to important changes (e.g. temperature).

2.3.3 DACs

The Medipix3 features several internal DACs[2]. The most commonly used one is the global threshold DAC, also named Threshold0 in Pixelman or abbreviated as THL. It sets the lower threshold for counting a pulse, as shown in figure 4. It is a 9-bit register, so its value ranges from 0 to 511. Setting THL to very low values will result in false counts because of noise, while setting THL to a high value will reject lower-energy photons (or test pulses). The Medipix3 also features an upper threshold, but this feature will not be used here.

[2]Digital-to-analog converters.

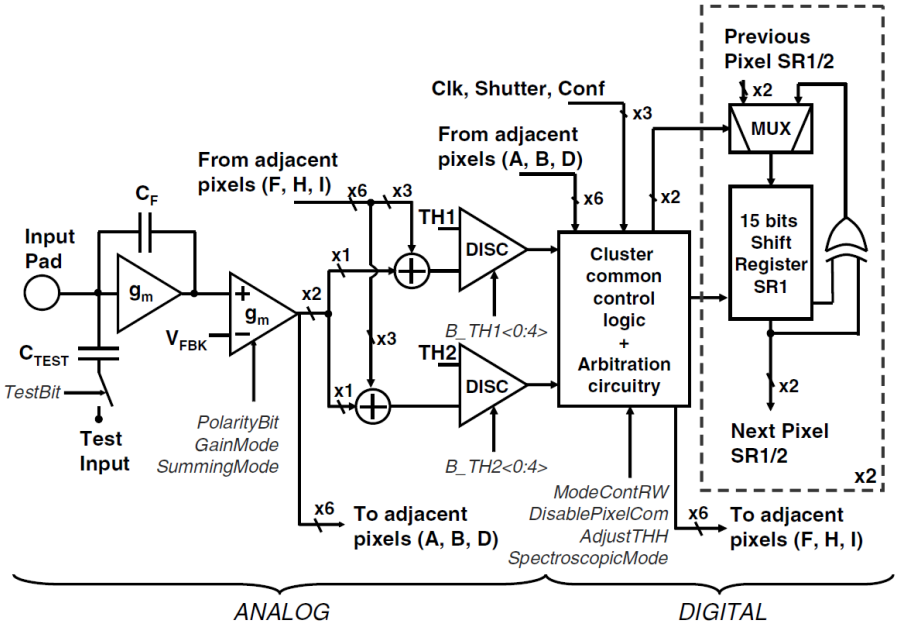

Figure 3: Block diagram of a single Medipix3 pixel cell.[2]

Other important DACs include ThresholdN and DACPixel (both 8 bits and global), together with the per-pixel adjustment settings B_TH<4> (1 bit) and B_TH<0:3> (4 bits). They are used for equalising the pixel array (see paragraph 2.3.2 for the concept behind this). B_TH<0:3> shifts the behaviour of a pixel in one direction, with a global multiplication factor set by DACPixel. It effectively works as a local *addition* to the global threshold value. If the pixel needs a shift in the other direction, the bit B_TH<4> can be set. This causes a fixed value, determined by ThresholdN as a multiplication factor, to be effectively *subtracted* from the global threshold.

Some DAC's influence the pulse amplification, shaping & discrimination process. These ones have the – mostly cryptic – names 'Preamp', 'IKrum', 'Shaper', 'Disc', 'Disc_LS', 'RPZ', 'GND', 'FBK', and 'CAS'. The exact role they play in the electronics will not be discussed here, but more information (from a 'circuit design' point of view) can be found in reference [4].

2.4 Readout system

In order to actually use the chip, one needs to interface it to a computer. To this end, some electronics are needed. The total arrangement consists of a chip board, an interface board, and a USB readout board. The Medipix3 chip is bonded to the chip board using bare wire bonds. Care has to be taken to avoid damaging them. The USB readout board enables one to simply plug it into a free USB port and start using the sensor with suitable software, e.g. Pixelman (3.1). The high voltage (100V) used as reverse-bias for the top layer is also generated by this board.

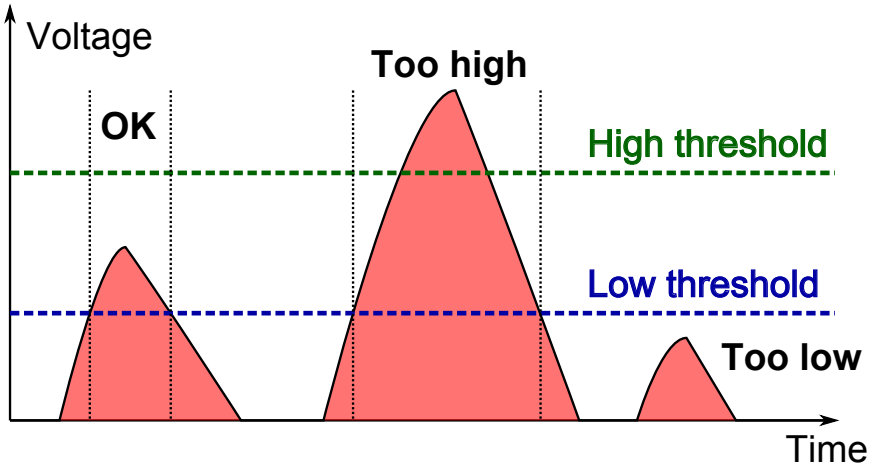

Figure 4: Illustration of photon (electrical pulse) counting by threshold discrimination.

3 Software

3.1 Pixelman

Pixelman v2.1.0 (July 4th 2011) provides a graphical interface for working with the sensor. It is designed to be used with the USB readout system, as discussed in 2.4. It runs on Windows and offers a (moderately) user-friendly way to perform image acquisitions. It exposes a lot of configuration settings, e.g. DAC settings, test pulse parameters, etc. Additionally, it has built-in routines to perform automated measurements and save the data to ASCII files. For instance, the "DAC Scan" feature scans a range of values for a particular DAC and saves the acquired image for each step. The software can be freely downloaded online [6].

3.2 IDL

In order to read and analyse the ASCII data files created by Pixelman, some scripts were written in IDL 8. This is a programming language focused on quick, array-oriented data processing; it is in this regard quite similar to Matlab. The built-in routines make it easy to perform tasks such as importing files, making plots and histograms, curve-fitting, and displaying image data. Almost all plots in this report were made in this way. Some example code (used for reading in the data files) can be found below.

```
PRO THLscanTestpulses
; Define the location and the name of the numbered ASCII data files
directory = 'D:\Medipix3\Equalized temperature dependency\-20\TP\'
fileprefix = 'DACScan100TP_Threshold0_'
filesuffix = '.txt'
; Define the location of the ASCII file template
template_path = '\\win.desy.de\home\tfdobbel\My Documents\IDLWorkspace8\
    Default\pixelmantemplate.sav'
```

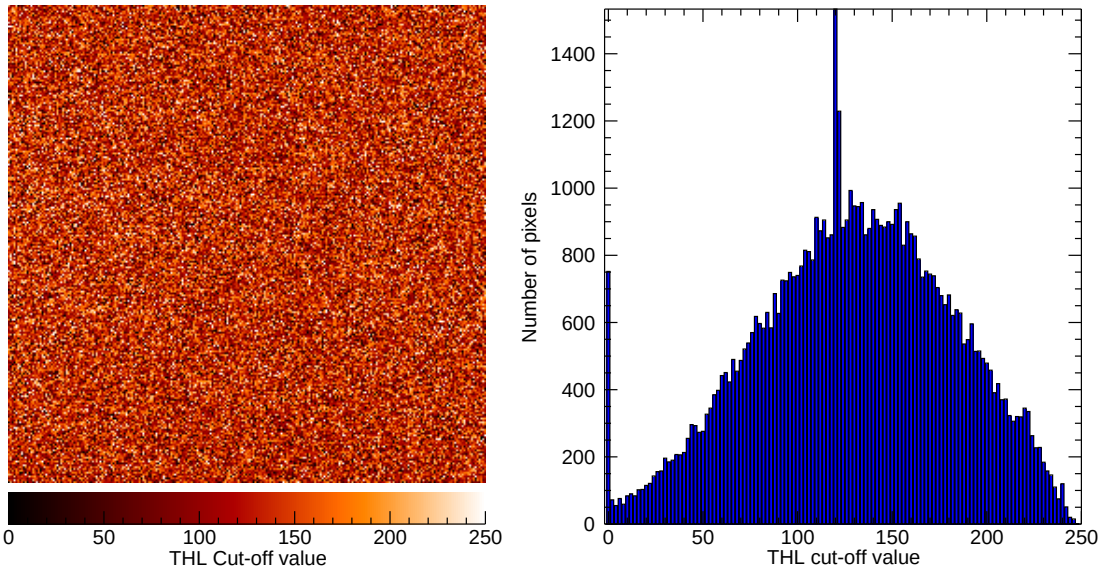

Figure 5: Image and histogram of the cut-off values determined for each pixel, before equalisation. Measurement conditions: −20 °C, 100 test pulses.

```
; Load the template file to a variable called 'pixelmantemplate'
restore, template_path
; Minimum and maximum index
THLmin = 0
THLmax = 150
THLsteps = THLmax-THLmin+1
defsysv, '!number_of_testpulses', 100
THLvector = indgen(THLsteps)+THLmin ; Create a vector counting from THLmin
    to THLmax
counts_total = lonarr(THLsteps) ; Create a vector storing the total pixel
    value count for each threshold level
counts_nonzero = lonarr(THLsteps) ; Create a vector storing the count of
    nonzero pixels for each threshold level
inputimage = intarr(256,256,THLsteps) ; This three-dimensional array will
    store the 256*256 pixel image for every threshold level
maxTHLdigits = strlen(strtrim(THLmax, 2)) ; Get the maximum number of
    digits that the THL index will contain
; The inputimage-array is saved on disk to avoid having to load the ASCII
    files again and again (which is slow).
if file_test(directory+'inputimage.sav') eq 0 then begin ; If not found,
    it's the first time the code is running, so read the ASCII anyway.
  print, 'File '+directory+'inputimage.sav'+' not found. Calculating...'
  for k = THLmin, THLmax do begin ; Load the image files into the array "
      inputimage"
    kstring = string(k, format = '(I0'+strtrim(maxTHLdigits,2)+')') ;
        Convert the index k to a string, padding with leading zeros to get
        the right number of digits
    filename = directory+fileprefix+kstring+filesuffix ; Create the path
        to the current data file
```

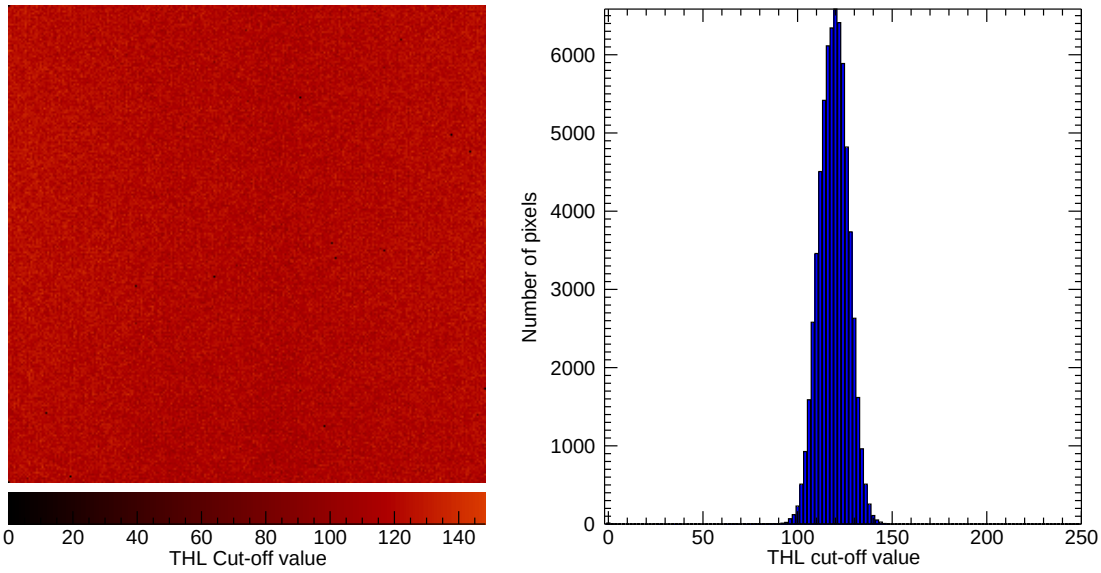

Figure 6: Image and histogram of the cut-off values determined for each pixel, after equalisation. Measurement conditions: −20 °C, 100 test pulses.

```
    inputdatafile = read_ascii(filename, template = pixelmantemplate) ;
        Read the ASCII data
    inputimage[*,*,k] = inputdatafile.pixels ; Get a 256x256 integer array
        for every threshold value
  endfor
  save, inputimage, filename = directory+'inputimage.sav' ; Save the
      inputimage-array to disk in a binary file
  print, 'Done.'
endif else begin ; If the file is found, just load it
  print, 'Found '+directory+'inputimage.sav'+'. Loading...'
  restore, directory+'inputimage.sav'
  print, 'Done.'
endelse
im1 = image((inputimage[*,*,60] < 150), rgb_table=3, min_value=0,
    max_value=150, dimensions = [1024,1024], position = [0.25, 0.25, 0.75,
    0.75]) ; Display the image
cb1 = colorbar(target=im1, title = 'Pixel value (counts)', position =
    [0.25, 0.20, 0.75, 0.23]) ; Display a colour bar
im1.save, '\\win.desy.de\home\tfdobbel\My Documents\Student2011_Medipix3\
    LaTeX report\pics\image_thl60_eq_min20.pdf', /centimeters, width = 40,
    height = 40, page_size = [19, 21.5], ymargin = -5 ; Save the image to a
     pdf, to be used in the LaTeX report
END
```

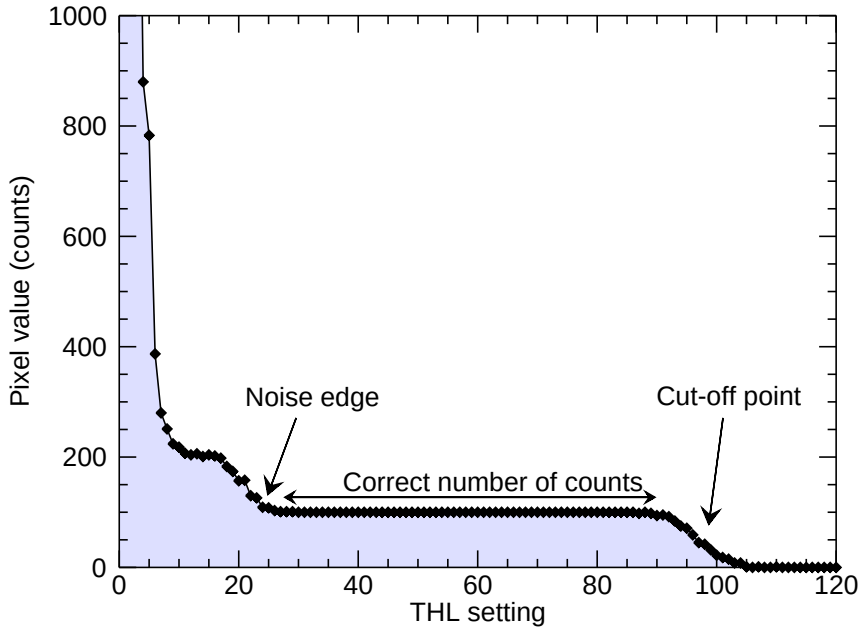

Figure 7: Example plot of the behaviour of one pixel under a changing threshold value. The data points are for the pixel with coordinates (50,50), measured at 20 °C. 100 test pulses were applied.

4 Measurements

4.1 Noise edge and cut-off point

Each pixel will only count photons (or test pulses) correctly if the threshold value is set within a certain range. This can be investigated by scanning over a range of THL settings while observing the number of test pulses counted by a pixel. The result is depicted in figure 7. There are three distinct regions: at low THL values, the pixel returns very high counts due to noise. The maximum THL value at which this happens will be referred to as the *noise edge*. When THL is set above the noise edge, the correct number of counts is reported. By increasing THL further, one encounters a *cut-off point*: beyond this THL value, the pixel value will always be zero. This is due to the threshold being above the pulse height. Of course, this position is dependent on the photon energy (or test pulse charge).

By determining the noise edge and cut-off point for every pixel, the images and histograms in figures 5, 6, 8 and 9 are obtained. They illustrate the effect of equalisation: each pixel is adjusted to get it closer to the centre of the distribution.

4.1.1 Cut-off error function

As observed in figure 7, the transition from the correct count to a zero count is not sudden. It rather follows an s-shaped curve, known as an *error function* or an integrated

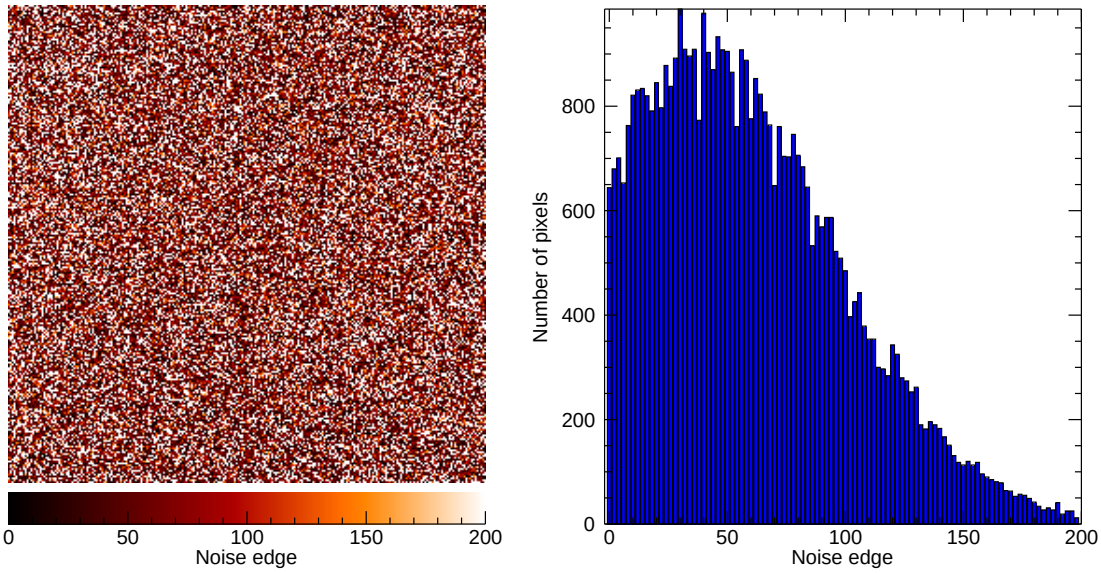

Figure 8: Image and histogram of the noise edge determined for each pixel, before equalisation. Measurement conditions: −20 °C, 100 test pulses.

Gaussian. In mathematical terms, if $F(\mu, \sigma, x)$ is a Gaussian probability distribution in x with mean μ and standard deviation σ, the error function becomes:

$$\mathrm{erf}(x) = \int_{x}^{+\infty} F(t, \mu, \sigma) dt$$

This indicates that there is some (Gaussian) noise, represented by the parameter σ, involved in the pulse discrimination process. If σ were zero, then the error function would become a step function with the step located at the threshold value μ.

The two free parameters μ and σ can be determined for each pixel by curve-fitting in IDL. Sigma then gives an indication of noise, while μ will be referred to as the cut-off value (i.e. the threshold value where 50% of the counts are registered). The result of such a curve-fit (for one pixel) is shown in figure 10. The following IDL code was written to curve-fit each pixel and store the found parameters into arrays:

```
THL_mean_image = fltarr(256,256) ; This array will hold the fitted 'mean'
    value of the error curve for every pixel.
THL_stdev_image = fltarr(256,256) ; This array will hold the fitted '
    standard deviation' (noise) value of the error curve for every pixel.
for pixel_x=0,255 do begin ; Loop over all pixels along the x axis
  for pixel_y=0,255 do begin ; (Inner) loop over all pixels along the y
      axis
    inputpixel = REFORM(inputimage[pixel_x, pixel_y, *]) ; Vector
        containing the pixel value for different THL steps (for the
        selected pixel)
    excess_counts = where(inputpixel gt !number_of_testpulses) ; Get all
        indices of pixel values exceeding the number of test pixels (due to
```

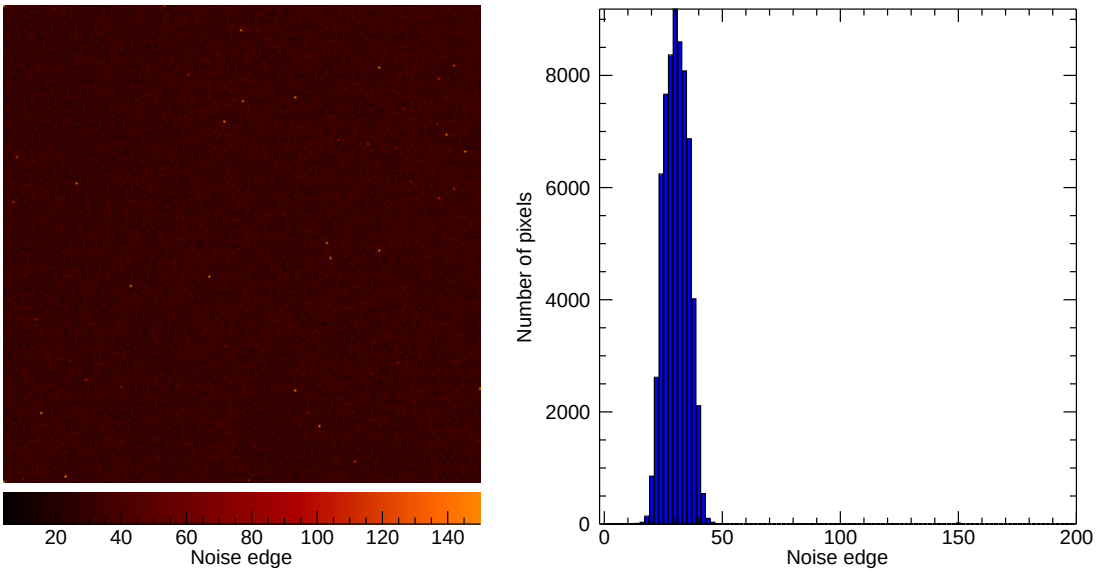

Figure 9: Image and histogram of the noise edge determined for each pixel, after equalisation. Measurement conditions: −20 °C, 100 test pulses.

```
        a THL value that is too low)
    weights = make_array(size(inputpixel, /dimensions), value=1.0) ;
        Create a "weights" array with length equal to the number of THL
        steps
    weights[excess_counts] = 0 ; Ignore excess counts (low THL) by setting
        the weight of them to zero - otherwise, it might confuse the curve
        fitter
    A = [120, 4] ; Initial parameter values [mean, stdev]- it should be a
        reasonably good guess, otherwise the curve fitter can get lost
    fit_y_val = CURVEFIT(THLvector, inputpixel, weights, A, FUNCTION_NAME
        = 'THLscanGaussfit', /noderivative, itmax = 50, status = cfstatus,
        /double) ; Attempt a curve fit
    if cfstatus ne 0 then print, pixel_x, pixel_y; Print the pixels where
        the curve fit failed, might be useful to inspect them later
    THL_mean_image[pixel_x, pixel_y] = A[0] ; Copy the found parameters
        into the appropriate array
    THL_stdev_image[pixel_x, pixel_y] = A[1]
  end
  print, 'Progress: line '+strtrim(pixel_x,2)+'/255'
end
```

4.2 Temperature dependency

To test the influence of ambient temperature on the sensor output, the sensor and its USB-interface were put into a climate chamber (Vötsch model VTL7006). Care was taken to insure that no ambient light enters the sensor, as this degrades the image quality. Using the 'DAC scan' feature in Pixelman, images were taken for a range of

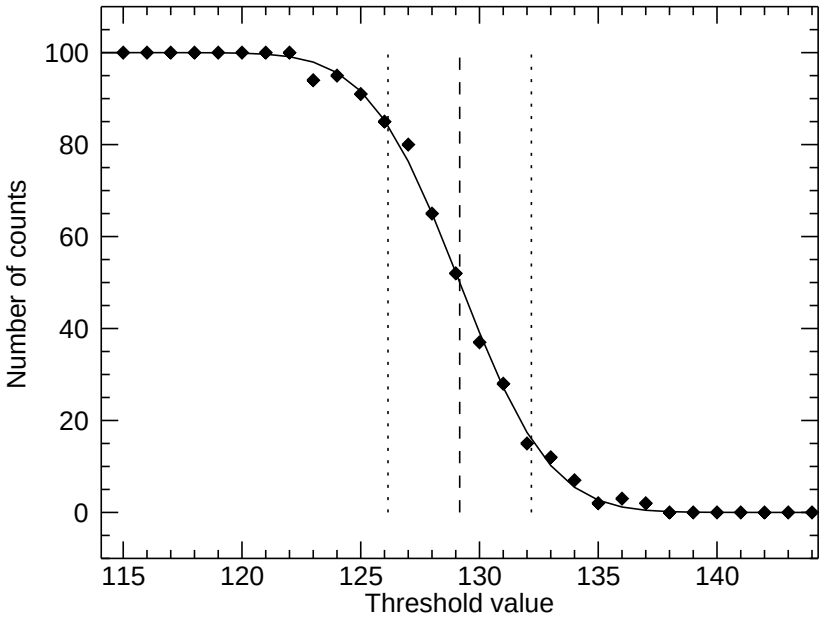

Figure 10: The result of curve-fitting an error function to a pixel cut-off. The mean μ is indicated with the dashed line, the dotted lines are at a distance of one standard deviation σ.

threshold values. This was repeated for ambient temperatures ranging from 20 °C to −60 °C in steps of 10 °C. The chip was configured to inject 100 test pulses charged to 5 ke^- into every pixel. The equalisation procedure was performed for each temperature.

Please note that the ambient temperature does not equal the actual chip temperature. Due to the low thermal conductivity of the test board and the chip producing about 1W while in use, the temperature as reported by the internal sensor is about 55 to 60 °C higher than the outside temperature. For instance, a measurement with the cooling chamber at −20 °C implies an actual chip temperature of about +40 °C.

4.2.1 Faulty pixel count

A simple way to evaluate the image quality is to count the number of pixels registering an incorrect count (often either zero, as in a dead pixel, or a large number, e.g. a noisy pixel). Figure 11 plots this number against the ambient temperature. To the left side, the threshold was fixed to 60. It can be observed that the sensor performs best between −10 and −40 °C. At lower temperatures, the behaviour gets worse again; this is partly because of the threshold DAC value drifting (which will be shown later). To the right side, the same plot was made, but while finding the optimal THL value for each temperature. It can be seen that the effect is not as pronounced in this case.

The most important thing to note about figure 11 is the poor performance of the chip at room temperature. The high number of faulty pixels in this case is most likely due

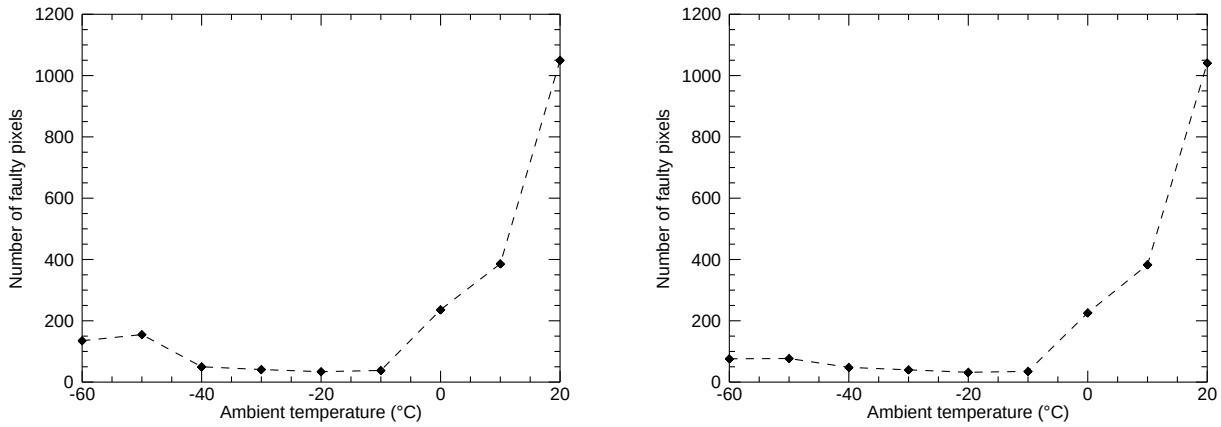

Figure 11: The number of pixels with an incorrect count as a function of outside temperature. Left: threshold fixed to 60. Right: optimal threshold value for each temperature.

to leakage current. Indeed, inspection of figure 12 (right side) shows that the failing pixels are the ones at the border of the sensor, where leakage is naturally higher due to crystal defects. It should also be noted that even at the optimal temperature of -20 °C (figure 12, left side) there are still some faulty pixels.

By inspecting the threshold scans of these pixels, one can distinguish between three categories of failure (figure 13). The first kind always reports a zero count, independent of the THL setting. The second kind is actually working, but is badly misadjusted (located in the tail of the distribution). Its curve will simply be shifted to the right. The third kind shows the right curve shape, but has a lot of noise superimposed.

Table 1: The default DAC settings at -20 °C.

THL	60	Shaper	140	RPZ	255
Preamp	100	Disc	150	GND	117
IKrum	20	FBK	181	Disc_LS	200

4.2.2 DAC drift

Using the 'DAC Dependency Scan' feature in Pixelman, the analog voltage produced by a DAC can be measured as a function of its digital setting. Repeating this for temperatures ranging from -60 °C to $+20$ °C gives additional information about the temperature dependency. After some data processing in IDL, the plots displayed in figure 14 can be drawn. They show how the output voltages depend on the temperature for the DACs at their default settings (see table 1). It appears that most DACs have a major temperature dependency. However, GND seems to be an exception.

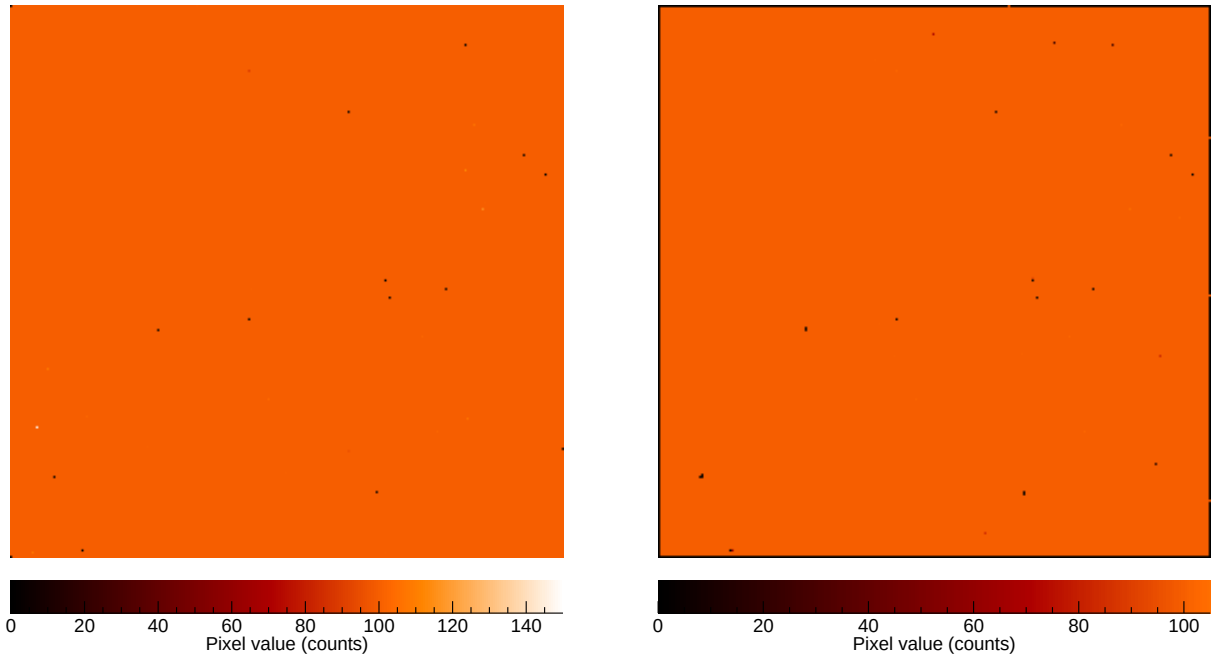

Figure 12: Images as obtained by Pixelman using 100 test pulses. Left side: −20 °C; right side: +20 °C.

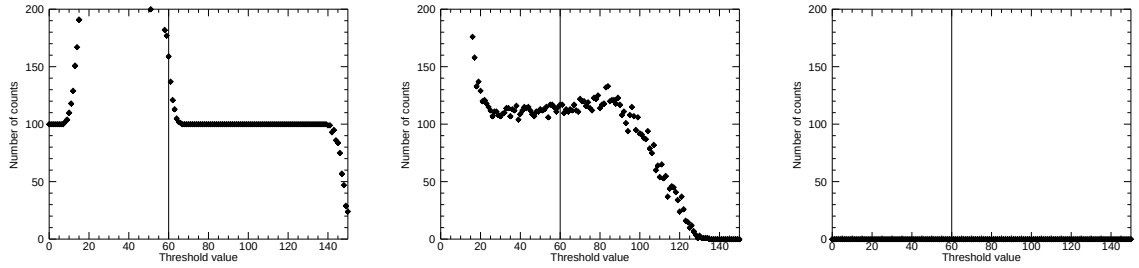

Figure 13: Three kinds of pixel 'failure'. Left: misadjusted, center: noisy, right: dead.

4.2.3 Average pixel properties

Using an IDL script, it is possible to find the noise edge and cut-off point for each pixel. Making a histogram of these values results in roughly Gaussian distributions. This was already shown in figures 6 and 9 for one temperature. However, if one does the same thing at different temperatures, a shift in the average value of both distributions is observed. This is shown in figure 15. Both the noise edge and the cut-off point tend to shift upwards when the temperature decreases. This could be due to the temperature dependent threshold DAC voltage, or changes in transistor gain for the pulse amplification. Perhaps the test pulses themselves change, as their properties are also determined by DAC settings.

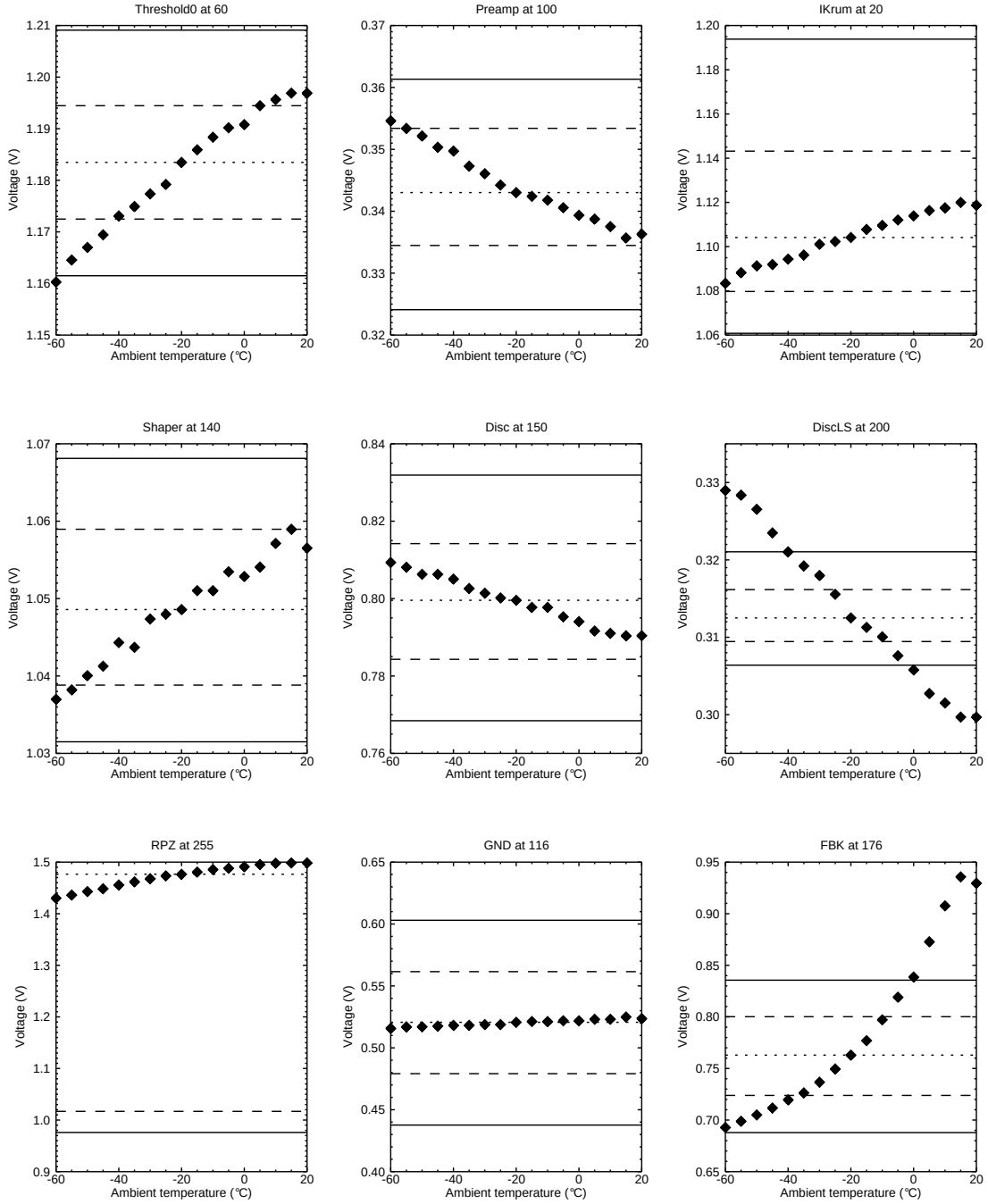

Figure 14: Plots showing the temperature dependency of several DAC voltages. To illustrate the scale of the voltage variation, the DAC voltages at -20 °C are drawn with a horizontal dotted line for the default setting, dashed lines for default $\pm$ 10 steps, and solid lines for default $\pm$ 20 steps.

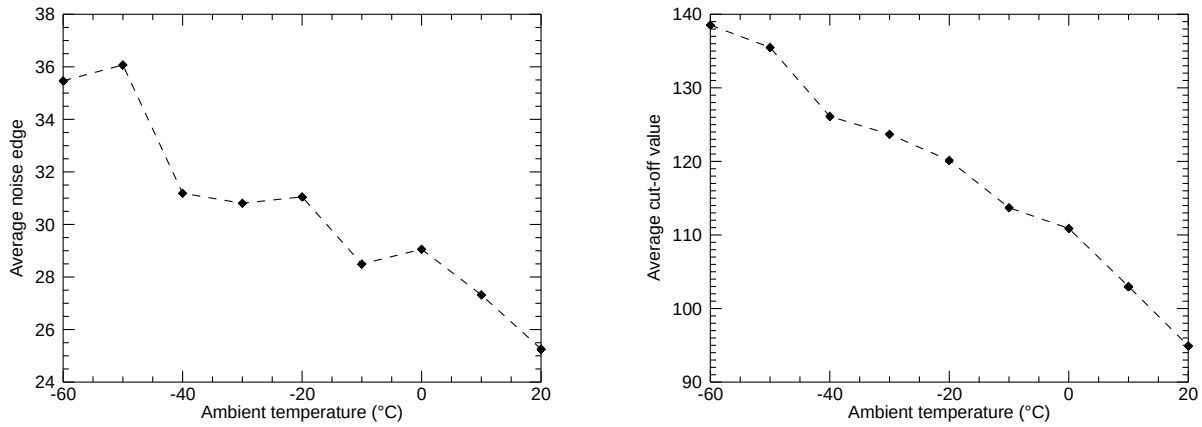

Figure 15: Shift in the average value of the noise edge (left) and cut-off value (right) distributions.

In figure 16, an attempt to calculate a 'signal-to-noise ratio' of the electronics circuitry was made. The y-axis displays the ratio between the range of correct counts (in other words, the difference between the cut-off value and the noise edge) and the cut-off width (caused by the Gaussian noise σ). The interesting thing is that the variation of the threshold DAC with temperature is cancelled out, as both the numerator and the denominator are expressed in the same units (threshold steps). It can be observed that the ratio gets worse at higher temperatures. This implies that the electronic noise in the discrimination circuitry increases with temperature, as can be expected.

4.2.4 Change of equalisation parameters

At the end of each equalisation process, the optimal settings as found by Pixelman can be seen. They include both global DAC settings (ThresholdN and DACPixel, see 2.3.3), and a 256×256 matrix of adjustment bits. Figure 17 shows that the values of ThresholdN and DACPixel are dependent on the temperature during equalisation. This implies that either the DACs themselves change (which is quite likely), or maybe at lower temperatures the uniformity of the pixel array gets worse, forcing the equalisation procedure to apply more adjustment and thus increase the global DAC settings.

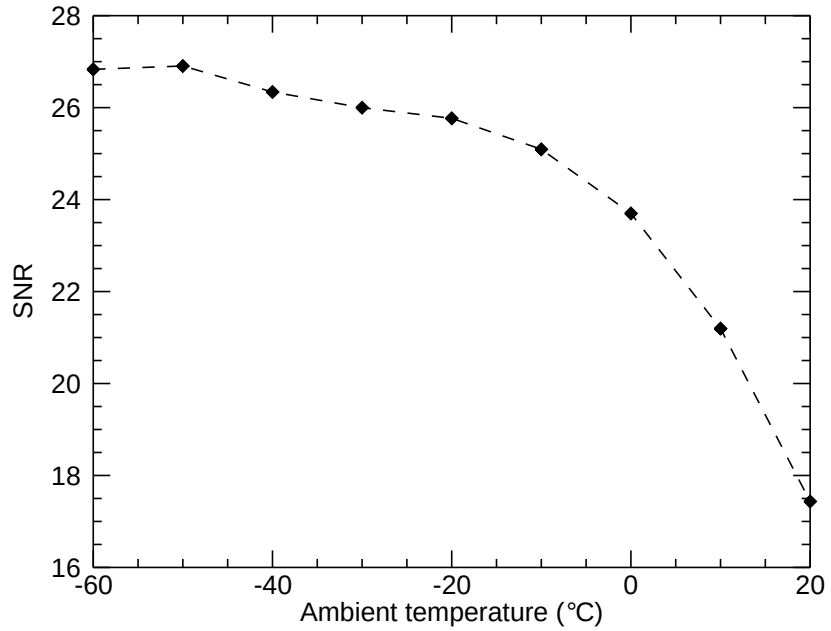

Figure 16: Variation of the average signal-to-noise ratio with temperature.

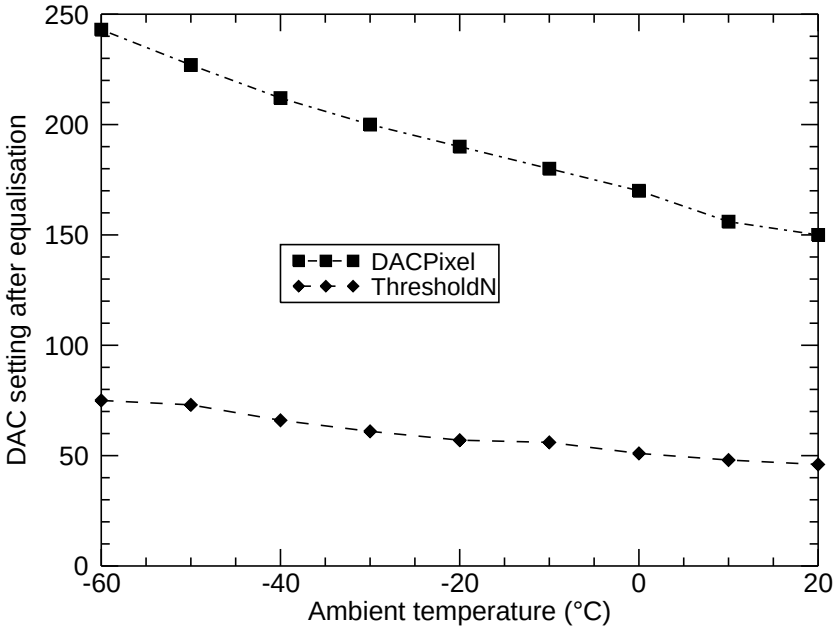

Figure 17: Temperature dependency of the values for ThresholdN and DACPixel as found by Pixelman after the equalisation.

4.3 Dependency on DAC settings

The 'DAC Scan' feature in Pixelman can be applied to any one of the DACs in table 1. By varying one of the DACs and keeping the others at their default setting, the influence on the image can be investigated. To this end, the ambient temperature is held constant at -20 °C and test pulses are injected. The number of faulty pixels, i.e. reporting an incorrect pulse count, is counted for every DAC setting and plotted in figure 18. This shows the usable range for each investigated DAC, and indicates that the default setting is usually the optimal one (lowest faulty pixel count). Figure 19 shows the same kind of measurement but for IKrum set to 14, which seems to improve the image by a small amount. It is difficult to find the absolute optimum setting due to the large parameter space (each pixel is dependent on all of the DAC settings working together).

4.4 Example X-ray images

The function of the sensor is of course to make X-ray images, and this report would not be complete without some demonstration pictures. To this end, a laboratory X-ray tube with a Molybdenum target and an acceleration voltage of 50 kV was used as a light source. This results in X-rays with characteristic peaks at 17.9 keV (K-α) and 19.5 keV (K-β) in addition to broad-spectrum Bremsstrahlung. The image quality can be improved by doing some extra steps:

- Take a flat-field image. This is done by exposing the sensor to the X-ray tube without anything in front of it. To reduce noise, one should take many exposures and average them. This works because photon arrival conforms to Poisson statistics (shot noise), and the relative standard deviation is inversely proportional to the square root of the mean. So e.g. by averaging 16 exposures, the shot noise is reduced by a factor of 4.

- Get some small, interesting objects and place them right in front of the sensor. Acquire and average several images again, for the same reason as above.

- Divide the image by the flat-field. This compensates for uneven X-ray exposure or variations in pixel sensitivity.

- Display the image with a proper colour scale, providing the most contrast. Sometimes a logarithmic scale may be useful.

The final images are shown in figure 20.

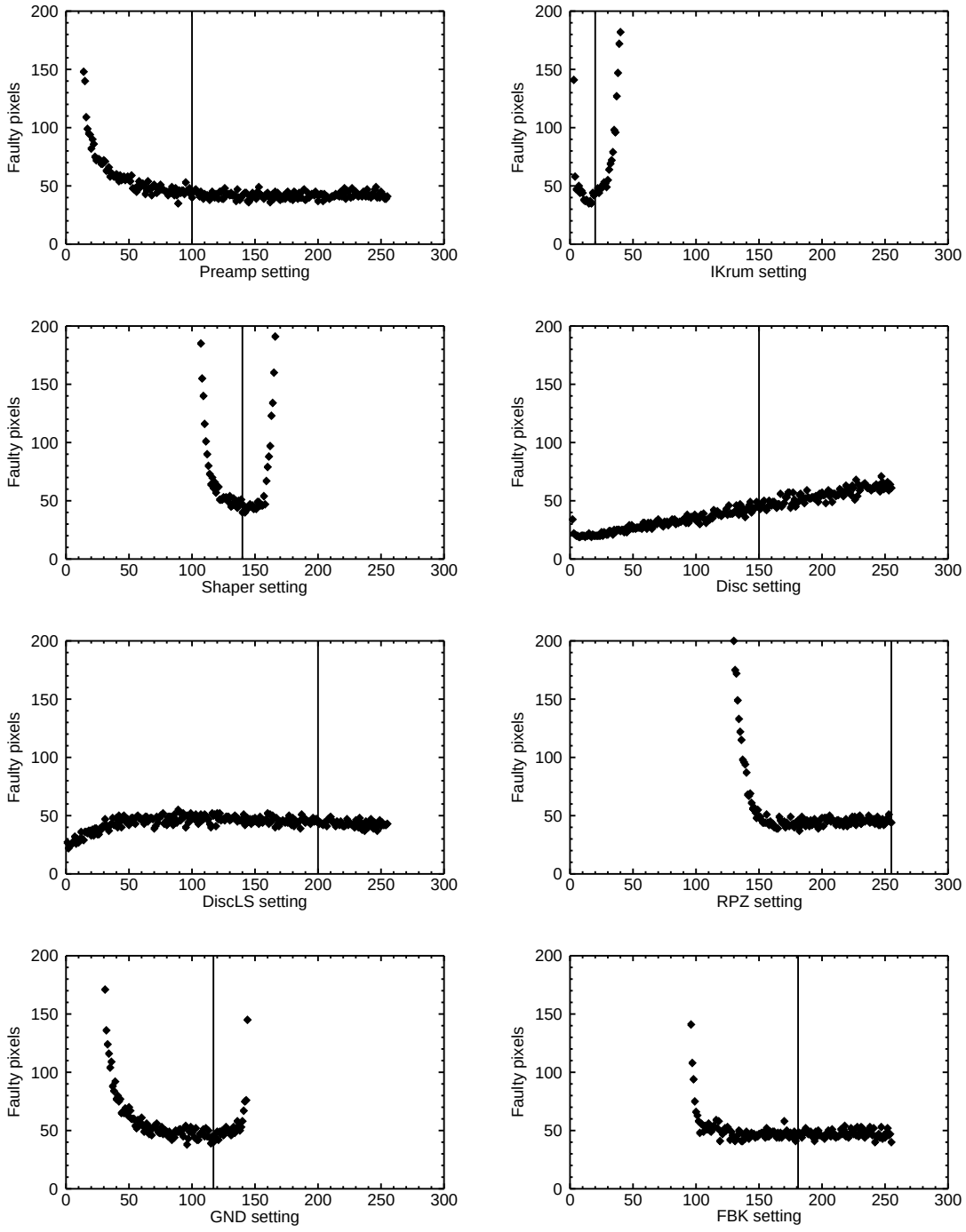

Figure 18: The number of pixels with an incorrect count as a function of various DAC settings. For every plot, one particular DAC was varied while the others were at a constant value (indicated by vertical lines). Test conditions: default DAC settings (table 1), 100 test pulses.

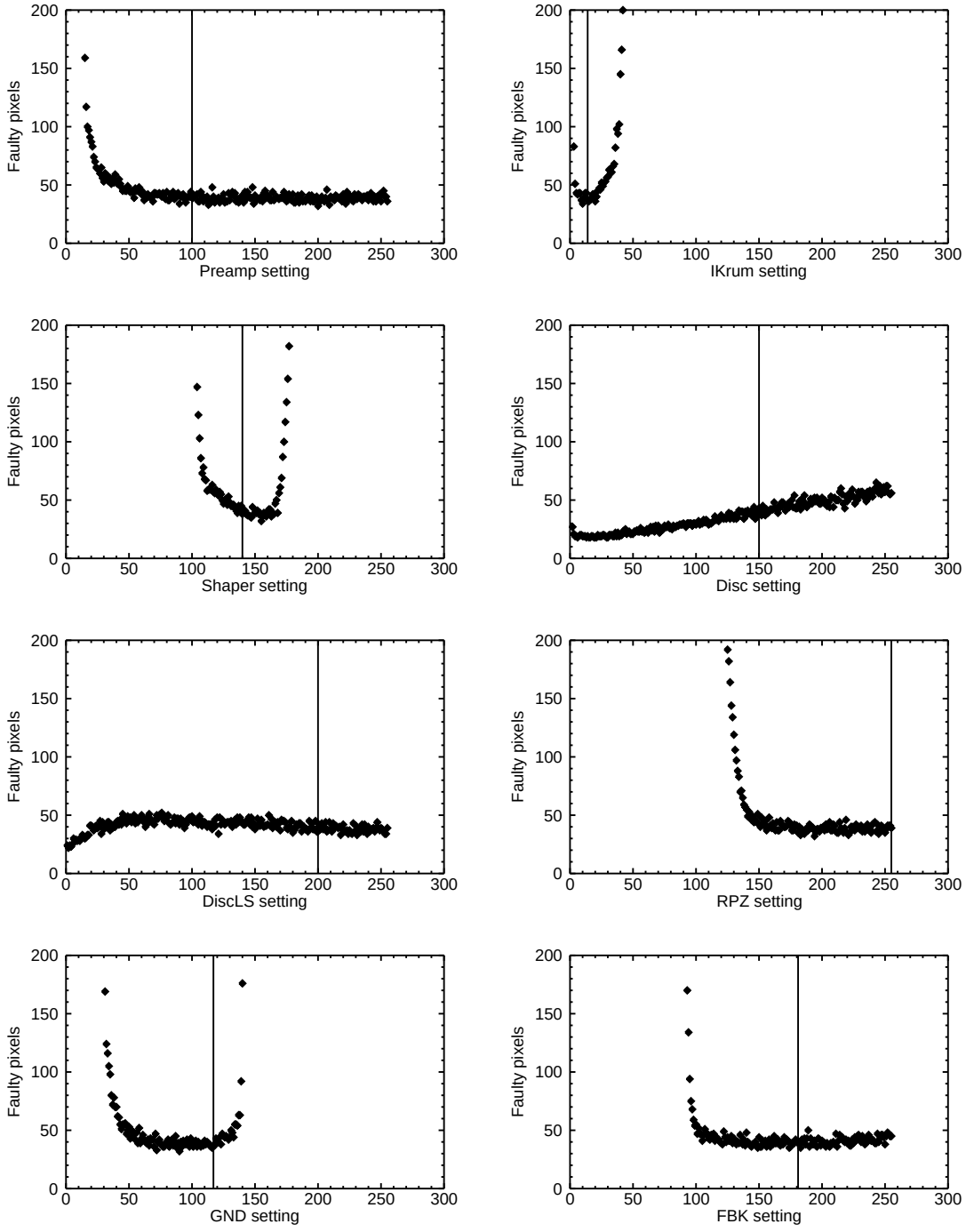

Figure 19: Identical to figure 18, but with IKrum set to 14 instead of 20 (default).

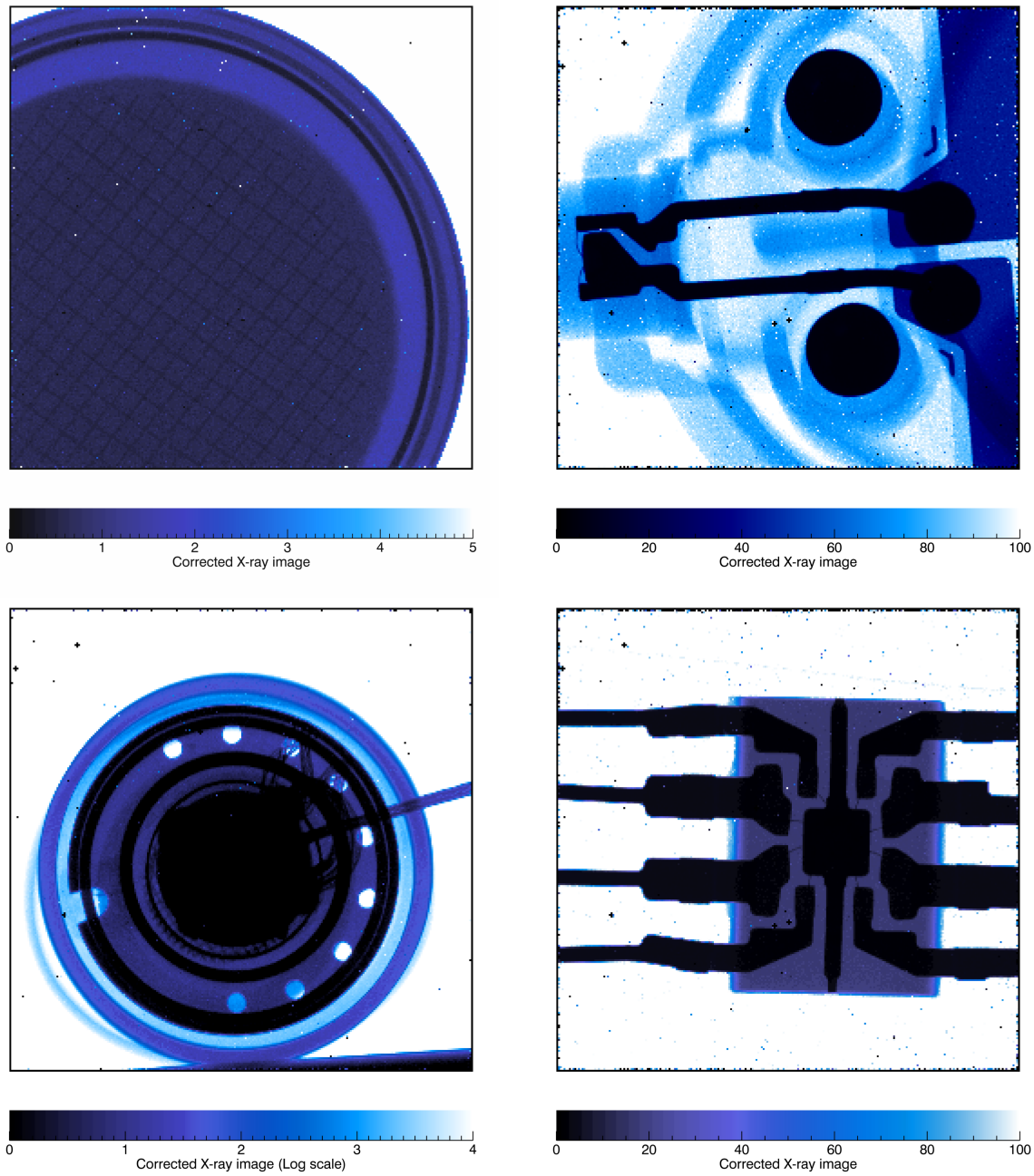

Figure 20: Demonstration X-ray images of various objects. Top left: CR2032 coin cell battery; top right: LED bicycle light; bottom left: Sennheiser CX300 in-ear headphone (top view); bottom right: Op-amp in 8-pin DIP package.

5 Conclusions

- It has been found that the behaviour of the Medipix3 has a major temperature dependency. There are several reasons for this. A high temperature causes excess leakage current which results in more dead pixels, mostly around the border. The on-chip DAC voltages are also influenced by temperature, and in turn they influence the image acquisition. Some DACs are automatically tuned by Pixelman during the equalisation. After major temperature changes, this process needs to be repeated.

- The ideal ambient temperature has been found to be -20 °C; this translates to a chip temperature of about $+40$ °C.

- The DAC settings that are either set by default or found automatically in Pixelman work well. Manually fine-tuning them could marginally improve the image, but certain pixels will stay dead anyhow.

References

[1] H. Graafsma, "X-ray detectors. How do they work? How are they characterized?", Photon Science Detector Group, DESY, Hamburg, 2011.

[2] R. Ballabriga, M. Campbell, E. Heijne, X. Llopart, and L. Tlustos, "The medipix3 prototype, a pixel readout chip working in single photon counting mode with improved spectrometric performance," *Nuclear Science Symposium Conference Record, 2006. IEEE*, vol. 6, pp. 3557–3561, 2006.

[3] R. Ballabriga, W. Wong and X. Llopart, "Medipix3 Manual," CERN, v1.9, 11/9/2009.

[4] R. Ballabriga, M. Campbell, X. Vilasís-Cardona, "The Design and Implementation in 0.13μm CMOS of an Algorithm Permitting Spectroscopic Imaging with High Spatial Resolution for Hybrid Pixel Detectors. oai:cds.cern.ch:1259673," Ph.D. Thesis, Universitat Ramon Llull, Barcelona, 2009.

[5] R. Ballabriga, D. Turecek, "Instructions for equalizing the Medipix3 chip with Pixelman"

[6] Tomáš Holý, Daniel Tureček, Zdeněk Vykydal, "Pixelman", `http://aladdin.utef.cvut.cz/ofat/others/Pixelman/Pixelman_download.html`, Institute of Experimental and Applied Physics, Prague.