Non-Uniformity Correction of Infrared Images by Midway Equalization

Yohann Tendero\(^1\), Stéphane Landeau\(^2\), Jérôme Gilles\(^3\)

\(^1\)CMLA, ENS Cachan, France (tendero@cmla.ens-cachan.fr)
\(^2\)DGA/CEP, France (stephane.landeau@dga.defense.gouv.fr)
\(^3\)UCLA Math Dept., United States (jegilles@math.ucla.edu)

Communicated by Jean-Michel Morel Demo edited by Yohann Tendero

Abstract

The non-uniformity is a time-dependent noise caused by the lack of sensor equalization. We present here the detailed algorithm and online demo of the non-uniformity correction method by midway infrared equalization. This method was designed to suit infrared images. Nevertheless, it can be applied to images produced for example by scanners, or by push-broom satellites. This single image method works on static images, is fully automatic, has no user parameter, and requires no registration. It needs no camera motion compensation, no closed aperture sensor equalization and is able to correct for a fully non-linear non-uniformity.

Source Code

The source code, version 2.0, is available from the article web page\(^1\). The documentation is included in the archive. Basic compilation and usage instructions are included in the README.txt file. The source code for the contrast adjustment preprocessing is available from the same location with its own documentation (see [10]).

The demo permits to try the proposed method on several well chosen test images, and on any uploaded image. To improve visibility (without changing the algorithm) the input image can be preprocessed using the “Simplest Color Balance” algorithm [10]. The s1 parameter is the percentage of pixels saturated to black and s2 the percentage saturated to white. If both s1 = s2 = 0 the image is simply stretched to [0, 255] by an affine contrast change (causing no loss of information) before applying the denoising algorithm. The output image is re-stretched to [0, 255] by an affine contrast change. The outputs are 1) the processed image and 2) the optimal scale parameter s found by the algorithm.

Keywords: non uniformity correction, infrared, fixed pattern noise, focal plane array, NUC, denoising

\(^1\)https://doi.org/10.5201/ipol.2012.glmt-mire
1 Introduction

An imaging sensor is a device that collects photons and converts them into charges. The majority of imaging sensors are charge-coupled devices (CCD). The standard readout technique of CCDs works for each line (or row) independently and consists of transporting charges from the pixels to a counter. Each pixel has its own function transfer response. Furthermore, for each line the counter transfer function is different. In most cases this non-uniformity (NU) of sensor response leads to a structured noise resulting in a row or line pattern in the images. This “noise” is sometimes called fixed pattern noise, or non-uniformity (NU) noise. It is a serious practical limitation to both civilian and military applications - as it severely degrades image quality. For uncooled infrared cameras the difficulty is increased because the detector response evolves quickly with time. This means that the correction cannot be done once and for all by the manufacturer. A correction is so much needed, that in many uncooled infrared cameras a flap closes [6, 7] every 30 seconds to perform a partial calibration. This interrupts the image flow, which is calamitous. Therefore a good NU algorithmic correction is a key factor in ensuring the best image quality and the robustness of the downstream applications [3, 5, 11, 14, 12].

![Observed image (left) taken with a Thales Minie-D camera, corrupted by a structured noise called non-uniformity.](image1)

The non-uniformity can be considered as a 1D flicker of the columns inside each frame. The adequate movie deflickering algorithms can be adapted, to equalize the columns (resp. the lines). The proposed method therefore applies a movie deflickering algorithm to the series formed by the columns (resp. lines) of an (infrared) image.

We present here the detailed algorithm and on line demo of the non-uniformity correction method introduced in [13]. This method was designed to suit infrared images. Nevertheless, it can be applied to images produced by scanners, push-broom satellites [8, 2] (where this problem is called relative radiometric correction or normalization), non equalized CCDs, etc.

The obtained single image method works on each single image. It is fully automatic and requires no user parameter. Being a single image process, it requires no registration, no camera motion compensation, and no closed aperture sensor equalization.

1.1 The Perturbation Model

The perturbation model is

\[ o(i, j, t) = \phi(i, j, t) (u(i, j, t) + \eta(i, j, t)), \forall (i, j, t) \in \{1, \cdots, N\} \times \{1, \cdots, M\} \times \mathbb{R}_+ \]
where

- \((i, j, t) \in \{1, \cdots, N\} \times \{1, \cdots, M\} \times \mathbb{R}_+\) is the pixel at position \((i, j)\) and time \(t \geq 0\),
- \(u(i, j, t)\) is the ideal landscape value,
- \(\eta(i, j, t)\) is a random photon noise,
- \(\phi(i,j,t) : \{0, \cdots, 255\} \rarrow \{0, \cdots, 255\}\) is the contrast change (transfer function) of the pixel sensor at position \((i, j)\) and time \(t \geq 0\)
- \(o(i, j, t)\) is the observed value at position \((i, j)\) and time \(t \geq 0\).

The goal of a non-uniformity correction algorithm is to compensate for the local contrast changes induced by the \(\phi(i,j,t)\).

### 1.2 Algorithm

The proposed non-uniformity correction is based on the fact that single columns (or lines, depending of the readout system) of an image carry enough information by themselves for an equalization. The images being continuous, the difference between two adjacent columns is statistically small, implying that two neighboring histograms are nearly equal. So the proposition is to transport the histogram of each column (or line) to the midway (see [1] for further information on the midway algorithm) of the histograms of the neighboring columns (resp. lines). In presence of strong non-uniformity (NU) it will be useful to perform this sliding midway method over a little more than two columns, because the NU is not independent in general. In the sequel, the non-uniformity correction is assumed to be required for the image columns.

### 1.3 “Midway Infrared Equalization (MIRE)”

Our method is described as algorithm 1. Since we work on images separately the method is not affected by motions or scene changes. This completely avoids “ghost artifacts” [11, 4] and any problem caused by the calibration parameters drifting over time. The above steps 1 – 3 of this algorithm will be summarized in a single formula in the final algorithm below, without recurring to inverse histograms. An algorithm selecting \(s\) is given in the next section.

#### 1.3.1 Automatically Fitting the Perfect Parameter

The non-uniformity leads to an increased total-variation norm. Hence, following the idea of [9], the smoothest image is also the one with little or no non-uniformity at all. So the simplest way to find the good \((s^*)\) parameter automatically is

\[
s^* = \arg\min_s \|I_s\|_{TV-line}
\]

where \(I_s\) is the image processed by MIRE with the parameter \(s\). The discrete total variation is defined by \(\|I\|_{TV-line} = \sum_{i,j} |(\nabla I)_{i,j}|\) with \((\nabla I)_{i,j} = (I_{i,j+1} - I_{i,j})\).

The optimization can be done by scanning a broad range of values for \(s\):

- Choose a \(s\_step\) and a \(s\_max\) (\(s\_step = 0.5\) and \(s\_max = 8\) by default for the demo).
- Start with \(s = 0\)
- Repeat until \(s > s\_max\):
**Algorithm 1**: The “Midway Infrared Equalization (MIRE)” algorithm.

<table>
<thead>
<tr>
<th>input</th>
<th>a discrete 8−bits image $o(i,j) \in {0, \cdots, 255}$ for $(i,j) \in {1, \cdots, N} \times {1, \cdots, M}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>output</td>
<td>a discrete image $d(i,j)$</td>
</tr>
</tbody>
</table>

for each column, $j \in \{1, \cdots, M\}$ do

1. compute the cumulative histogram $H_j$ of each column $c_j$

   $H_j : \{0, \cdots, 255\} \rightarrow [0, 1]$

   $l \mapsto H_j(l) = \frac{1}{N} \sum_{k=0}^{l} \sum_{i=1}^{N} 1_{\{o(i,j)=k\}}$

2. for each column $c_j$ do

   3. compute a local midway histogram

      \[ \tilde{H}_j^{-1} := \sum_{k \in (-n, \cdots, n)} g_k H^{-1}_{k+j} \]

      using Gaussian weights $g(k) = g_s(k) = \frac{1}{s\sqrt{2\pi}} e^{-k^2/2s^2}$ with standard deviation $s$, the window size is $2n$ where $n = \text{floor}(4s)$, and

      \[ H_j^{-1} : [0, 1] \rightarrow \{0, \cdots, 255\} \]

      \[ z \mapsto H_j^{-1}(z) = \min\{l \in \{0, \cdots, 255\} \mid H_j(l) \geq z\} \]

4. end

5. specify the histogram of the column $c_j$ onto this midway histogram $\tilde{H}_j$

   \[ d(i,j) = \tilde{H}_j^{-1}(H_j(o(i,j))) \forall i \in \{1, \cdots, N\} \]

6. end
1. Process the image.
2. Increase $s$ of $s_{\text{step}}$.

**Examples : Automatically Fitting the $s$ Parameter.** Figure 2 illustrates the parameter influence on the processed image and the parameter-guess step. The raw input image and the MIRE-processed images are shown for increasing values of the scale $s$.

![Images](image.png)

Figure 2: First the raw image, then the images processed with increasing values of $s$ (from top to bottom and left to right : $s = 3, 5, 15, 17, 22, 25$). The parameter $s$ is measured in pixel units.

### 1.4 Implementation

The midway algorithm has a clever and simple implementation (see remark 1, p.7 of [1]) that avoids inverting explicitly histograms (although the above formulation as an average of inverse histograms is more transparent). The whole midway procedure boils down to sorting the values on each column, and averaging the sorted values as stated below in the formula defining $d(i, j)$. This implementation is detailed as algorithm 2.

### 1.5 Examples

The examples in figures 3-8 illustrate several aspects of the MIRE algorithm. We display the original raw image and the MIRE processed image. While the quality is not perfect good improvement can be seen. Notice that once the $NU$ has been removed, infrared images could be further processed with classic image restoration algorithms. Remark that on non corrupted images the algorithm is nearly the identity (see the last example in figure 4).

### Acknowledgement

Work partially supported by the Direction Générale de l’Armement, the Office of Naval Research under grant N00014-97-1-0839 and by the European Research Council, advanced grant Twelve labours.

### Credits

All the images used in this article are the work of the authors, and available under the CC-BY license, except for the standard boat test image.
Figure 3: Examples of the MIRE algorithm results.
Figure 4: Examples of the MIRE algorithm results. The last “boat” experiment shows that the algorithm does not alter the quality if an uncorrupted image is used as input.
Figure 5: Examples of the MIRE algorithm results.
Figure 6: Examples of the MIRE algorithm results.
real raw infrared images

processed images

Figure 7: Examples of the MIRE algorithm results.
Algorithm 2: The MIRE algorithm implementation.

input: corrupted image \( o \) (supposed corrupted with \( NU \) among columns in the following)
output: the \( o \) image, denoised

1. apply classic mirror-symmetry of size \( \text{floor}(4s_{\text{max}}) \) in the \( j \) direction to get rid of border effects (\texttt{borders.cpp} routine);
2. set \( s = 0 \), \( s_{\text{step}} = 0.5 \), \( s_{\text{max}} = 8 \) (not a critical parameter);
3. while \( s < s_{\text{max}} \) (\texttt{MIRE.cpp}) do
   4. process \( o \) with \( s \) (see above) to get the corrected image \( d \)
      
      \[
      d(i, j) := \sum_{k=-n}^{n} g_s(k) o(\sigma_{j+k}^{-1}(i), j + k)
      \]
      where \( g_s \) is the Gaussian with standard deviation \( s \) and \( \sigma_{j+k}(i) \) the permutation associated to the ascending reordering of the column \( j + k \) \((o(\sigma_{j+k}(i), j + k) \leq o(\sigma_{j+k}(i+1), j + k) \forall j)\);
   5. compute \( TV - \text{line} \) and store it;
   6. increment \( s \) of \( s_{\text{step}} \);
4. end
7. keep the value for \( s \) minimizing the \( TV - \text{line} \) criterion;
8. apply an affine contrast stretch so that the \textit{min} of the image becomes 0 and the \textit{max} 255;

Figure 8: Examples of the MIRE algorithm results.
References


