IPOL IMAGE PROCESSING ON LINE

Published in Image Processing On Line on 2017–04–24.
Submitted on 2016–07–15, accepted on 2017–03–22.
ISSN 2105–1232 © 2017 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
https://doi.org/10.5201/ipol.2017.181

Midway Video Equalization

Javier Sánchez

CTIM, University of Las Palmas de Gran Canaria, Spain
(jsanchez@ulpgc.es)

Abstract

This article presents an implementation of the 'Midway Equalization' method for videos. This technique allows us to modify the image histograms so that they present similar luminances. We propose two algorithms: the first one based on histogram inversion and the second one on the sorting of images intensities. The former computes the histograms and then finds the contrast change functions by convolving the inverse histograms with a Gaussian function. The latter starts by sorting the pixels of each image by intensity; the temporal signals, composed of all gray levels of the same rank, are then convolved with a Gaussian function. In this sorting method, the resulting histograms are more similar and homogeneous. Nevertheless, the histogram strategy is faster and provides good results in general. The algorithms include a 'dithering' option for reducing quantization artifacts. The whole implementation depends on a single parameter, the standard deviation, that is used for Gaussian convolutions. The experiments show several examples, including the quantization artifacts that appear in some situations and the benefits of dithering. We observe that these artifacts are usually more important in the histogram method.

Source Code

The reviewed source code and documentation for this algorithm are available from the web page of this article[1]. Compilation and usage instruction are included in the `README.txt` file of the archive.

Keywords: histogram matching; midway equalization; dithering method

1 Introduction

Midway equalization [3] is a method for assigning the same intermediate histogram to a pair of images. It permits to remove some types of brightness changes in image pairs. It is especially useful for equalizing two images of a scene taken from the same camera with different exposure times. This method is implemented for two images in [8].

Midway equalization belongs to the class of global histogram equalization (GHE) methods [7], which applies a unique contrast change function to the whole image. These methods are interesting

[1] https://doi.org/10.5201/ipol.2017.181

because of their simplicity, but they usually present several shortcomings, such as the difficulty to achieve a well-balanced enhancement over different image parts, the appearance of unpleasant visual artifacts or the change in the average image luminance [18].

Adaptive histogram equalization (AHE) techniques [15, 1, 17], on the other hand, compute several change functions for different parts of the image. They usually enhance the contrast in each pixel depending on the information of its local neighborhood, with the aim of isolating the effects on dark and bright regions. Local image contrast methods are usually based on human perception, like, e.g., the multiscale retinex method [9, 13].

In this work, we extend the implementation proposed in [8] to a sequence of images, following the method proposed in [4]. The implementation is similar to [8], with the histogram of the frames modified according to the information of several images in a temporal window. We describe two implementations: the sorting method [16], based on the sorting of the images intensities; and the histogram method, based on the matching of normalized cumulative histograms.

The sorting method orders the pixels of the images by intensity; then, it temporally convolves all intensities of a given rank with a Gaussian function; and finally, it rearranges the pixels in their original locations. This method allows for a finer control on the equalization, which results in more similar and homogeneous histograms. It is the base for exact histogram specification [2]. In the histogram method, the contrast change functions are computed using inverse cumulative histograms. This method is faster but definitely more sensitive to quantization artifacts.

In order to work with color images, we convert them to grayscale levels. The color channels are modified according to the rate of change between the gray values before and after the equalization in each pixel independently. It is also possible to treat the channels separately, but this may introduce important color changes in the sequence.

This type of equalization works correctly when the relative intensity of the objects is preserved in the scene. However, when the appearance of the scene changes, or when the illumination changes do not preserve the order of the intensities, severe artifacts may appear after the equalization. Section 4 shows successful examples, but also illustrates potential artifacts.

2 Midway Equalization for Video

We define a set of color images as $\mathbf{I}_i$, $i := 1, \ldots, N_f$, with N_f the number of frames in the input video. At the beginning of the process, the methods convert the color images to gray levels as $I_g := 0.2989 \times \mathbf{I}_R + 0.5871 \times \mathbf{I}_G + 0.1140 \times \mathbf{I}_B$. They also add white noise if the dithering option is chosen. In this case, since the intensities of the images are expected to be in the range $[0, 255]$, we add a small amount of noise, with a standard deviation of $\sigma_d := 2$.

2.1 Method Based on Histograms

Given a set of gray level images, $\{I_i\}$, defined on the same discrete domain, $\Omega = \{1, \ldots, n\} \times \{1, \ldots, m\}$, their cumulative normalized histograms are defined by

$$H_i(\lambda) := \frac{1}{|\Omega|} \left| \{(i,j) \in \Omega / I_i(i,j) \leq \lambda\} \right|, \tag{1}$$

where $|\Omega|$ denotes the cardinal of the set Ω. In the midway video equalization [4], the contrast change functions f_i are calculated through a temporal convolution with a Gaussian function as

$$f_i(\mu) := \sum_{j=i-k}^{i+k} G_\sigma(j-i) H_j^{-1}(H_i(\mu)), \tag{2}$$

with $G_\sigma(s) := We^{\frac{s^2}{2\sigma^2}}$ and $W := 1/\sum_{i=-k}^{i=k} e^{\frac{i^2}{2\sigma^2}}$ a normalizing coefficient. The image intensities are then modified as

$$I_i(x) := f_i(I_i(x)). \tag{3}$$

The method first computes the cumulative normalized histograms H_i as in Equation (1). Then, it estimates the contrast change functions f_i based on temporal Gaussian convolutions (Equation (2)). Finally, it transforms the images according to Equation (3). Algorithm 1 includes these steps with the initial conversion to gray levels and the dithering option.

Algorithm 1: Main Algorithm

input : {**I**}, σ, *dithering*
output: {**I**}
Convert images {**I**} to grayscale $\{I_g\}$
if *dithering* **then**
  Add white noise to $\{I_g\}$ with $\sigma_d := 2$
for $i \leftarrow 1$ **to** N_f **do**
  Normalize Histogram($I_g(i)$, $H(i)$) *// Algorithm 2*
Compute Contrast Changes Functions($\{H\}$, σ, $\{f\}$) *// Algorithm 3*
Apply Contrast Change({**I**}, $\{I_g\}$, $\{f\}$) *// Algorithm 4*

Algorithm 2 computes the cumulative normalized histogram for an image. This function initializes the histograms bins to zero and increases the count of the histogram positions for each corresponding intensity value. Constant `BIN_RATE` is used for converting intensity values to histogram positions. At the end, the histogram is normalized and accumulated. N_p is the number of pixels per image, $N_p := n \times m$.

These algorithms use the following constants:

- `PIXEL_RANGE`: Defines the range of intensity values or color depth. For 8 bits, it is equal to 256. For color images of 24 bits, it should also be defined as 256 (channels are treated separately). It is usually a power of 2 (2^8, 2^{10}, 2^{12} or, e.g., 2^{16}).
- `BIN_SIZE`: Defines the histogram resolution. If it is equal to `PIXEL_RANGE`, then each bin corresponds to each intensity level. Nevertheless, it is typical to use higher resolutions, such as 1024 or 2048, in order to have a finer control over the contrast change.
- `BIN_RATE`: Is the rate `BIN_SIZE`/`PIXEL_RANGE` and is used to convert image intensities to histogram positions.

The constant `PIXEL_RANGE` depends on the format of the images and can be read from the files. `BIN_SIZE` determines the number of bins in the histogram. Choosing a constant `BIN_SIZE` larger than `PIXEL_RANGE` does not have any effect on gray level images. However, it matters for color images since we interpolate the color channels.

Algorithm 3 calculates the contrast change functions. For each image, it selects the corresponding gray levels in the cumulative normalized histograms of the neighbor frames and applies a Gaussian smoothing. Variable *radius* defines the number of frames used in the Gaussian convolutions. It is chosen as $radius := 2\,\sigma$, which provides a good precision in general. In order to accelerate the

Algorithm 2: Normalize Histogram

```
input  : I, H
output: H
// initialize histogram
for i ← 1 to BIN_SIZE do
    H(i) ← 0
// compute histogram
for i ← 1 to N_p do
    p ← round(BIN_RATE × I(i))
    if p < 1 then
        p ← 1
    else if p > BIN_SIZE then
        p ← BIN_SIZE
    H(p) ← H(p) + 1
// normalize histogram
for i ← 1 to BIN_SIZE do
    H(i) ← H(i)/N_p
// accumulate histogram
for i ← 2 to BIN_SIZE do
    H(i) ← H(i) + H(i − 1)
```

method, we use an array, p_2, to find the correspondences for each gray level in the neighbor frames. We use two separate loops for finding the intensity values in the previous and following frames. The Gaussian weights are computed at the same time.

Finally, Algorithm 4 applies the contrast change to the images. The color channels are transformed according to the ratio of the original and equalized gray levels in each position. Function $gray2rgb$ scales the color pixels, given in $\mathbf{I}(i,j)$, by the factor $post/I_g(i,j)$. This function controls that the intensity values remain inside the limits of the allowable intensity range, through a scaling of the color channels. This is the strategy followed by other IPOL articles, such as [10, 11]. Another alternative is to project these values in the RGB cube in order to preserve the Hue value of the original color, like in [14]. Constant `BIN_RATE`, and its inverse, are used for converting image intensities to histogram positions and vice-versa.

The overall computational complexity of this method is $O(N_f N_p)$. The first and third steps go through the images once. The second step depends on the size of the histogram bins (BIN_SIZE), which is typically much smaller than the image dimensions.

2.2 Method Based on the Sorting of the Pixels

The first step of the sorting method is to sort the pixels of each image by intensity. This creates a new volume where intensities with the same rank, in different frames, are temporally aligned in the same position. It is important that this sorting does not correlate with the image reading direction, because it may produce strips of constant intensities after equalization, like in Figure 1. To avoid this, we need to randomize the gray levels before sorting them.

In the second step, this volume is smoothed in each rank through a one dimensional temporal Gaussian convolution. This yields a new set of images with histograms having more similar distributions. In the final step, the pixels are rearranged in their original positions in order to obtain the

Algorithm 3: Compute Contrast Changes Functions

input : $\{H\}$, σ
output: $\{f\}$
$radius \leftarrow round(2\,\sigma)$
for $i \leftarrow 1$ **to** N_f **do**
  $min \leftarrow i - radius$
  $max \leftarrow i + radius$
  if $min < 1$ **then**
    $min \leftarrow 1$
  if $max > N_f$ **then**
    $max \leftarrow N_f$
  for $j \leftarrow 1$ **to** $2\,radius + 1$ **do**
    $p_2(j) \leftarrow 0$
  for $p_1 \leftarrow 1$ **to** *BIN_SIZE* **do**
    // variables for the Gaussian convolution
    $sum \leftarrow p_1$
    $weight \leftarrow 1$
    // compute contrast change function with respect to previous frames
    for $j \leftarrow min$ **to** $i - 1$ **do**
      $k \leftarrow j - min + 1$
      // find the corresponding level in histogram $H(j)$
      while $p_2(k) \leq$ *BIN_SIZE* ***and*** $H(j, p_2(k)) < H(i, p_1)$ **do**
        $p_2(k) \leftarrow p_2(k) + 1$
      if $p_2(k) >$ *BIN_SIZE* **then**
        $p_2(k) \leftarrow$ BIN_SIZE
      // update the variables for the Gaussian convolution
      $weight \leftarrow weight + e^{\frac{(i-j)^2}{2\,\sigma^2}}$
      $sum \leftarrow sum + e^{\frac{(i-j)^2}{2\,\sigma^2}} \times p_2(k)$
    // compute contrast change function with respect to following frames
    for $j \leftarrow i + 1$ **to** max **do**
      $k \leftarrow j - min + 1$
      // find the corresponding level in histogram $H(j)$
      while $p_2(k) <$ *BIN_SIZE and* $H(j, p_2(k)) < H(i, p_1)$ **do**
        $p_2(k) \leftarrow p_2(k) + 1$
      if $p_2(k) >$ *BIN_SIZE* **then**
        $p_2(k) \leftarrow$ BIN_SIZE
      // update the variables for the Gaussian convolution
      $weight \leftarrow weight + e^{\frac{(i-k)^2}{2\,\sigma^2}}$
      $sum \leftarrow sum + e^{\frac{(i-k)^2}{2\,\sigma^2}} \times p_2(k)$
    $f(i, p_1) \leftarrow sum/weight$

Algorithm 4: Apply Contrast Change

input : $\{\mathbf{I}\}$, $\{I_g\}$, $\{f\}$
output: $\{\mathbf{I}\}$
for $i \leftarrow 1$ **to** N_f **do**
  for $j \leftarrow 1$ **to** N_p **do**
    $prev \leftarrow I_g(i,j)$
    // select relative position in the contrast change function
    $p \leftarrow round(\texttt{BIN\_RATE} \times prev)$
    if $p < 1$ **then**
      $p \leftarrow 1$
    else if $p >$ *BIN_SIZE* **then**
      $p \leftarrow \texttt{BIN\_SIZE}$
    // get corresponding position in the contrast change function
    $q \leftarrow i \times \texttt{BIN\_SIZE} + p$
    // obtain the corresponding contrast change in grayscale
    $post \leftarrow 1/\texttt{BIN\_RATE} \times f(i,q)$
    // apply the contrast change to each color channel
    $\mathbf{I}(i,j) \leftarrow gray2rgb(post/prev, \mathbf{I}(i,j))$

Figure 1: Artifacts appearing with the sorting method: On the left, one frame of the *Pieds Nickelés* sequence (see experiment in Figure 4); in the middle, the result of the sorting method without randomizing the gray levels (observe the strips of constant intensities); on the right, the result using the randomization.

final equalized images. This method is detailed in Algorithm 5.

Algorithm 5: Sorting Method

input : $\{\mathbf{I}\}$, σ, *dithering*
output: $\{\mathbf{I}\}$
Convert images $\{\mathbf{I}\}$ to grayscale $\{I_g\}$
if *dithering* **then**
  Add white noise to $\{I_g\}$ with $\sigma_d := 2$
Sort Images($\{I_g\}$, $\{index\}$)
Gaussian Convolution($\{I_g\}$, $\{I_s\}$, σ)
Apply Contrast Change($\{\mathbf{I}\}$, $\{I_g\}$, $\{I_s\}$, $\{index\}$) *// Algorithm 6*

The *quick sort* algorithm (supplied by the C++ STL) is used for sorting the pixels of the images, which has a complexity of $O(N \log N)$. Since it must be applied to every frame, then its complexity

is $O(N_f N_p \log N_p)$, with N_f the number of frames and N_p the number of pixels per image. The result is obtained in the same array. This function is modified to provide the original position of each pixel in the input images (array *index*). The intensity values are smoothed at each position of the ordered images using a one dimensional temporal Gaussian convolution. We choose the Deriche's recursive filter [5] since it is fast, with a complexity of $O(N)$, and has a good accuracy as shown in [6] (we use the implementation given in this article). The total computational complexity of this step is $O(N_f N_p)$.

Algorithm 6: Apply Contrast Change

input : $\{\mathbf{I}\}$, $\{I_g\}$, $\{I_s\}$, $\{index\}$
output: $\{\mathbf{I}\}$
for $i \leftarrow 1$ **to** N_f **do**
  for $j \leftarrow 1$ **to** N_p **do**
    $\mathbf{I}(i, index(i,j)) \leftarrow gray2rgb(I_s(i,j)/I_g(i,j), \mathbf{I}(i, index(i,j)))$

After the images have been smoothed pixel by pixel, the intensities are rearranged using Algorithm 6. For grayscale images, this is a simple assignment of the intensities using the *index* array computed in the first step. For color images, we follow the same strategy as in the previous method [10, 11]: the colors are modified according to the rate of change between I_s and I_g.

The benefit of this technique is that the histograms of the resulting images are more similar. However, one of the drawbacks of this alternative is that the sorting of the pixels and the N_p convolutions with a Gaussian kernel have a higher computational cost than in the previous strategy. Another drawback is that pixels with the same initial intensity may get different values after equalization. Nevertheless, there are several strategies that take into account this effect to improve the equalization result; see [2] or [12], for instance.

In summary, the complexity of the algorithms are $O(N_f N_p \log N_p)$ and $O(N_f N_p)$, for the sorting and histogram methods, respectively; although with a much bigger constant in the sorting method. Note that both strategies are highly parallelizable, so the runtime is, in practice, inversely proportional to the number of cores.

3 Implementation Details

The implementation of the algorithm works with raw videos. This facilitates the task of reading and writing the information inside the program. The main procedure loads the whole video at the beginning and writes the output at the end. For converting a video to raw data, we use the `avconv` program from the command line as

```
avconv -i input_video -f rawvideo -pix_fmt rgb24 -y output_video.raw
```

This converts a video into raw data with 24 bits per pixel in RGB format, 8 bits per channel. The three components of the pixels are stored consecutively. The video can be directly read using an `unsigned char` array. This can be done in a single instruction with the `fread` function in C as

```
fread(I, sizeof(unsigned char), video_size, filename);
```

and written to disk with `fwrite` as

```
fwrite(I, sizeof(unsigned char), video_size, filename);
```

In a raw video, the information is stored in binary format without any header. This means that the size and the number of frames must be passed to the program. The following instruction shows how to call the program from the command line:

```
midway_video input_video.raw width height nframes [OPTIONS]
```

with `width` and `height` the size of the frames and `nframes` the number of frames. Additionally, we may specify the standard deviation for the Gaussian convolutions, with `-r 30` for example, and the use of the dithering method, with `-d`. If we want to convert the output raw video to any other format, we may also use the `avconv` program as

```
avconv -f rawvideo -pix_fmt rgb24 -video_size 'width'x'height' -framerate 'N' -i
       input_video.raw -pix_fmt yuv420p output_video
```

Note that it is necessary to specify the dimensions of the frames, `width x height`, and the frame rate, `N`. In the accompanying distributable, we include a script, called `cmdline_execute.sh`, that covers all the steps, from the conversion of the input video to raw data, the call to the midway equalization, and the conversion from the output raw data to a video format supported by `avconv`.

4 Experiments

In this section, we show several examples using our algorithms. The value of σ is 100 by default. This means that the equalization in every image takes into account the information of 201 frames. This large kernel will yield very similar histograms for all the images. However, if the contents of the video change, this may produce some problems, as in the last example.

In the first experiment, Figure 2, we show the equalization of a Chaplin's film, *The Cure*. The first row shows several frames of the original video and, the second, the equalized images with the histogram method. Figure 3 shows the corresponding histograms for these images and also for the images obtained by the sorting method. We observe that the midway equalization produces a more stable brightness range. The intensities of the resulting images are similarly distributed, as we can see in the histograms. The sorting method, in the third row, produces histograms which are more similar and homogeneous than in the histogram method. It also fills the regular empty values that we observe in the histograms of the original images.

In the second example, Figure 4, some quantization artifacts appear after the equalization. In the case of the histogram method, we observe a large white region in the first image of the second row. This method cannot cope with the saturation in the extreme values of the intensity range. In the sorting method, this saturation artifact does not appear.

Figure 5 shows the histograms for these images. The first histogram in the second row contains a large value in the last intensity bin, which corresponds to the large white region in Figure 4. The histogram method cannot distribute these values and its shape is very different from the other histograms. On the contrary, the histograms of the sorting method are much more homogeneous and similar, like in the previous example.

Figure 6 shows the results using the dithering option. The quantization artifacts are reduced. Nevertheless, in the case of the histogram method, we still observe the large white region. Increasing the amount of noise may still reduce these artifacts at the expense of reducing the quality of the images. In the sorting method, the use of dithering does not introduce any improvement.

The corresponding histograms for these images are given in Figure 7. Introducing a small amount of noise contributes to a better distribution of the intensities and the histograms become more similar in both methods.

Figure 2: Equalization of Chaplin's film *The Cure*: Top row, three frames of the original video; bottom row, the same frames after equalization using the histogram method.

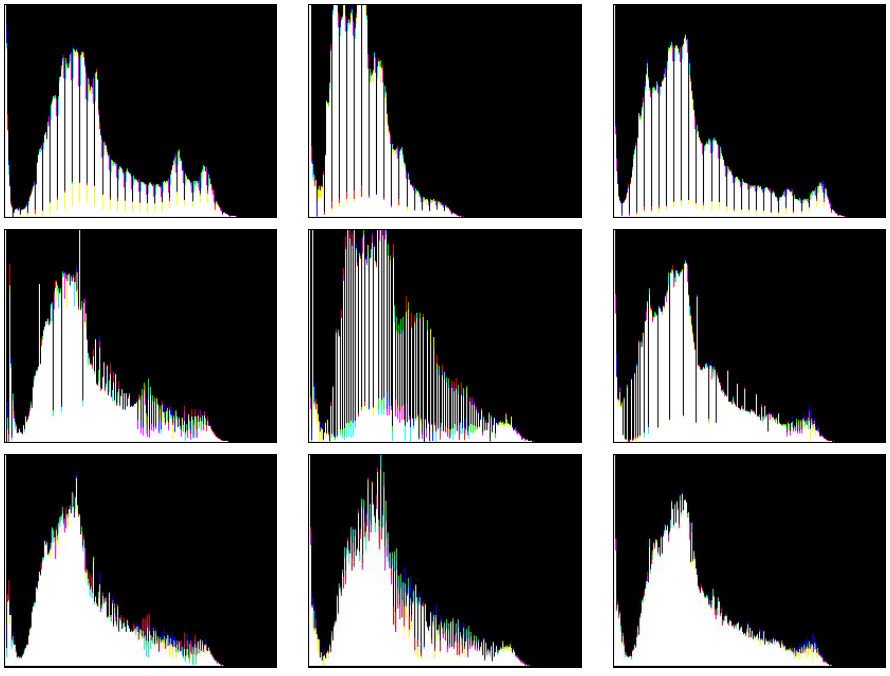

Figure 3: Histograms corresponding to the images in Figure 2. In the last row, the histograms of the images obtained by the sorting method.

In the third experiment, Figure 8, we observe the effect of midway equalization on color videos. The histograms of the second and third rows of Figure 9 show that the distribution of intensities is better in the equalized sequences. The solutions for the histogram and sorting methods are very similar.

The last experiment in Figures 10 and 11 shows a failure example. This is a sequence of a walking person where the scene changes drastically in appearance and lighting conditions. In the second row of Figure 10 and the first row of Figure 11, corresponding to the histogram and sorting methods, respectively, we show the result using a value of $\sigma := 30$. In that case, the equalization works

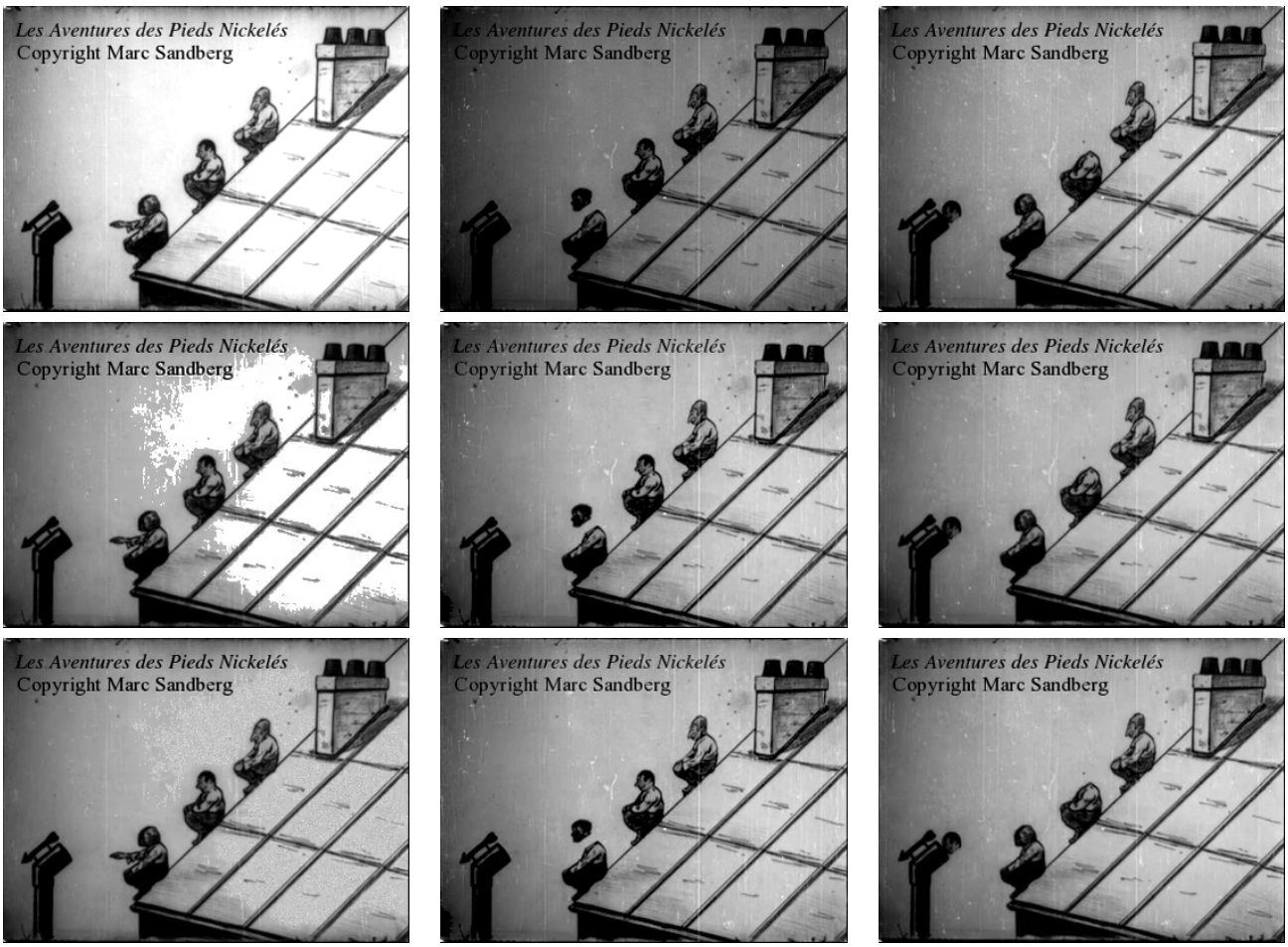

Figure 4: Equalization of *Pieds Nickelés* film: Top row, three frames of the original video; middle row, the same frames after equalization with the histogram method; bottom row, equalization using the sorting method. Notice the quantization artifacts that appear on the left image of the middle row.

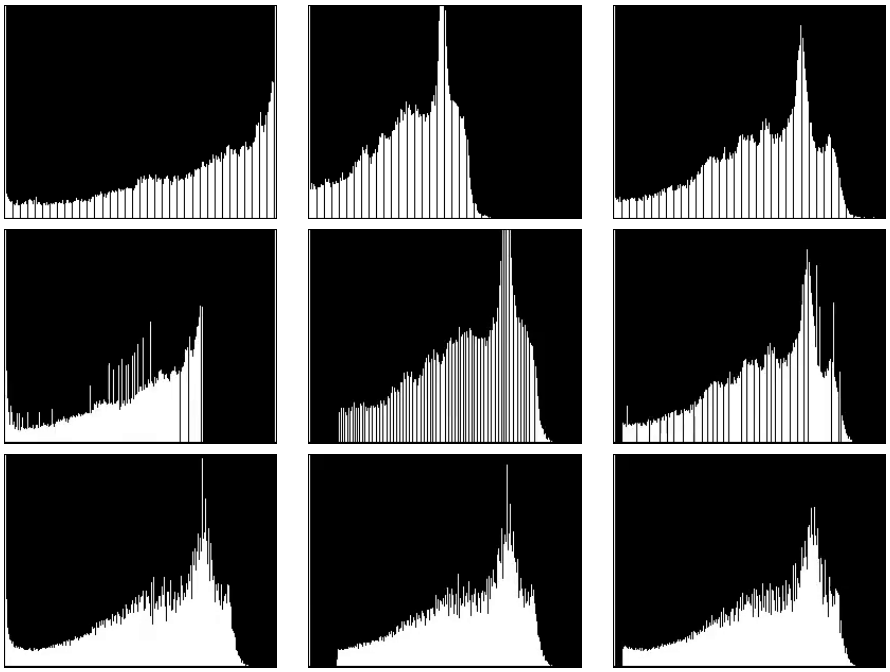

Figure 5: Histograms corresponding to the images in Figure 4.

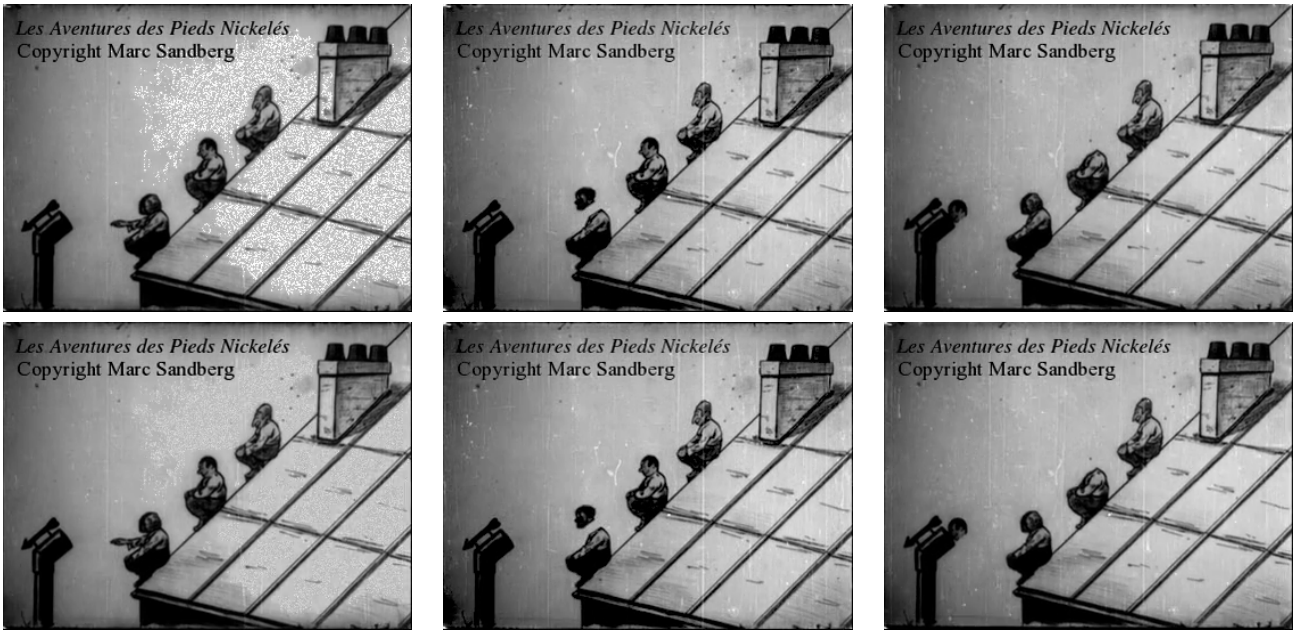

Figure 6: Equalization of *Pieds Nickelés* film using the dithering option: Top row, results obtained by the histogram method; bottom row, the results obtained by the sorting method. The quantization artifacts are still important for the first method, as we can see in the top–left image. The effect on the sorting method is not relevant in this case.

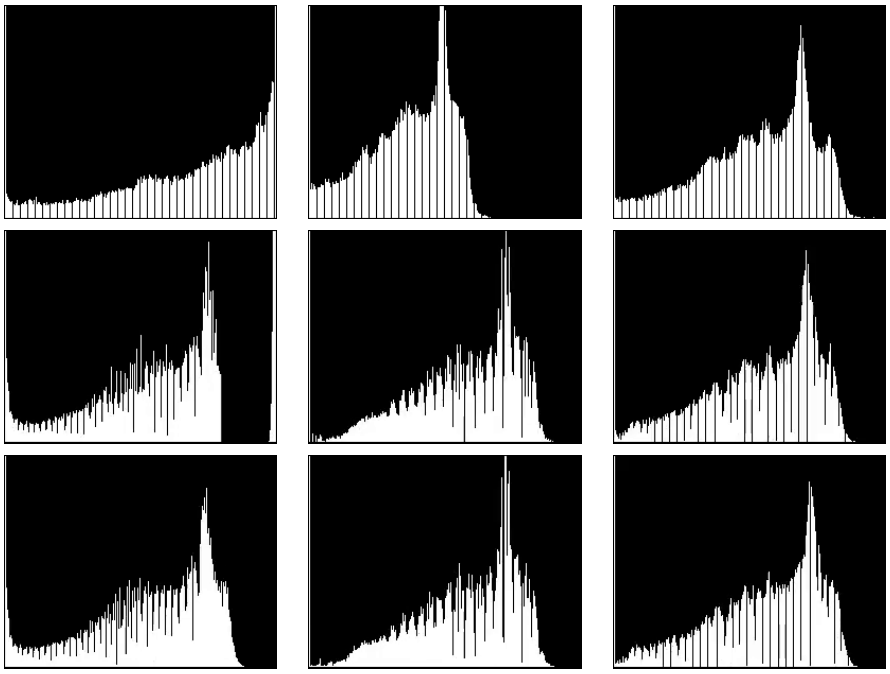

Figure 7: Histograms corresponding to the images in Figure 6. The top row corresponds to the histograms of the original images.

properly, although we start to observe some color changes in the sky. In the last row of both figures, $\sigma := 100$, there are severe color artifacts in the sky and asphalt, e.g., the blue color appearing in some parts of the first image.

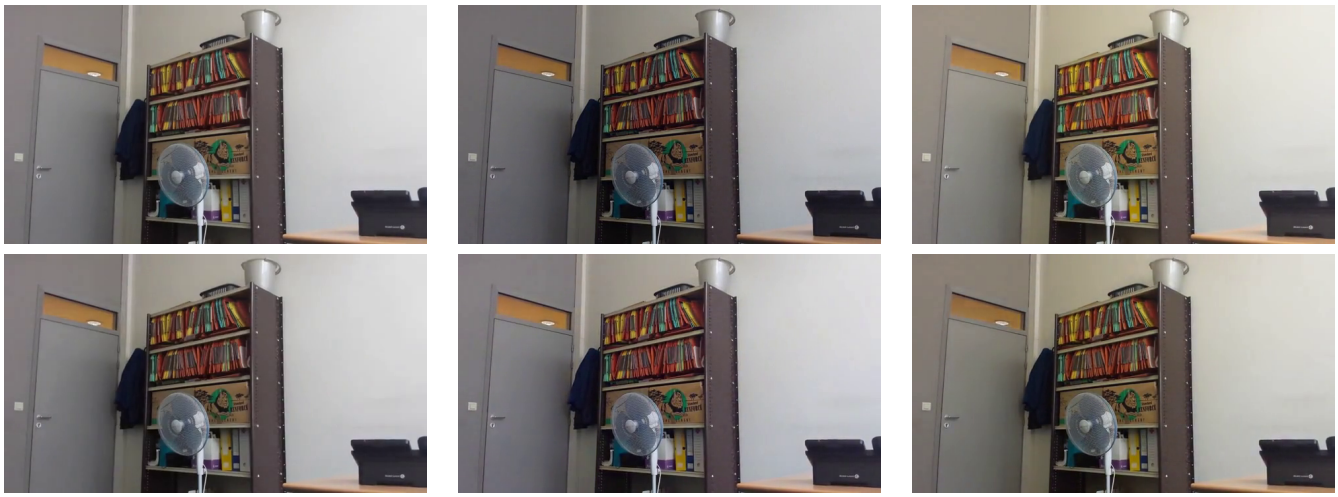

Figure 8: Equalization of the *Office* sequence: Top row, three frames of the original video; bottom row, the same frames after equalization with the histogram method.

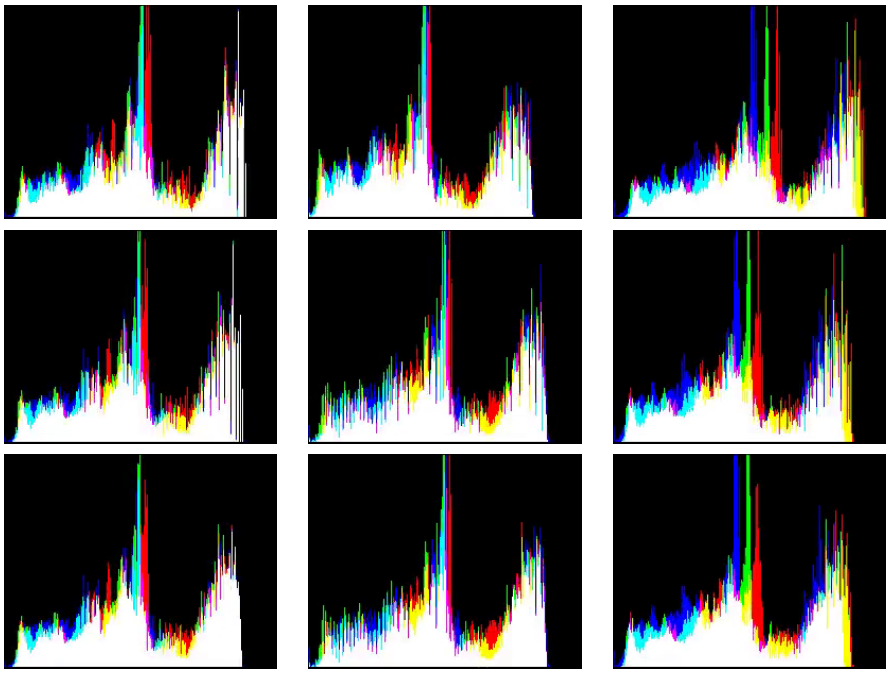

Figure 9: *Office* sequence: Histograms corresponding to the images in Figure 8. The third row contains the histograms of the sorting method.

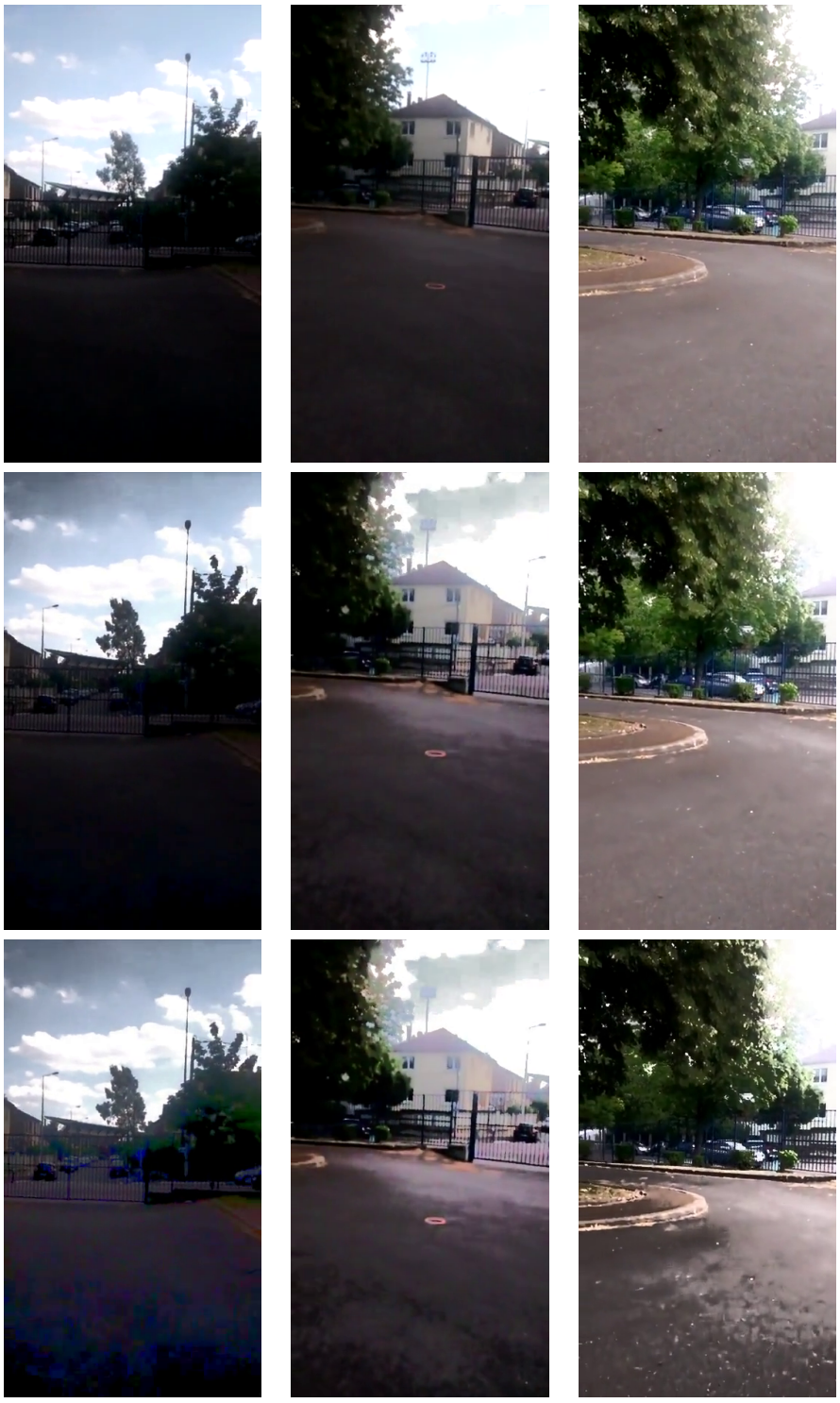

Figure 10: Equalization of the *Walking* sequence: Top row, three frames of the original video; middle row, frames after equalization with the histogram method and $\sigma := 30$; bottom row, frames after equalization with the same method and $\sigma := 100$. We observe that increasing the value of σ introduces important color artifacts.

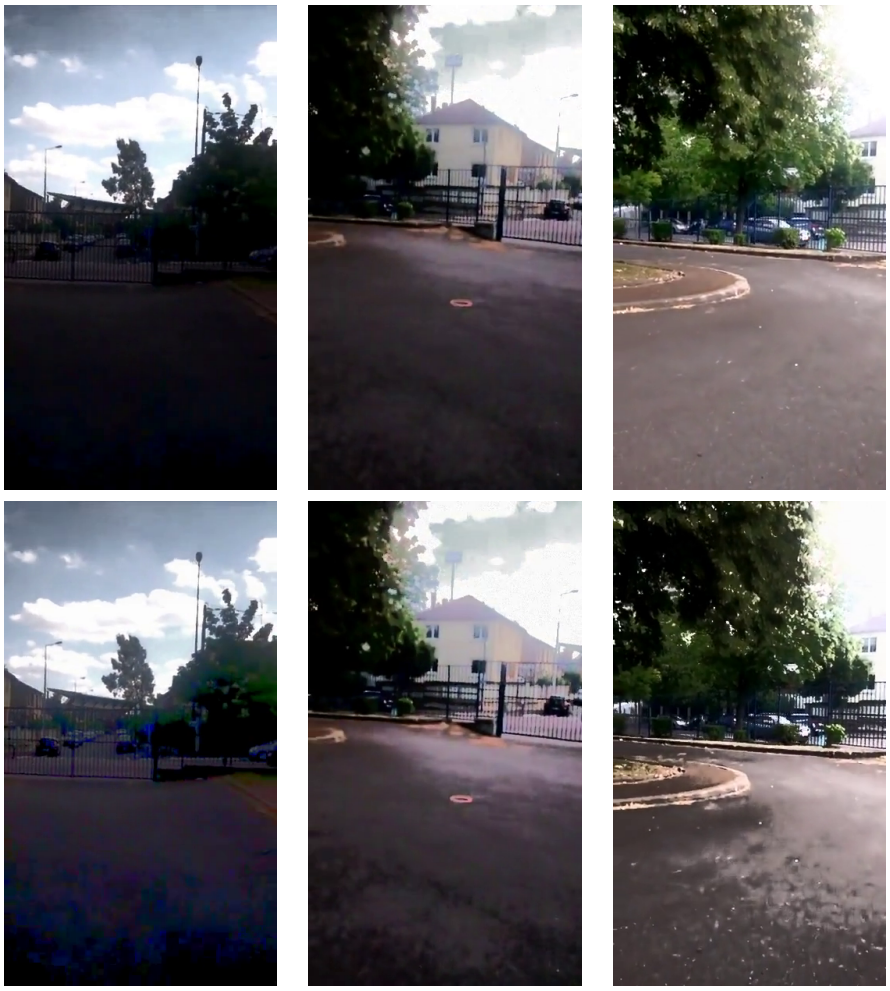

Figure 11: Equalization of the *Walking* sequence: Top row, equalization with the sorting method and $\sigma := 30$; bottom row, equalization with the same method and $\sigma := 100$. We observe similar artifacts as in the histogram method (see Figure 10).

Acknowledgment

This work has been partly founded by the BPIFrance and Région Ile de France, in the framework of the FUI 18 Plein Phare project, the European Research Council (advanced grant Twelve Labours) and the Office of Naval research (ONR grant N00014-14-1-0023).

Image Credits

All images by the author (license CC-BY-SA) except:

 Creative Commons

 copyright by Marc Sandberg

References

[1] V. Caselles, J. L. Lisani, J. M. Morel, and G. Sapiro, *Shape preserving local histogram modification*, IEEE Transactions on Image Processing, 8 (1999), pp. 220–230. http://dx.doi.org/10.1109/83.743856.

[2] D. Coltuc, P. Bolon, and J. M. Chassery, *Exact histogram specification*, IEEE Transactions on Image Processing, 15 (2006), pp. 1143–1152. http://dx.doi.org/10.1109/TIP.2005.864170.

[3] J. Delon, *Midway image equalization*, Journal of Mathematical Imaging and Vision, 21 (2004), pp. 119–134. http://dx.doi.org/10.1023/B:JMIV.0000035178.72139.2d.

[4] ——, *Movie and video scale-time equalization application to flicker reduction*, IEEE Transactions on Image Processing, 15 (2006), pp. 241–248. http://dx.doi.org/10.1109/TIP.2005.860328.

[5] R. Deriche, *Recursively implementating the Gaussian and its derivatives*, Research Report RR-1893, INRIA, 1993. https://hal.inria.fr/inria-00074778.

[6] P. Getreuer, *A Survey of Gaussian Convolution Algorithms*, Image Processing On Line, 3 (2013), pp. 286–310. http://dx.doi.org/10.5201/ipol.2013.87.

[7] R.C. Gonzalez and R.E. Woods, *Digital Image Processing*, Pearson, Inc., 3 ed., 2007. ISBN 013168728X.

[8] T. Guillemot and J. Delon, *Implementation of the Midway Image Equalization*, Image Processing On Line, 6 (2016), pp. 114–129. http://dx.doi.org/10.5201/ipol.2016.140.

[9] D. J. Jobson, Z. Rahman, and G. A. Woodell, *A multiscale retinex for bridging the gap between color images and the human observation of scenes*, IEEE Transactions on Image Processing, 6 (1997), pp. 965–976. http://dx.doi.org/10.1109/83.597272.

[10] N. Limare, J.L. Lisani, J-M. Morel, A.B. Petro, and C. Sbert, *Simplest Color Balance*, Image Processing On Line, 1 (2011). http://dx.doi.org/10.5201/ipol.2011.llmps-scb.

[11] J.L. Lisani, A.B. Petro, and C. Sbert, *Color and Contrast Enhancement by Controlled Piecewise Affine Histogram Equalization*, Image Processing On Line, 2 (2012), pp. 243–265. http://dx.doi.org/10.5201/ipol.2012.lps-pae.

[12] M. Nikolova, Y-W. Wen, and R. Chan, *Exact histogram specification for digital images using a variational approach*, Journal of Mathematical Imaging and Vision, 46 (2013), pp. 309–325. http://dx.doi.org/10.1007/s10851-012-0401-8.

[13] A.B. Petro, C. Sbert, and J-M. Morel, *Multiscale Retinex*, Image Processing On Line, 4 (2014), pp. 71–88. http://dx.doi.org/10.5201/ipol.2014.107.

[14] F. Pierre, J-F. Aujol, A. Bugeau, and V-T. Ta, *Luminance-Hue Specification in the RGB Space*, Springer International Publishing, 2015, pp. 413–424. http://dx.doi.org/10.1007/978-3-319-18461-6_33.

[15] S.M. Pizer, E.P. Amburn, J.D. Austin, R. Cromartie, A. Geselowitz, T. Greer, B.H. Romeny, J.B. Zimmerman, and K. Zuiderveld, *Adaptive histogram equalization and its variations*, Computer Vision, Graphics, and Image Processing, 39 (1987), pp. 355 – 368. http://dx.doi.org/10.1016/S0734-189X(87)80186-X.

[16] J. P. Rolland, V. Vo, B. Bloss, and C. K. Abbey, *Fast algorithms for histogram matching: Application to texture synthesis*, Journal of Electronic Imaging, 9 (2000), pp. 39–45. http://dx.doi.org/10.1117/1.482725.

[17] J. A. Stark, *Adaptive image contrast enhancement using generalizations of histogram equalization*, IEEE Transactions on Image Processing, 9 (2000), pp. 889–896. http://dx.doi.org/10.1109/83.841534.

[18] Q. Wang and R. K. Ward, *Fast image/video contrast enhancement based on weighted thresholded histogram equalization*, IEEE Transactions on Consumer Electronics, 53 (2007), pp. 757–764. http://dx.doi.org/10.1109/TCE.2007.381756.