

# Enhancement of images with uneven illumination

Enhancing images with uneven illumination using ensemble learning

Group ID: #1

Til Mohr  
tv.mohr@stud.uis.no

Alexander Mühleisen  
???

## ABSTRACT

## KEYWORDS

image processing, image enhancement, uneven illumination, ensemble learning

## 1 INTRODUCTION

## 2 THEORY

In this section we will dive into different methods to enhance images with uneven illumination. We will start with a brief introduction to the problem and then discuss different methods to solve it, as well as how to evaluate the results.

### 2.1 Problem description

Uneven illumination refers to the irregular distribution of light intensity across an image. In essence, it disrupts the uniformity of the visual output, leading to disparities in brightness and contrast, often observable as glares or shadows. These disparities can mask essential features and details, making the subject of the image less identifiable. This becomes especially problematic when images need to be processed further for various computer vision tasks. In fields like optical microscopy, for example, consistent illumination is crucial for accurately identifying and segmenting microscopic entities. Uneven lighting can obscure crucial cellular structures or make similar-looking entities appear distinct, hampering accurate analysis [2].

To counter this issue, the goal is to enhance the image in a manner that simulates its capture under uniform illumination conditions. By doing so, we aim to restore a natural appearance to the image, preserving details and minimizing artifacts introduced by uneven lighting. This correction enables better analysis, ensuring that conclusions drawn are based on the actual subject and not on lighting imperfections [2].

### 2.2 Unsharp Masking

Unsharp masking is a sharpening technique that uses a blurred version of the original image to enhance edges and fine details. The name stems from the fact that the blurred image is subtracted from the original, leaving only the high-frequency components, which are then added back to the original image. This results into an image with sharper edges, more pronounced detail, and more contrast. This approach can be formulated as follows [1, 4, 5]:

$$g(x, y) = f(x, y) + \lambda \cdot (f(x, y) - \text{Blur}(f)(x, y)) \quad (1)$$

where  $f(x, y)$  is the input image,  $\text{Blur}(f)(x, y)$  is the blurred input image, and  $\lambda > 0$  is a parameter that controls the strength of the sharpening effect. Typically, a Gaussian filter is used to blur the input image [1, 4, 5].

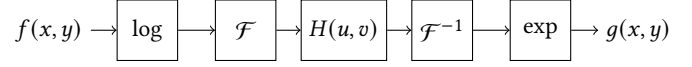


Figure 1: Homomorphic filtering pipeline.

### 2.3 Retinex

### 2.4 Homomorphic Filtering

The intensity of an image at pixel  $(x, y)$  can be described as the product of the illumination  $i(x, y)$  and the reflectance  $r(x, y)$  [3, 6]:

$$f(x, y) = i(x, y) \cdot r(x, y) \quad (2)$$

In the frequency domain, illumination changes across the image are typically manifested by low frequencies, while high frequencies are associated with reflectance changes. Therefore, by applying the logarithm to the image, one can separate the illumination and reflectance components of the image [3, 6]:

$$\log(f(x, y)) = \log(i(x, y)) + \log(r(x, y)) \quad (3)$$

Applying the Fourier transform to this log-image, a filter  $H(u, v)$  can be applied to attenuate the low frequencies, that is the frequencies responsible for illumination changes, and increasing the high frequencies responsible for detail. Afterwards, by applying the inverse Fourier transform and the exponential function, the image can be enhanced [3, 6]:

$$f(x, y) = \exp(\mathcal{F}^{-1}(\mathcal{F}(\log(f(x, y))) \cdot H(u, v))) \quad (4)$$

This process is illustrated in Figure 1.

Many approaches to the linear filter  $H(u, v)$  exist. Voicu et al. propose to use a second order Butterworth filter [6], to reduce the low frequencies and enhance the high frequencies:

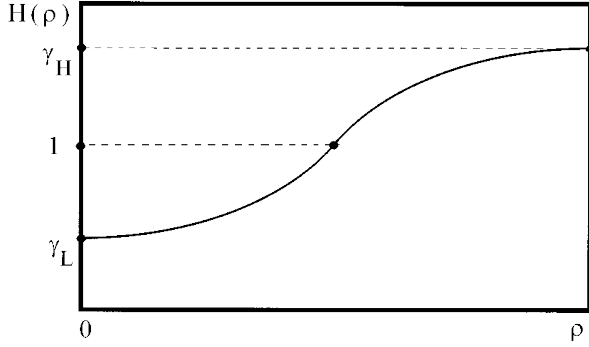
$$H(u, v) = H'(\rho) = \gamma_1 - \gamma_2 \cdot \frac{1}{1 + 2.415 \cdot \left(\frac{\rho}{\rho_c}\right)^4}, \quad (5)$$

$$\text{where } \rho = \sqrt{u^2 + v^2} \quad (6)$$

where  $\gamma_H, \gamma_L, \rho_c$  are parameters that can be tuned to achieve the desired effect, and  $\gamma_1 \approx \gamma_H, \gamma_2 \approx \gamma_H - \gamma_L$  [6]. The resulting filter has the general form shown in Figure 2.

Finally, Fan et al. [3] propose to append a histogram equalization step to the homomorphic filtering pipeline, in order to improve the contrast of the image.

In order to enhance colored images using homomorphic filtering, this pipeline can be applied to a single channel, e.g. the illumination channel of HSI images, or all channels, as in RGB images. [3, 6]. An implementation of this approach is shown in Listing 2.



**Figure 2: General form of the filter used in homomorphic filtering [6].**

### 2.5 Evaluation of Enhancement

The quality of the enhancement can be evaluated in a few different ways. If the image was enhanced simply to improve its visual appearance, visual inspection often suffices. On the other hand, if the image was enhanced as a preprocessing step for some other computer vision task such as segmentation, the quality of the enhancement should be evaluated by measuring the performance of the computer vision task on the enhanced image. However, there are also some objective metrics that can be used to get an idea of how well an image has been enhanced:

**2.5.1 RMS Contrast.** Contrast is a measure of the difference in brightness between the darkest and brightest parts of an image, i.e. it is a measure of how well objects are distinguishable. After enhancing an image with uneven illumination, we hope to increase the contrast in the areas of the image that originally had the same illumination. Therefore, an enhanced image might not experience a global increase in contrast, and rather some local increases. The RMS contrast is defined as the variance of the pixel intensities across the entire image [2]:

$$\text{RMS Contrast} = \frac{1}{N \cdot M} \sum_{i=1}^N \sum_{j=1}^M (I(i, j) - \bar{I})^2 \quad (7)$$

**2.5.2 Discrete Entropy.** Entropy describes the amount of information in an image, where a high entropy means that the image contains a lot of information, and a low entropy means that the image contains little information, i.e. a flat image has zero entropy. Enhancing an image with uneven illumination should increase the amount of information in the image, and therefore increase the entropy. The discrete entropy is defined as [2, 7]:

$$\text{Discrete Entropy} = - \sum_i P_i \cdot \log_2(P_i) \quad (8)$$

where  $P_i$  is the probability that the difference between two adjacent pixels is  $i$ .

## 3 METHODOLOGY

## 4 RESULTS

## 5 DISCUSSION AND CONCLUSIONS

## REFERENCES

- [1] Guang Deng. 2010. A generalized unsharp masking algorithm. *IEEE transactions on Image Processing* 20, 5 (2010), 1249–1261.
- [2] Nilanjan Dey. 2019. Uneven illumination correction of digital images: A survey of the state-of-the-art. *Optik* 183 (2019), 483–495.
- [3] Chun-Nian Fan and Fu-Yan Zhang. 2011. Homomorphic filtering based illumination normalization method for face recognition. *Pattern Recognition Letters* 32, 10 (2011), 1468–1479.
- [4] Koichi Morishita, Shimbu Yamagata, Tetsuo Okabe, Tetsuo Yokoyama, and Kazuhiko Hamatani. 1988. Unsharp masking for image enhancement. US Patent 4,794,531.
- [5] Zenglin Shi, Yunlu Chen, Efstratios Gavves, Pascal Mettes, and Cees GM Snoek. 2021. Unsharp mask guided filtering. *IEEE Transactions on Image Processing* 30 (2021), 7472–7485.
- [6] Liviu I Voicu, Harley R Myler, and Arthur Robert Weeks. 1997. Practical considerations on color image enhancement using homomorphic filtering. *Journal of Electronic Imaging* 6, 1 (1997), 108–113.
- [7] Zhengmao Ye, Habib Mohamadian, and Yongmao Ye. 2007. Discrete entropy and relative entropy study on nonlinear clustering of underwater and arial images. In *2007 IEEE International Conference on Control Applications*. IEEE, 313–318.

## A LISTINGS

### A.1 Unsharp Masking

```
1 import numpy as np
2 import cv2
3
4 def process_image(self, image: np.ndarray) -> np.ndarray:
5     blurred = cv2.GaussianBlur(image, (self.ksize, self.
6         ksize), 0)
7     sharpened = cv2.addWeighted(image, 1 + self.alpha,
8         blurred, -self.alpha, 0)
9     return sharpened
```

Listing 1: Unsharp masking

### A.2 Homomorphic Filtering

```
1 import numpy as np
2 import cv2
3 from util import BGR2HSI, HSI2BGR
4
5
6 def filter(value, gamma_1: float = 1.0, gamma_2: float =
7     0.6, rho: float = 2.0):
8     return gamma_1 - gamma_2 * (1 / (1 + 2.415 * np.power
9         (value / rho, 4)))
10
11 def process_image(image: np.ndarray) -> np.ndarray:
12     """
13     Process image using the model.
14
15     Parameters
16     -----
17     image : np.ndarray
18         Image to be processed, as BGR.
19
20     Returns
21     -----
22     np.ndarray
23         Processed image, as BGR.
24     """
25     # Convert image to HSI space
26     image = image.astype(np.float32)
27     hsi = BGR2HSI(image)
28
29     # Extract intensity channel and apply homomorphic
30     # filtering
31     i = hsi[:, :, 2]
32     i_log = np.log2(i + 1.0)
33     i_log_fft_shifted = np.fft.fftshift(np.fft.fft2(i_log
34         ))
35     i_log_fft_shifted_filtered = np.zeros_like(
36         i_log_fft_shifted)
37     for i in range(i_log_fft_shifted.shape[0]):
38         for j in range(i_log_fft_shifted.shape[1]):
39             i_log_fft_shifted_filtered[i, j] =
40                 i_log_fft_shifted[
41                     i, j
42                     ] * filter(
43                         np.sqrt(
44                             (i - i_log_fft_shifted.shape[0] / 2)
45                             ** 2
46                             + (j - i_log_fft_shifted.shape[1] /
47                                 2) ** 2
48                         )
49                     )
```

```
43     i_log_filtered = np.real(np.fft.ifft2(np.fft.
44         ifftshift(i_log_fft_shifted_filtered)))
45     i_filtered = np.exp2(i_log_filtered) - 1.0
46     # Replace intensity channel with filtered one
47     hsi_filtered = hsi.copy()
48     hsi_filtered[:, :, 2] = i_filtered
49
50     # Convert image back to BGR space
51     image = HSI2BGR(hsi)
52     image = np.clip(image, 0, 255)
53     image = image.astype(np.uint8)
54
55     # Equalize histogram of value channel
56     image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
57     image[:, :, 2] = cv2.equalizeHist(image[:, :, 2])
58
59     # Convert image back to BGR space
60     image = cv2.cvtColor(image, cv2.COLOR_HSV2BGR)
61     return image
```

Listing 2: Homomorphic filtering