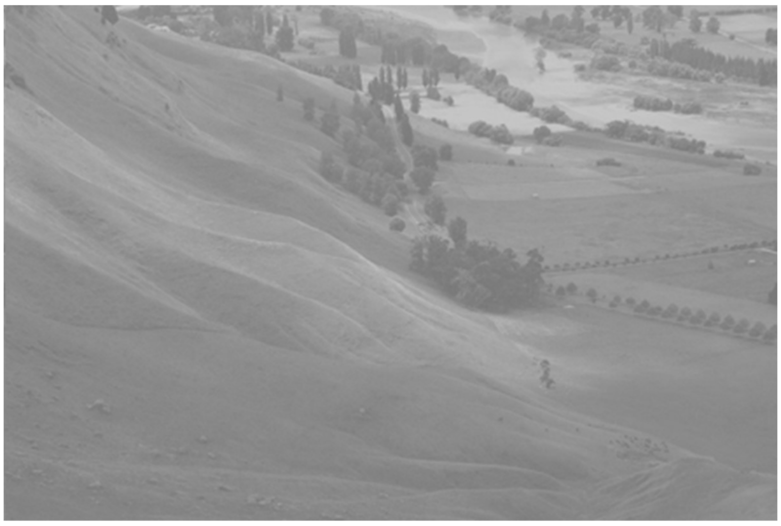
TP2 : Manipulation d’histogramme - Egalisation

How to Use Histogram Equalization

1. We first need to import :
- the OpenCV-Python package
- NumPy and Matplotlib to demonstrate the histogram equalization.

```
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt
```

Here is our test image that I named *vallee.png*



2. Assign a variable to the location of the image and then read the image.

COMPLETE CODE

```
path = "...\vallee.png"
```

3. Use the adequate method to show the image

```
plt.imshow(img,cmap="gray", vmin=0, vmax=255)
plt.show()
```

4. Now that our test image has been read, we can use the following code to view its histogram.

```
hist,bins = np.histogram(img.flatten(),256,[0,256])
cdf = hist.cumsum()
#cdf= Cumulative Distribution Function of an image
cdf_normalized = cdf * float(hist.max()) / cdf.max()
plt.plot(cdf_normalized, color = 'b')
plt.hist(img.flatten(),256,[0,256], color = 'r') plt.xlim([0,256])
plt.legend(('cdf','histogram'), loc = 'upper left')
plt.show()
```

Remark: Flattening is a technique that is used to convert multi-dimensional arrays into a 1-D array, Multi-Dimensional arrays take more amount of memory while 1-D arrays take less memory, which is the most important reason why we flatten the Image Array before processing.

COMPLETE with the Histogram of the Test Image

5. What are your observations regarding the histogram and image contrast?

COMPLETE CODE

6. We use OpenCV-Python's `.equalizeHist()` method to spread out the pixel intensity values. We assign the resulting image to the variable 'equ'.

```
equ = cv.equalizeHist(img)
```

7. Now, let's view this image and its histogram.

COMPLETE with code + Resulting image + new Histogram

Remark: Unlike the original histogram, the pixel intensity values now range from 0 to 255 on the X-axis. In a way, the original histogram has been stretched to the far ends. You may also notice that the cumulative distribution function (CDF) line is now linear as opposed to the original curved line.
==

In addition to the ordinary histogram equalization, there are two advanced histogram equalization techniques called:

- Adaptive Histogram Equalization

- Contrastive Limited Adaptive Equalization

==

Adaptive Histogram Equalization (AHE)

Unlike ordinary histogram equalization, adaptive histogram equalization uses the adaptive method to compute several histograms, each corresponding to a distinct section of the image. Using these histograms, this technique spread the pixel intensity values of the image to improve the contrast. Thus, adaptive histogram equalization is better than the ordinary histogram equalization if you want to improve the local contrast and enhance the edges in specific regions of the image.

8. With adaptive histogram equalization, we divide an input image into an M x N grid. We then apply equalization to each cell in the grid, resulting in a higher quality output image.

In this example, we use another library skimage:

```
import numpy as np
import matplotlib
import matplotlib.pyplot as plt

from skimage import data
from skimage.util import img_as_ubyte
from skimage import exposure
import skimage.morphology as morp
from skimage.filters import rank
from skimage.io import imsave, imread
```

```
# Original image
path = "...\vallee.png"
img = img_as_ubyte(imread(path, as_gray=True))

# Global equalize
img_global = exposure.equalize_hist(img)
# Local Equalization, disk shape kernel
# Better contrast with disk kernel but could be different
kernel = morp.disk(30)
img_local = rank.equalize(img, selem=kernel)
plt.rcParams['figure.figsize'] = [16, 16]
fig, (ax_img, ax_global, ax_local) = plt.subplots(1, 3)
ax_img.imshow(img, cmap=plt.cm.gray)
ax_img.set_title('Low contrast image')
ax_img.set_axis_off()
ax_global.imshow(img_global, cmap=plt.cm.gray)
ax_global.set_title('Global equalization')
ax_global.set_axis_off()
ax_local.imshow(img_local, cmap=plt.cm.gray)
ax_local.set_title('Local equalization')
ax_local.set_axis_off()
plt.show()
```

COMPLETE with CODE to plot resulting Histogram

Contrastive Limited Adaptive Equalization

Contrastive limited adaptive equalization (CLAHE) can be used instead of adaptive histogram equalization (AHE) to overcome its contrast overamplification problem. In CLAHE, the contrast implication is limited by clipping the histogram at a predefined value before computing the CDF. This clip limit depends on the normalization of the histogram or the size of the neighborhood region. The value between 3 and 4 is commonly used as the clip limit.

```
import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt

path = "...\vallee.png"
img = cv.imread(path)
grayimg = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
# create a CLAHE object (Arguments are optional).
clahe = cv.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))
cl1 = clahe.apply(grayimg)
#cv.imwrite('clahe_2.jpg',cl1)
plt.imshow(cl1, cmap="Greys_r")
plt.show()
```

9. Try tuning `createCLAHE` parameters and comment the results.