GPRP&O

Grupo de Pesquisa em Reconhecimento de Padrões e Otimização

CETELI

UFAM

DISCIPLINA: PROCESSAMENTO DIGITAL DE IMAGENS

Project number: 2

Course name: Computer Engineering

Student's name: João Victor Félix and Caroline Braz

Date due: 19/05/2024

Date handed in: 19/05/2024

Technical discussion and results

(One to three pages - max).

The following report aims to primarily report on the development of a project to explore the concepts learned in the course of Digital Image Processing. The project consists of elaborating algorithms/functions to reduce the number of intensities of an image. In the end, 7 new different images will be generated, reducing the intensities from 256 to 2 in integer powers of 2.

The first function called ***reduce_intensity*** receives as parameter an image and the level of intensity that is desired. It first calculates the value of denormalization, that is basically the value 255 (the max value considering that the original image has 256 levels of intensity) by the desired intensity level. Then it reduces the intensity level of the image dividing all the pixels of the original image by the denormalized value and converting the result for an int value (np.uint8). Finally, it normalizes the reduced image on the range of 0 and 255 using the **cv2.normalize** function.

```
# Função para reduzir a intensidade da imagem
def reduce_intensity(image, intensity_level):
      # Calcula o valor de desnormalização com base no nível de intensidade
      value_denormalized = 255 / intensity_level
      # Reduz o nível de intensidade da imagem, convertendo para np.uint8
      image_reduced_level = np.uint8(np.floor(np.double(image) / value_denormalized))
      # Normaliza a imagem reduzida para o intervalo de 0 a 255
      image_normalized = cv2.normalize(image_reduced_level, None, 0, 255,
norm_type=cv2.NORM_MINMAX)
      return image_normalized
```

In the main function, the image is loaded and converted to the grayscale using the function ***convert('L')***, then the **r.info.get('dpi')** save the DPI value of the original image and the image is converted to an Numpy matrix.

```
      # Abre a imagem usando PIL
      r = Image.open('drip-bottle-256.tif')

      # Converte a imagem para escala de cinza
      image_gray = r.convert('L')

      # Obtém o DPI original da imagem
      dpi_original = r.info.get('dpi')

      # Converte a imagem em escala de cinza para uma matriz Numpy
      image_matrix = np.asarray(image_gray)
```

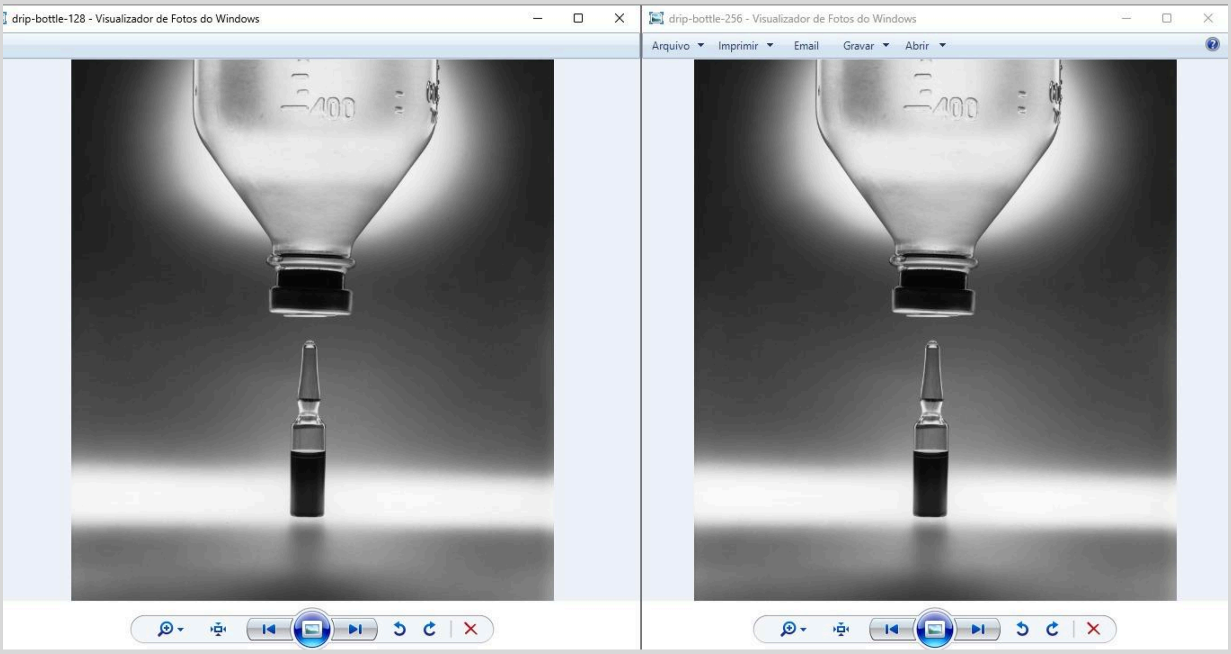
The next algorithm iterates the value of k between 0 and 7. Then it calculates the level of intensity raising the number 2 to the value of k. Finally, the function ***reduce_intensity*** is called passing the image and the calculated value of k as parameters. Considering that the return of that function is a matrix, it's necessary to convert it back to an image using the function Image.fromarray and finally save the final picture with the new level of intensity but with the original dpi.

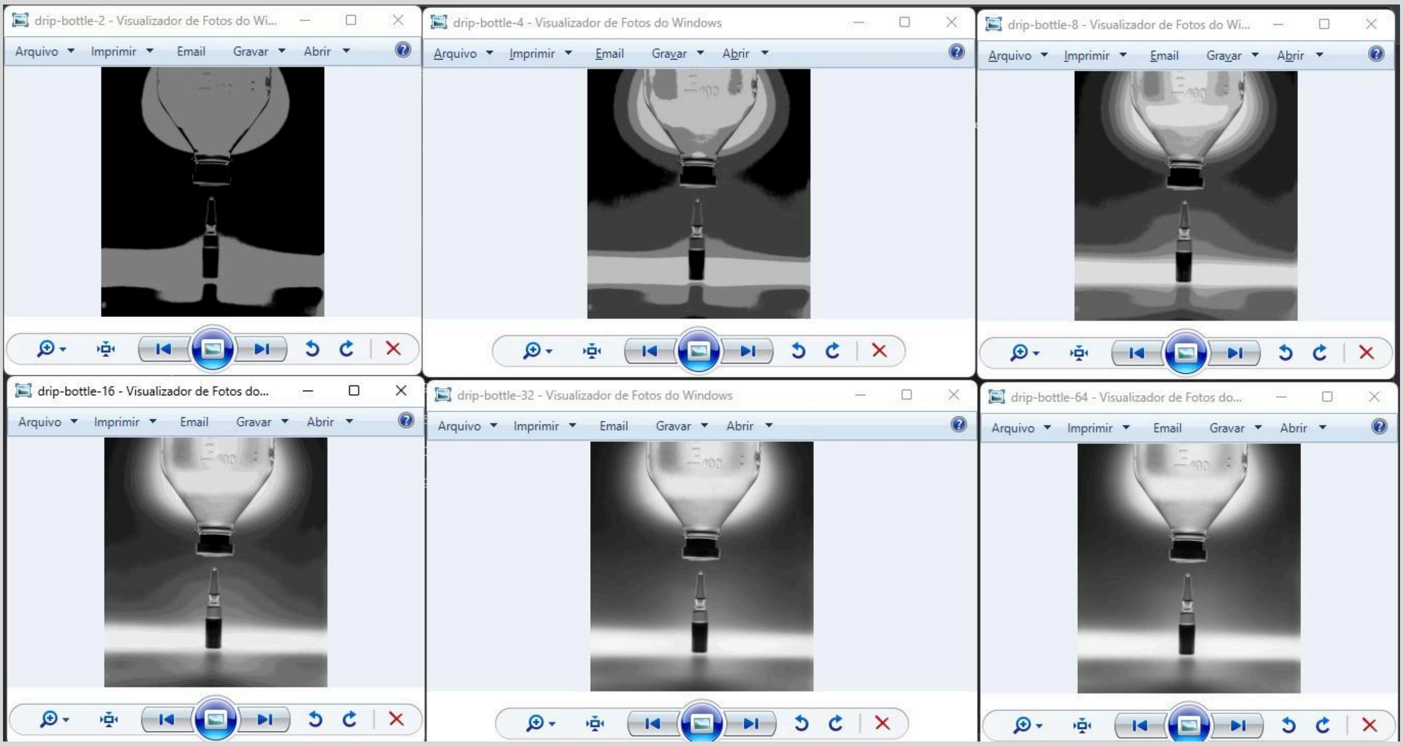
```
# Itera sobre os valores de k de 1 a 7 (total de 8 níveis de intensidade)
for k in range(8):
      intensity_level = 2 ** k  # Calcula o nível de intensidade como 2 elevado a k

      # Chama a função de redução de intensidade e recebe a imagem normalizada
      matrix_result = reduce_intensity(image_matrix, intensity_level)
      # Transforma a matriz resultante de volta em um objeto de imagem
      s = Image.fromarray(matrix_result)
      # Define o DPI alvo como o valor original do DPI
      dpi_target = dpi_original

      # Salva a imagem resultante com o nível de intensidade ajustado
      s.save('drip-bottle-' + str(intensity_level) + '.tif', optimize=False,
dpi=dpi_target)
```

The final results are 8 pictures with 256, 128, 64, 32, 16, 4 and 2 levels of intensity. As a conclusion, it is possible to notice that the fewer intensity levels (grayscale levels), the more visible the points of color change became, because the markings became increasingly divided. This continued until reaching level 2, which is a binary image, where it is clear that there are only 2 tones, black and light gray.


drip-bottle-128 - Visualizador de Fotos do Windows
drip-bottle-256 - Visualizador de Fotos do Windows
Arquivo
Imprimir
Email
Gravar
Abrir
400
400


drip-bottle-2 - Visualizador de Fotos do Wi...
Arquivo
Imprimir
Email
Gravar
Abrir
drip-bottle-4 - Visualizador de Fotos do Windows
drip-bottle-8 - Visualizador de Fotos do Wi...
drip-bottle-16 - Visualizador de Fotos do...
drip-bottle-32 - Visualizador de Fotos do Windows
drip-bottle-64 - Visualizador de Fotos do...

Through this experiment, it was possible to observe that the level of detail that can be found in an image with a higher number of grayscale levels is much greater. We can separate and observe more details because it becomes easier to differentiate each element of the image.

References

(submit here the references cited).

Python. “The Python Standard Library — Python 3.8.1 Documentation.” Python.org, 2020, docs.python.org/3/library/index.html.

Guimarães, Marly. “UNIDADE III -Fundamentos das transformações de intensidade e de filtragem espacial.” *Google Docs*, https://drive.google.com/file/d/1NPXd7K9xMn-C82uqfxAVnKnfLWrDwUtF/view.

Spring. Disponível em: <http://www.dpi.inpe.br/spring/portugues/tutorial/introducao_pro.html>.

Villares, Alexandre. Disponível em: <http://abav.lugaralgum.com/material-aulas/Processing-Python/pixels.html>.

Image module. Disponível em: <http://pillow.readthedocs.io/en/stable/reference/Image.html>.

Gonzalez, R C & **Woods**, R E. Digital Image Processing 2018 – 4th edition Prentice Hall.