**Photo Processing HW 1**

**运行结果**

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  |  |

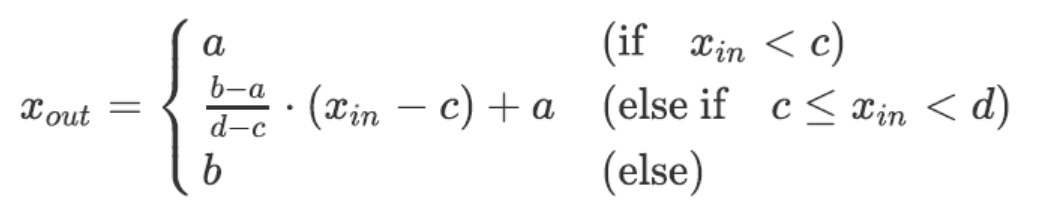
|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  |  |

**文字分析**

略，代码呈现的已经相对比较完整了，可以很简单的从代码中了解具体的思路

其中牵涉到的一个相对比较奇怪的地方在于归一灰度直方图的生成过程，使用了这样的公式，达成了对应的效果，除此之外，本文中牵涉到的其他内容复杂度均相对较低，因此不做额外说明。



**全部内容的源代码**

|  |
| --- |
| Python from matplotlib import pyplot as plt import numpy as np  def show\_original\_image(image: np.ndarray) -> None:  plt.imshow(image.astype('int'))  # plt.suptitle('original image')  plt.show()  def show\_gray\_image(image: np.ndarray) -> None:  gray = get\_gray\_image(image)  plt.imshow(gray.astype('int'))  plt.show()  def show\_binary\_image(image: np.ndarray) -> None:  bin = get\_binary\_image(image)  plt.imshow(bin.astype('int'))  plt.show()  def add\_and\_show\_two\_image(image\_a: np.ndarray, image\_b: np.ndarray) -> None:  plt.imshow(image\_a.astype('int'), alpha=0.5)  plt.imshow(image\_b.astype('int'), alpha=0.5)  plt.show()  # add\_image = image\_a + image\_b  # add\_image = normalization(add\_image, left=0, right=255)  # plt.imshow(add\_image.astype('int'))  # plt.show()  # Method of image convert  def get\_gray\_image(image: np.ndarray) -> np.ndarray :  Y = image[:,:,0] \* 0.299 + image[:,:,1] \* 0.587 + image[:, :, 2] \* 0.114  gray = np.zeros((Y.shape[0], Y.shape[1], 3))  gray[:,:,0] = gray[:,:,1] = gray[:, :, 2] = Y # create a image with same value   return gray  def get\_binary\_image(image: np.ndarray) -> np.ndarray:  # convert to grey  Y = image[:,:,0] \* 0.299 + image[:,:,1] \* 0.587 + image[:, :, 2] \* 0.114  # binaryzation  Y[Y<=128] = 0  Y[Y>128] = 255  # putting it back into a image to show   gray = np.zeros((Y.shape[0], Y.shape[1], 3))  gray[:,:,0] = gray[:,:,1] = gray[:, :, 2] = Y # create a image with same value   return gray  # Method about statistics  def get\_grey\_level\_histogram(image: np.ndarray) -> None:  plt.hist(image.ravel(), bins=256, fc='k', ec='k')  plt.show()  def get\_hist\_normalization\_image(image: np.ndarray) -> np.ndarray:  normalizaiton\_hist: np.ndarray = normalization(image)  plt.hist(normalizaiton\_hist.ravel(), bins=256, fc='k', ec='k')  plt.show()  #[deprecated] def normalize(\_d, to\_sum=True, copy=True):  # d is a (n x dimension) np array  d = \_d if not copy else np.copy(\_d)  d -= np.min(d, axis=0)  d /= (np.sum(d, axis=0) if to\_sum else np.ptp(d, axis=0))  return d   def normalization(img: np.ndarray, left=0, right=1) -> np.ndarray:  """  convert img[min, max] to [left, right]   min ( if x < c )  ( right - left ) / ( max - min ) \* ( x - min ) + left ( elif min <= x < max)  max ( else )  """  # init  min, max = img.min(), img.max()  out = img.copy()   # normalization  out = ( right - left ) / ( max - min ) \* ( out - min ) + left  out[out < left] = left  out[out > right] = right   return out  if \_\_name\_\_ == "\_\_main\_\_":  path = "./HW1./OIP-C.jpg"  x = plt.imread(path)  plt.imshow(x.astype('int'))  plt.show()   # # Question 1  show\_gray\_image(x)  show\_binary\_image(x)   # # Question 2  get\_grey\_level\_histogram(x)  get\_hist\_normalization\_image(x)   # # Question 3  background = plt.imread("./HW1./background.jpg")  chicken = plt.imread("./HW1./chicken.jpg")  add\_and\_show\_two\_image(background, chicken) |

最新版本实验报告：https://e3vqhv9jh5.feishu.cn/docx/doxcnAr018n4zQPnfXtojLzTcsh