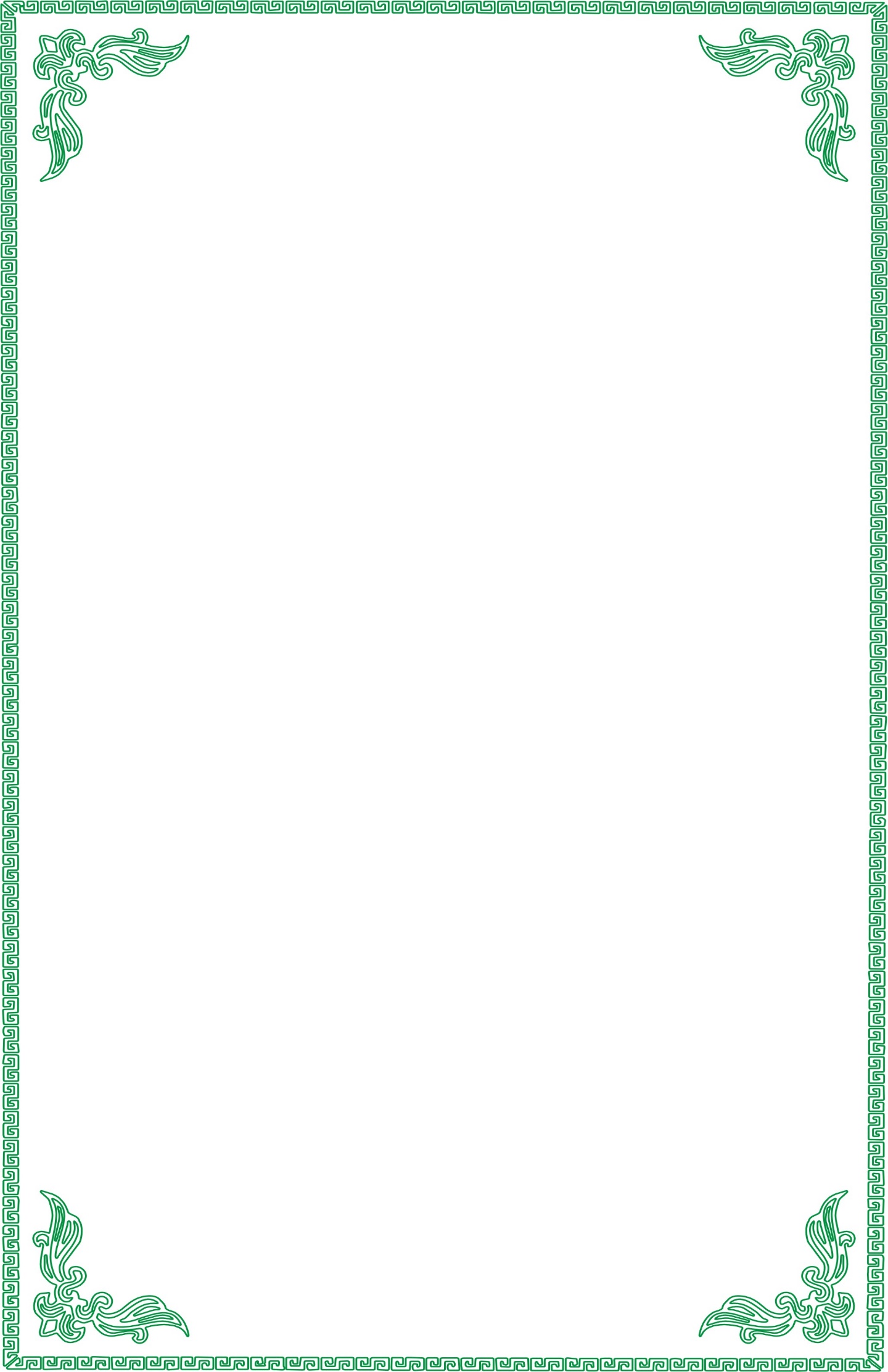
****

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA KHOA HỌC MÁY TÍNH**

**MÔN: TÍNH TOÁN ĐA PHƯƠNG TIỆN**



**BÁO CÁO ĐỒ ÁN CUỐI KỲ**

**VINTAGE-STYLE IMAGE TRANSFER**

**Giảng viên: Nguyễn Vinh Tiệp**

**Sinh viên thực hiện**:

17520324 – Nguyễn Thành Danh

17520828 – Phan Nguyên

Nội dung Báo cáo

[I. Define a problem (Phát biểu bài toán) 3](#_Toc10491752)

[1. Task (Mục đích) 3](#_Toc10491753)

[2. Input (Đầu vào) 3](#_Toc10491754)

[3. Output (Đầu ra) 3](#_Toc10491755)

[II. Relevant Knowledges (Các kiến thức liên quan cần biết) 3](#_Toc10491756)

[1. Ảnh Grayscale, RGB 3](#_Toc10491757)

[2. Histogram 4](#_Toc10491758)

[III. Idea (Ý tưởng) 5](#_Toc10491759)

[IV. Tools and Libraries (Các công cụ sử dụng) 5](#_Toc10491760)

[Các thư viện đã sử dụng: 5](#_Toc10491761)

[V. Implement (Lập trình cài đặt) 7](#_Toc10491762)

[1. Khai báo thư viện 7](#_Toc10491763)

[2. Main Source 7](#_Toc10491764)

[3. Demo kết quả: 11](#_Toc10491765)

[i. Landscape (Ảnh phong cảnh): 11](#_Toc10491766)

[ii. Poitrait (Ảnh chân dung): 12](#_Toc10491767)

[4. Giải thích sơ bộ: 13](#_Toc10491768)

[VI. Explanation (Giải thích) 13](#_Toc10491769)

[1. Phương pháp đánh giá Bhattacharya: 13](#_Toc10491770)

[2. Histogram in Grayscale – Compare original image and result image: 13](#_Toc10491771)

[3. Histogram in RGB – Compare original image and result image: 16](#_Toc10491772)

[4. Histogram of Original Image 22](#_Toc10491773)

[5. Histogram of Result Image 23](#_Toc10491774)

[VII. Conclusions and References: (Rút ra kết luận và Nguồn tham khảo) 25](#_Toc10491775)

[1. Conclusions: 25](#_Toc10491776)

[2. References: 25](#_Toc10491777)

# Define a problem (Phát biểu bài toán)

## Task (Mục đích)

Chuyển đổi một tấm ảnh chụp bình thường thành một tấm ảnh có style vintage.

## Input (Đầu vào)

Một tấm ảnh bất kỳ mà ta muốn chuyển đổi style.

## Output (Đầu ra)

Tấm ảnh đó với màu sắc mang style vintage (bạc màu, có nhiễu, có hiệu ứng cũ kĩ).

# Relevant Knowledges (Các kiến thức liên quan cần biết)

## Ảnh Grayscale, RGB

* Ảnh grayscale là ảnh mà tại mỗi điểm ảnh có 1 giá trị mức xám, ví dụ ảnh mức xám 8 thì giá trị từ 0 đến 7, ảnh mức xám 256 thì giá trị mức xám từ 0 đến 255…



Figure 1 Thang mức xám

Với mức xám 256, giá trị điểm ảnh bằng 0 nghĩa là điểm ảnh đó đen, giá trị điểm ảnh bằng 255 nghĩa là điểm ảnh đó trắng. Tức là giá trị mỗi điểm ảnh càng lớn thì điểm ảnh càng sáng.

* Ảnh RGB là ảnh mà tại mỗi điểm ảnh có 3 kênh màu Red, Green, Blue tạo thành.

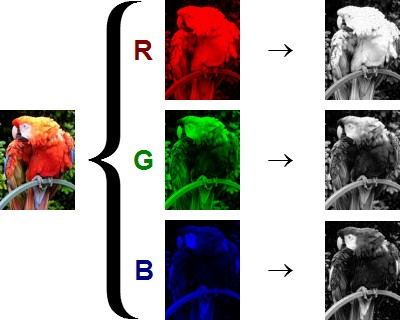


Figure 2 Ví dụ ảnh RGB và grayscale tương ứng mỗi kênh màu

## Histogram

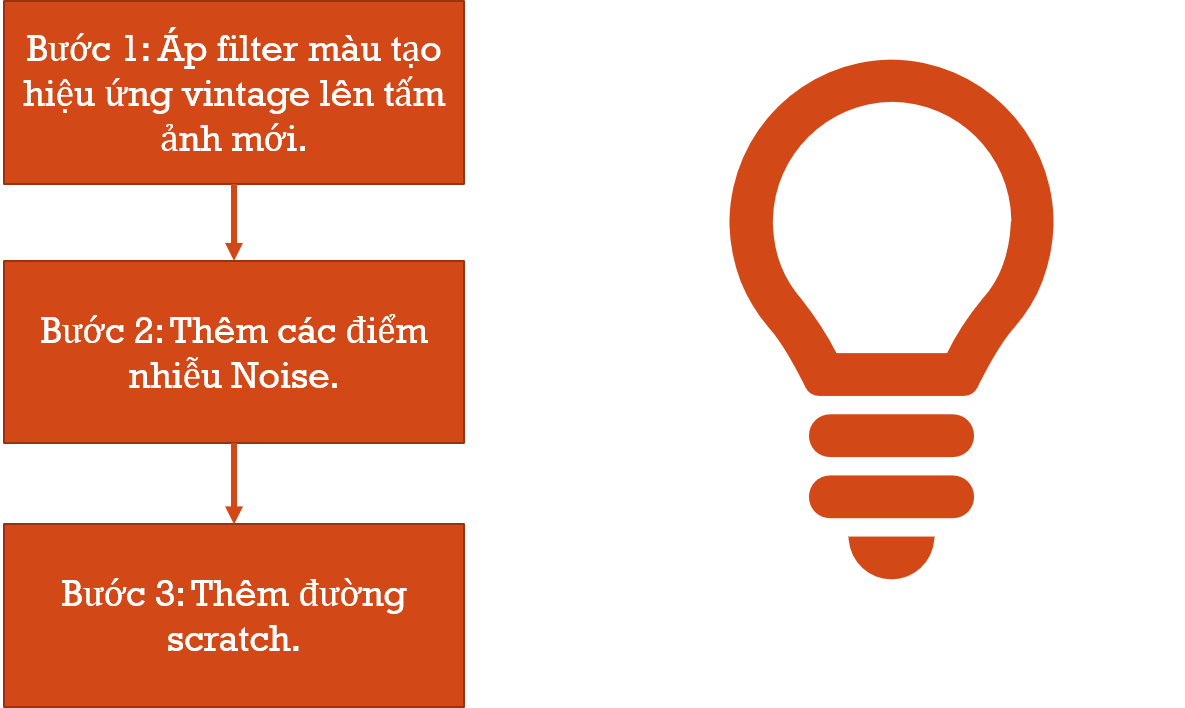
* Histogram là biểu đồ phân phối tấn số xuất hiện của dữ liệu.
* Image histogram là biểu đồ phân phối tấn số xuất hiện giá trị điểm ảnh của ảnh.

A screenshot of a cell phone

Description automatically generated

Figure 3 Ví dụ về histogram của grayscale

# Idea (Ý tưởng)



# Tools and Libraries (Các công cụ sử dụng)

## Các thư viện đã sử dụng:

1. **NumPy**: (viết tắt của Numerical Python) là một thư viện rất cần thiết khi chúng ta xây dựng các ứng dụng Máy học trên Python. Numpy cung cấp các đối tượng và phương thức để làm việc với mảng nhiều chiều và các phép toán đại số tuyến tính. Nó là một mô-đun mở rộng mã nguồn mở cho Python, cung cấp các chức năng biên dịch nhanh cho các thao tác toán học và số. Hơn nữa, NumPy làm phong phú ngôn ngữ lập trình Python với các cấu trúc dữ liệu mạnh mẽ để tính toán hiệu quả các mảng và ma trận đa chiều.

**Các hàm cơ bản:**

* ***import numpy as np:*** import thư viện.
* ***np.zeros(shape, dtype):*** trả về một mảng mới có hình dạng đã cho(shape) chứa các giá trị 0.
* ***np.sum(array):***trả về tổng các phần từ trong mảng array.

1. **Thư viện PIL:**

PIL là thư viện hỗ trợ việc xử lý hình ảnh của Python. Thư viện này hỗ trợ nhiều định dạng tập tin, và cung cấp khả năng xử lý hình ảnh và đồ hoạ mạnh mẽ.

**Các hàm cơ bản**

* ***from PIL import Image:*** import thư viện.
* **Tạo ảnh**
* ***img = Image.new("RGB", (256, 256), "white")***

Imaging cho phép tạo một hình ảnh trong Python với đầy đủ kích thước, màu sắc,...

*hoặc*

* *from PIL import Image #import thư viện*
* ***im = Image.new("RGB", (256, 256), (0, 0, 255))***
* **Mở ảnh**
* ***img = Image.open("path-to-image")***
* **Hiển thị ảnh**
* ***img.show()***
* **Lưu ảnh**
* ***img.save(outfile, format, options…)***

***outfile****: Tên tập tin hoặc đối tượng tập tin.*

***format****: Định dạng tuỳ chọn. Nếu bỏ qua, các định được xác định từ phần mở rộng của tên tập tin.*

***options****: Thông số bổ sung.*

* ***img.save("image.png", "PNG")***

1. **OpenCV:** là một thư viện mã nguồn mở hàng đầu cho thị giác máy tính (computer vision), xử lý ảnh và máy học, và các tính năng tăng tốc GPU trong hoạt động thời gian thực.OpenCV được phát hành theo giấy phép BSD, do đó nó hoàn toàn miễn phí cho cả học thuật và thương mại. Nó có các interface C++, C, Python, Java và hỗ trợ Windows, Linux, Mac OS, iOS và Android. OpenCV được thiết kế để tính toán hiệu quả và với sự tập trung nhiều vào các ứng dụng thời gian thực. Được viết bằng tối ưu hóa C/C++, thư viện có thể tận dụng lợi thế của xử lý đa lõi. Được sử dụng trên khắp thế giới, OpenCV có cộng đồng hơn 47 nghìn người dùng và số lượng download vượt quá 6 triệu lần. Phạm vi sử dụng từ nghệ thuật tương tác, cho đến lĩnh vực khai thác mỏ, bản đồ trên web hoặc công nghệ robot.

**Các hàm cơ bản:**

* import cv2
* ***cv2.imread(path):*** *Đọc ảnh từ đường dẫn path.*
* ***cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY****)****:*** *chuyển đổi từ ảnh RGB sang ảnh Grayscale.*
* ***cv2.addWeighted(array\_1, weight\_of\_array\_1, array\_2, weight\_of\_array\_2):*** *tính toán tổng 2 ma trận theo trọng số.*

1. **Các thư viện khác:**

Bên cạnh đó nhóm cũng có sử dụng một số thư viện như:

* Random
* Tkinter
* Math
* Matplotlib

Các thư viện này hỗ trợ một số chức năng nhỏ trong khi cài đặt và không giữ vai trò chủ đạo của đồ án.

# Implement (Lập trình cài đặt)

## Khai báo thư viện

|  |
| --- |
| from PIL import Image as IM  import random  import cv2  import numpy as np  import tkinter as tk  from tkinter import filedialog |

## Main Source

|  |
| --- |
| # this filter is to reduce the appearance of blue color and increase those of red and green, for the image looks like older  VINTAGE\_COLOR\_LEVELS = {  'r': [0, ..., 255, 255],  'g' : [0, ..., 254, 255],  'b' : [53, ... , 198, 199]  }  # apply changes to all pixels of the image  def modify\_all\_pixels(im, adjust\_levels):  width, height = im.size # take sizes of the image  pxls = im.load() # load value of pixels  for x in range(width):  for y in range(height):  # at pixel [x,y] we have 3 values of each color channel, one by one is red - green - blue  # adjust\_levels fuction get 5 parameters and return 3 values corresponding to r-g-b at pixel[x, y]  pxls[x, y] = adjust\_levels(x, y, pxls[x, y][0], pxls[x, y][1], pxls[x, y][2])  # apply color\_map to transfer image to vintage style  def vintage\_colors(im,color\_map=VINTAGE\_COLOR\_LEVELS):  r\_map = color\_map['r']  g\_map = color\_map['g']  b\_map = color\_map['b']  def adjust\_levels(x, y, r, g, b):  return r\_map[r], g\_map[g], b\_map[b] # return values of r-g-b from color map  modify\_all\_pixels(im, adjust\_levels) # update all pixels with values from the color map  return im  # add noise randomly  def add\_noise(im, noise\_level=50):  def pixel\_noise(x, y, r, g, b):  noise = random.randint(0, noise\_level) - noise\_level//2 # noise generation  return max(0, min(r + noise, 255)), max(0, min(g + noise, 255)), max(0, min(b + noise, 255))  modify\_all\_pixels(im, pixel\_noise) # update all pixels after generating noise  return im  def add\_scratch(im):  test\_vintage = np.array(im) #convert pil image to matrix format preparing for cv2 format cvt  scratch\_ = cv2.imread('scratch.jpg') #335 x 507  scratch = cv2.resize(scratch\_, (test\_vintage.shape[1],test\_vintage.shape[0])) # resize scratch image to fit original image  x\_ = cv2.addWeighted(test\_vintage, 0.8, scratch, 0.2, 0) # blend original img and scratch img with rate 8:2  # cv2.imwrite('test\_vintage\_scratch\_rgb2.jpg', x)  # cv2.imshow("s", x)  im\_pil = IM.fromarray(x\_) #convert back to pil format  return im\_pil  def main():  root = tk.Tk()  root.title("Vintage Style Transfer v0.0")  root.geometry("400x250")  root.configure(background='black')  def close\_window():  root.destroy()  def choose\_file():  root.filename = filedialog.askopenfilename(initialdir = "/",title = "Select file",filetypes = (("jpeg files","\*.jpg"),("all files","\*.\*")))  print(root.filename)  def transfer():  filename = root.filename  image = IM.open(filename)  # print(type(image))  image.show(title = "original")  if image.mode != 'RGB':  image = image.convert('RGB')  vintage\_colors(image)  add\_noise(image)  image\_vintage\_final = add\_scratch(image)  base\_path, ext = filename.rsplit('.', 1)  image\_vintage\_final.save('%s\_vintage\_scrt.%s' % (base\_path, ext))  image\_vintage\_final.show(title = "vintage")  lbl = tk.Label(root, text = "\nVINTAGE STYLE TRANSFER\n",bg = "black", fg = "white", font = ('Helvetica', 18))  lbl.pack(side = "top")  btn\_choosefile = tk.Button(root, text = "Choose Image", bg = "white", fg = "black", width = 20, height = 1, font = ('Arial', 15), command = choose\_file)  btn\_choosefile.pack(side = "top")  btn\_choosefile = tk.Button(root, text = "Start Transfer", bg = "white", fg = "black", width = 20, height = 1, font = ('Arial', 15), command = transfer)  btn\_choosefile.pack(side = "top")  btn = tk.Button(root, text = "Close", bg = "white", fg = "black", width = 20, height = 1, font = ('Arial', 15), command = close\_window)  btn.pack(side = "top")  root.mainloop()  if \_\_name\_\_ == '\_\_main\_\_':  main() |

## Demo kết quả:

### **Landscape (Ảnh phong cảnh):**

**Input:A large brick building with grass and trees

Description automatically generated**

Figure V‑1 Demo Input Image

**Output:A large brick building

Description automatically generated**

Figure V‑2 Demo Result Image

### **Poitrait (Ảnh chân dung):**

**Input:**

**A person posing for the camera

Description automatically generated**

Hình 1 Demo input Image (ảnh lấy từ picswe.com)

**Output:**

**A person wearing glasses

Description automatically generated**

Hình 2 Demo output Image

## Giải thích sơ bộ:

* **Nhận xét:** Tấm ảnh kết quả sẽ cho ra cảm nhận có hiệu ứng ảnh cũ, thêm vào các điểm bị nhiễu và các điểm ảnh bị mất nét (vintage style).
* **Giải thích:** Thêm vào ảnh các hiệu ứng trên là do nguyên liệu để làm giấy là từ gỗ tự nhiên, trải qua quá trình sản xuất thành phẩm, gỗ biến thành giấy mà chúng ta đang sử dụng hiện nay. Hiện tượng giấy in, sách báo sau 1 thời gian sử dụng sẽ ố vàng do tác động của oxy trong không khí (Oxy hóa).

# Explanation (Giải thích)

## Phương pháp đánh giá Bhattacharya:

Để xác định sự khác biệt giữa ảnh gốc và ảnh sau khi đã chuyển đổi, chúng em sử dụng đơn vị đo Bhattacharya. Hệ số Bhattacharya là một phép đo gần đúng về số lượng trùng lặp giữa hai mẫu thống kê. Hệ số có thể được sử dụng để xác định độ gần tương đối của hai mẫu được xem xét.

BC (P, Q) =

* Với P, Q là các tập mẫu (1 lớp của ảnh).
* n là số lượng mẫu.
* Pi và Qi là giá trị tương ứng thứ i của P và Q.

## Histogram in Grayscale – Compare original image and result image:

|  |
| --- |
| import numpy as np  import matplotlib.pyplot as plt  import cv2  import math  # to read the two images here and convert to grayscale  img\_ = cv2.imread('test\_\_vintage\_scrt.jpg') # load an image with the original color on it  img = cv2.cvtColor(img\_, cv2.COLOR\_BGR2GRAY) # convert the color image to gray scale  img2\_ = cv2.imread('test\_.jpg', 1) # load an image with the original color on it  img2 = cv2.cvtColor(img2\_, cv2.COLOR\_BGR2GRAY) # convert the color image to gray scale  # initialize two 1D array of size 256 elements for both pictures  array1 = np.zeros((256))  array2 = np.zeros((256))  rows1, columns1 = img.shape # pixel location determined by row x column  rows2, columns2 = img2.shape # pixel location of the second image by row x column  # next two double for loops is to determine the histograms for each picture img and img2  # which are the histograms for vintage and original stored in array1 and array2  for i in range(rows1):  for j in range(columns1):  pixel = img[i,j] # obtain the pixel value of that at location i,j  array1[pixel] += 1 # increment the histogram[pixel] location by 1  for a in range(rows2):  for b in range(columns2):  pixel1 = img2[a,b] # obtain the pixel value of that at location a,b  array2[pixel1] += 1 # increment the histogram[pixel2] location by 1  plt.plot(array1) # using this for testing if my histograms are correct  plt.plot(array2)  plt.xlim([0,256])  plt.show()  # get the total sum of each array1 and array2 separately, we need this  # to do the next step to normalize the histogram  arr\_sum1 = int(math.floor(np.sum(array1)))  arr\_sum2 = int(math.floor(np.sum(array2)))  # normalize each histogram (i.e. array1 and array2 are histograms, so all the  # values in both arrays now are divided by there total sum therefore obtaining decimals)  for item in range(len(array1)):  array1[item] = array1[item] / arr\_sum1  for items in range(len(array2)):  array2[items] = array2[items] / arr\_sum2  # using the Bhattacharya Coefficient formula to calculate 1 for a match, 0 for not  # a match, but closely resembling each other is a high decimal number  array3 = np.zeros((256)) # use this container to see the sum of array1 and array2  for index in range(256):  array3[index] = math.sqrt(array1[index] \* array2[index])  #b\_coefficient = math.floor(np.sum(array3))  b\_coefficient = np.sum(array3)  print("The Bhattacharya Coefficient is " + str(b\_coefficient)) |

**A close up of a device

Description automatically generated**

Figure VI‑1 The Bhattacharya Coefficient is 0.84083405328705

Đường màu cam: Histogram in Grayscale của ảnh gốc.

Đường màu xanh: Histogram in Grayscale của ảnh kết quả.

* **Nhận xét:** Đối với Histogram của Grayscale thì 2 tấm ảnh thể hiện sự khác nhau rõ ràng nhất ở đoạn từ 0 đến 50 và từ 220 đến 255, các phần còn lại có nhiều nét tương đồng cho nên độ đo Bhattacharya Coefficient đạt giá trị tương đối cao 84%.
* **Giải thích:** Với ảnh đã được convert sang Grayscale thì tương đối giống nhau nên ít phân biệt được ảnh mới hay cũ, vì vậy độ đo Bhattacharya Coefficient tương đối ổn.

## Histogram in RGB – Compare original image and result image:

|  |
| --- |
| import numpy as np  import matplotlib.pyplot as plt  import cv2  import math  # to read the two images here and convert to grayscale  img\_ = cv2.imread('test\_\_vintage\_scrt.jpg') # load an image with the original color on it  img\_ = cv2.cvtColor(img\_, cv2.COLOR\_BGR2RGB)  img\_r = img\_[:,:,0]  img\_g = img\_[:,:,1]  img\_b = img\_[:,:,2]  img2\_ = cv2.imread('test\_.jpg', 1) # load an image with the original color on it  img2\_ = cv2.cvtColor(img2\_, cv2.COLOR\_BGR2RGB)  img2\_r = img2\_[:,:,0]  img2\_g = img2\_[:,:,1]  img2\_b = img2\_[:,:,2]  # initialize 6 1D array of size 256 elements for both pictures day and night  array1\_r = np.zeros((256))  array1\_g = np.zeros((256))  array1\_b = np.zeros((256))  array2\_r = np.zeros((256))  array2\_g = np.zeros((256))  array2\_b = np.zeros((256))  rows1\_r, columns1\_r = img\_r.shape  rows1\_g, columns1\_g = img\_g.shape  rows1\_b, columns1\_b = img\_b.shape  rows2\_r, columns2\_r = img2\_r.shape  rows2\_g, columns2\_g = img2\_g.shape  rows2\_b, columns2\_b = img2\_b.shape  for i in range(rows1\_r):  for j in range(columns1\_r):  pixel = img\_r[i,j]  array1\_r[pixel] += 1  for i in range(rows1\_g):  for j in range(columns1\_g):  pixel = img\_g[i,j]  array1\_g[pixel] += 1    for i in range(rows1\_b):  for j in range(columns1\_b):  pixel = img\_b[i,j]  array1\_b[pixel] += 1    for a in range(rows2\_r):  for b in range(columns2\_r):  pixel1 = img2\_r[a,b]  array2\_r[pixel1] += 1    for a in range(rows2\_g):  for b in range(columns2\_g):  pixel1 = img2\_g[a,b]  array2\_g[pixel1] += 1    for a in range(rows2\_b):  for b in range(columns2\_b):  pixel1 = img2\_b[a,b]  array2\_b[pixel1] += 1  plt.figure(figsize = (15, 8))    #red  plt.subplot(311)  plt.title('Red-layer Comparision')  plt.plot(array1\_r)  plt.plot(array2\_r)  plt.gca().legend(('Vintage Img','Original Img'))  plt.xlim([0,256])  arr\_sum1\_r = int(math.floor(np.sum(array1\_r)))  arr\_sum2\_r = int(math.floor(np.sum(array2\_r)))  for item in range(len(array1\_r)):  array1\_r[item] = array1\_r[item] / arr\_sum1\_r  for items in range(len(array2\_r)):  array2\_r[items] = array2\_r[items] / arr\_sum2\_r  array3\_r = np.zeros((256))  for index in range(256):  array3\_r[index] = math.sqrt(array1\_r[index] \* array2\_r[index])  b\_coefficient\_r = np.sum(array3\_r)  print("The Bhattacharya Coefficient Red is " + str(b\_coefficient\_r))  #green  plt.subplot(312)  plt.title('Green-layer Comparision')  plt.plot(array1\_g)  plt.plot(array2\_g)  plt.xlim([0,256])  arr\_sum1\_g = int(math.floor(np.sum(array1\_g)))  arr\_sum2\_g = int(math.floor(np.sum(array2\_g)))  for item in range(len(array1\_g)):  array1\_g[item] = array1\_g[item] / arr\_sum1\_g  for items in range(len(array2\_g)):  array2\_g[items] = array2\_g[items] / arr\_sum2\_g  array3\_g = np.zeros((256))  for index in range(256):  array3\_g[index] = math.sqrt(array1\_g[index] \* array2\_g[index])  b\_coefficient\_g = np.sum(array3\_g)  print("The Bhattacharya Coefficient Green is " + str(b\_coefficient\_g))  #blue  plt.subplot(313)  plt.title('Blue-layer Comparision')  plt.plot(array1\_b)  plt.plot(array2\_b)  plt.xlim([0,256])  arr\_sum1\_b = int(math.floor(np.sum(array1\_b)))  arr\_sum2\_b = int(math.floor(np.sum(array2\_b)))  for item in range(len(array1\_b)):  array1\_b[item] = array1\_b[item] / arr\_sum1\_b  for items in range(len(array2\_b)):  array2\_b[items] = array2\_b[items] / arr\_sum2\_b  array3\_b = np.zeros((256))  for index in range(256):  array3\_b[index] = math.sqrt(array1\_b[index] \* array2\_b[index])  b\_coefficient\_b = np.sum(array3\_b)  print("The Bhattacharya Coefficient Blue is " + str(b\_coefficient\_b))  plt.show() |

**A screenshot of a cell phone

Description automatically generated**

Table 1 Bhattacharya Coefficient Comparision R-G-B layers

|  |  |  |
| --- | --- | --- |
| **The Bhattacharya Coefficient for Red layer** | **The Bhattacharya Coefficient for Green layer** | **The Bhattacharya Coefficient for Blue layer** |
| 0.792558132499811 | 0.8385072678731701 | 0.5719918419784089 |

* **Nhận xét:** Đối với histogram của Red-layer và Green-layer thì các giá trị màu tương đối đồng đều. Nhưng Blue-layer thì lại chênh lệch rõ ràng hơn nên độ đo BC đạt thấp hơn hẳn.
* **Giải thích:** Sự khác biệt rõ nhất giữa ảnh cũ và ảnh hiện đại là ở kênh màu Blue, vì ảnh cũ thường có tone màu ám vàng (kênh Red và kênh Green), số lượng điểm ảnh có giá trị từ 0 đến 50 và 200 đến 250 của kênh Blue sẽ giảm xuống, thay vào đó là số điểm ảnh có giá trị 50 đến 200 sẽ tăng lên (giảm mức độ quá sáng hoặc quá tối của kênh Blue).

## Histogram of Original Image

|  |
| --- |
| def histogram(img, channel):  img = img[:, :, channel]  freq = np.zeros((256,))  for i in range(img.shape[0]):  for j in range(img.shape[1]):  freq[img[i, j]] += 1  return freq  Original\_ = cv2.imread('test\_\_vintage\_scrt.jpg')  Original = cv2.cvtColor(Original\_, cv2.COLOR\_BGR2RGB)  hist\_r = histogram(Original, 0)  hist\_g = histogram(Original, 1)  hist\_b = histogram(Original, 2)  # Show result  f, axarray = plt.subplots(1, 4, figsize = (24,9))  f.tight\_layout()  axarray[0].set\_title('Image')  axarray[0].imshow(Original)  axarray[1].set\_title('Histogram R')  axarray[1].bar(np.arange(256), hist\_r, color='r')  axarray[2].set\_title('Histogram G')  axarray[2].bar(np.arange(256), hist\_g, color='g')  axarray[3].set\_title('Histogram B')  axarray[3].bar(np.arange(256), hist\_b, color='b')  plt.show() |

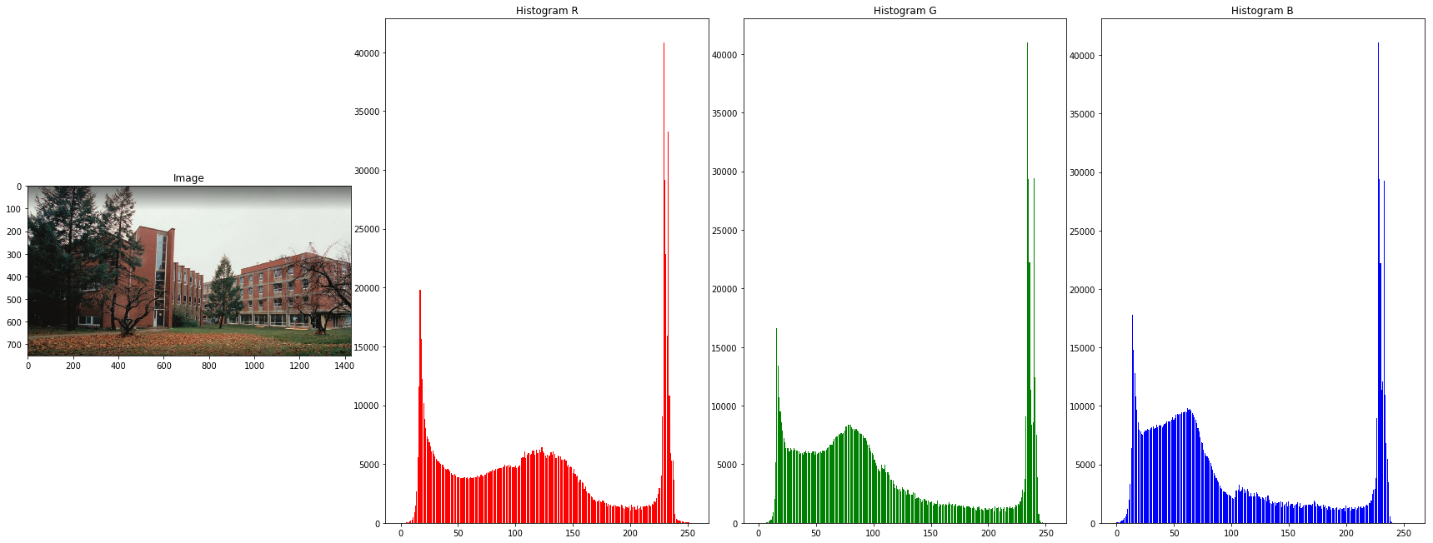


Figure VI‑2 Histogram of Original Image

* **Nhận xét:** 3 kênh R-G-B có mật độ phân phổi tương đối ổn định và giống nhau.
* **Giải thích:** Bởi vì đây là ảnh ban đầu, nên các giá trị màu của 3 kênh màu phân bố đồng đều.

## Histogram of Result Image

|  |
| --- |
| Original\_ = cv2.imread('test\_.jpg')  Original = cv2.cvtColor(Original\_, cv2.COLOR\_BGR2RGB)  hist\_r = histogram(Original, 0)  hist\_g = histogram(Original, 1)  hist\_b = histogram(Original, 2)  # Show result  f, axarray = plt.subplots(1, 4, figsize = (24,9))  f.tight\_layout()  axarray[0].set\_title('Image')  axarray[0].imshow(Original)  axarray[1].set\_title('Histogram R')  axarray[1].bar(np.arange(256), hist\_r, color='r')  axarray[2].set\_title('Histogram G')  axarray[2].bar(np.arange(256), hist\_g, color='g')  axarray[3].set\_title('Histogram B')  axarray[3].bar(np.arange(256), hist\_b, color='b')  plt.show() |

A close up of a logo

Description automatically generated

Figure VI‑3 Histogram of Result Image

* **Nhận xét:** Kênh Red và Green có mật độ phân phổi tương đối ổn định hơn so với kênh Blue. Đối với kênh Blue, thì giá trị các điểm ảnh chỉ nằm trong đoạn từ 50 đến 200.
* **Giải thích:** Áp dụng histogram của ảnh có phong cách cũ cũng có nghĩa là chuyển màu nền ảnh ngả vàng, theo hệ màu RGB, thì màu vàng được tạo ra nhờ kết hợp giữa ít đỏ và nhiều xanh lá, nên ta thấy histogram của xanh dương sẽ không còn ổn định nữa. Mà số điểm màu từ 0 đến 50 và từ 200 đến 255 giảm xuống, số điểm màu từ 50 đến 200 tăng lên làm cho màu xanh dịu lại, không còn ám xanh nhiều như các ảnh hiện đại.

A close up of a logo

Description automatically generated

Figure VI‑4 Color Mixture

# Conclusions and References: (Rút ra kết luận và Nguồn tham khảo)

## Conclusions:

Trong phạm vi đồ án môn học, nhóm đã thành công với mục đích của đề tài là: với ảnh input bất kỳ, chuyển đổi ảnh đó thành một ảnh có phong cách xưa cũ (vintage); tạo được giao diện đơn giản giúp người dùng dễ dàng tương tác với ứng dụng.

Hướng phát triển: phát triển thành ứng dụng di động độc lập, một web-app hoặc trở thành một tính năng của các ứng dụng chỉnh sửa ảnh đang có trên thị trường hiện nay.

## References:

1. <https://github.com/rendro/vintageJS/>
2. <https://machinelearningcoban.com/2017/10/12/fundaml_vectors/>
3. <https://docs.scipy.org/doc/numpy/index.html>
4. <https://docs.opencv.org/3.4/>
5. <https://www.python-course.eu/python_tkinter.php>

**--//--**

**Đồ Án Cuối Kỳ - Môn Tính Toán Đa Phương Tiện**

Giảng viên: Nguyễn Vinh Tiệp

Sinh viên thực hiện: 17520324 – Nguyễn Thành Danh

17520828 – Phan Nguyên