ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KHOA HỌC MÁY TÍNH

BÁO CÁO ĐỒ ÁN

MÔN NHẬP MÔN THỊ GIÁC MÁY TÍNH

Đề tài: Các hiệu ứng chỉnh sửa ảnh trong ứng dụng PicsArt

GVHD: Nguyễn Vinh Tiệp

Nhóm sinh viên thực hiện: 25

1. Nguyễn Hoàng Thắng	MSSV: 18521394
2. Phan Quang Tấn	MSSV: 18521377
3. Trần Ngọc Sương	MSSV : 18521353

□□ Tp. Hồ Chí Minh, 12/2020 □□

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

.....

Mục Lục

CHƯƠNG 1 : LÝ DO CHỌN ĐỀ TÀI , HIỆN TRẠNG

1. **Đề tài :**

   Các hiệu ứng chỉnh sửa ảnh trong PicsArt.

2. **Lý do chọn đề tài :**

   Việc phát triển ứng dụng chỉnh sửa ảnh và video (cụ thể là PicsArt) đang được các nhà phát triển ứng dụng chú trọng phát triển . Khi ứng dụng của bạn có các chức năng dễ sử dụng , các hiệu ứng đẹp mặt , ấn tượng thì thị trường người dùng sẽ phát triển nhanh chóng thông qua hiệu ứng đám đông . Từ đó, ứng dụng sẽ đem lại cho nhà phát triển một nguồn doanh thu khổng lồ. Mà phần quan trọng nhất quyết định đến sự phát triển của ứng dụng này là các hiệu ứng chỉnh sửa .

3. **Hiện trạng :**

   Những năm gần đây, trên các mạng xã hội nổi lên trào lưu “sống ảo”, thu hút hàng loạt người dùng. Trào lưu này chưa có dấu hiệu hạ nhiệt mà đang ngày một phát triển mạnh mẽ, song song với đó là sự cạnh tranh ngầm giữa những người dùng cũng dần rõ rệt hơn, thể hiện qua sự đầu tư trau chuốt cho từng tấm ảnh được đăng lên mạng. Hay đơn giản hơn, cũng có những người dùng chỉ muốn lưu lại những khoảnh khắc cuộc sống của riêng mình một cách thật “nghệ”, cái mà chỉ với một cái nháy máy ảnh không thể làm được. Chính vì điều đó, các ứng dụng chỉnh sửa ảnh lần lượt ra đời, với những hiệu ứng đặc sắc, vui nhộn mà cũng rất nghệ thuật. Điểm nhấn của những ứng dụng này đó là sự đơn giản, giao diện thân thiện với người dùng, và chỉ với vài phút thao tác là người dùng đã có thể hô biến những tấm ảnh theo ý muốn của bản thân. Và với những đặc điểm như trên, PicArts trở thành một trong những ứng dụng được lòng rất nhiều người dùng, đặc biệt đối với các bạn trẻ. Nắm bắt được điều đó, nhóm chúng em đã quyết định tìm hiểu về ứng dụng này, cụ thể là các hiệu ứng được cung cấp, qua đó hiểu thêm về các kĩ thuật liên quan, và với hi vọng rằng có thể làm ra những hiệu ứng đa dạng, phù hợp với nhiều mục đích hơn nữa.

CHƯƠNG 2 : GIỚI THIỆU VỀ PICSART

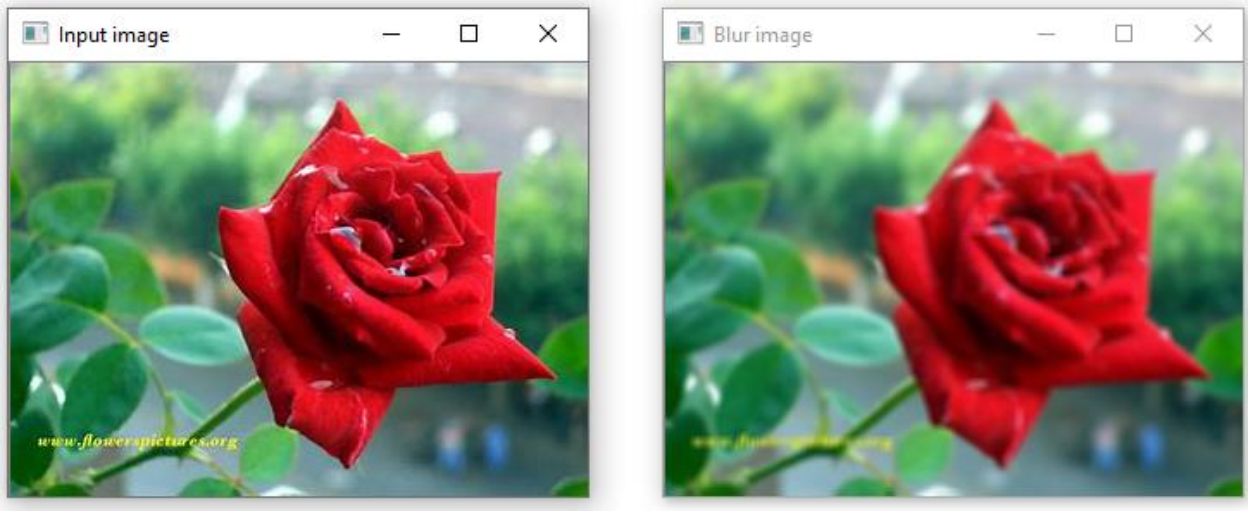
PicsArt là một trình chỉnh sửa tất cả ảnh và video tốt nhất trên thiết bị di động. Bạn được thỏa sức sáng tạo với các hiệu ứng từ cơ bản tới nâng cao, từ tự làm hiệu ứng cho đến các hiệu ứng có sẵn mà không cần các thao tác quá phức tạp.

Hiện nay, Bạn có thể sử dụng miễn phí 7 ngày đầu tiên khi sử dụng PicsArt hoặc bạn có thể mua bản premium để có thể trải nghiệm các hiệu ứng độc đáo, bắt mắt, … với giá 120.000đ/ 1 tháng

Với PicsArt bạn có thể chỉnh sửa các tính năng cơ bản như cắt , xoay, làm mờ,… cho tới các chức năng nâng cao như tạo hiệu ứng mảnh vỡ , ghép ảnh,… Ngoài tự sửa chửa bạn có thể sử dụng các template do người dùng tạo ra trước để áp dụng cho việc chỉnh sửa của mình. Chỉ bản một vài thao tác đơn giản bạn đã có thể có một bức ảnh hợp “trend” hiện tại để có thể đăng lên facebook , instagram,…

CHƯƠNG 3 : CÁC KỸ THUẬT DÙNG TRONG BÀI TOÁN

1. Làm mờ ảnh



- Làm mờ ảnh bằng Gaussian Blur : Trong xử lý ảnh, hiệu ứng làm mờ Gaussian (còn được gọi là làm mịn Gaussian) là kết quả của việc làm mờ hình ảnh bởi một hàm Gauss.
- Đây là một hiệu ứng được sử dụng rộng rãi trong phần mềm đồ họa, điển hình là để giảm nhiễu ảnh và giảm chi tiết. Hiệu ứng hình ảnh của kỹ thuật làm mờ này là một hiệu ứng mờ mịn giống như khi xem hình ảnh qua màn hình mờ, khác hẳn với hiệu ứng bokeh được tạo ra bởi thấu kính mất tiêu cự hoặc bóng của một vật thể dưới ánh sáng thông thường.
- Hàm Gauss trong hai chiều :

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{\frac{x^2+y^2}{2\sigma^2}}$$

+ Đây là tích của ham hàm Gauss trong mỗi chiều x , y
+ Trong đó :
  - X là khoảng cách từ điểm gốc theo trục hoành
  - Y là khoảng cách từ điểm gốc theo trục tung

- σ là độ lệch chuẩn

- Bài toán sử dụng hàm GaussianBlur của thư viện cv2 :

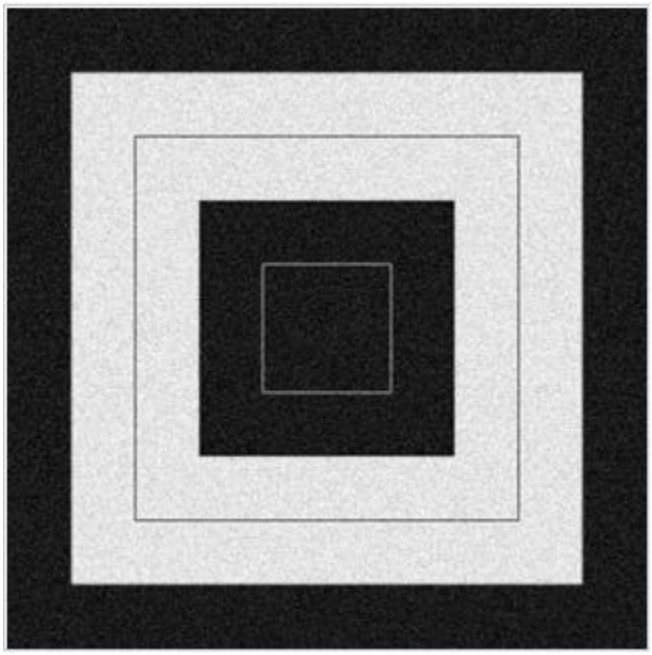
```
blur = cv2.GaussianBlur(img,(5,5),0)
```

+ Trong đó :

- img : là hình ảnh cần làm mờ
- (5 , 5) là chiều cao và chiều rộng của kernel phải là số dương lẻ
- 0 : độ lệch chuẩn theo hướng sigma X

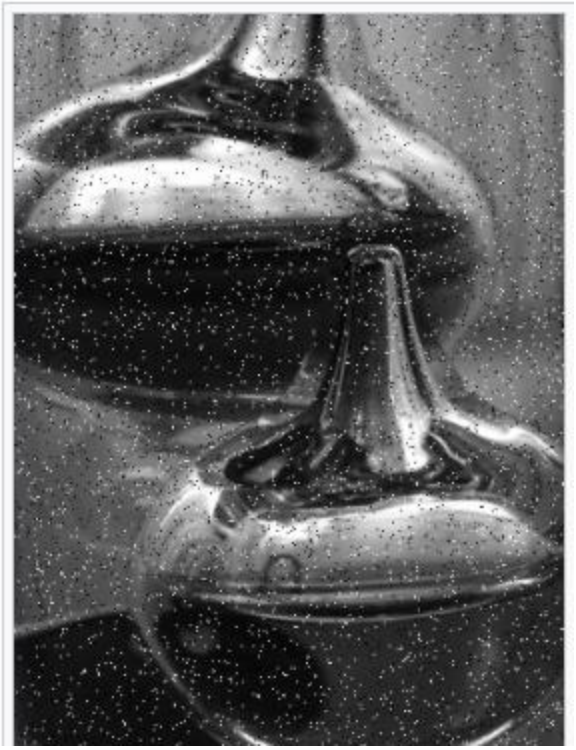
2. Làm nhiễu ảnh

- Gaussian Noise: là nhiễu có các giá trị nhiễu được phân phối theo hàm phân phối Gaussian. Nguồn nhiễu Gauss trong ảnh thường phát sinh trong quá trình thu nhận ảnh.



Gaussian Noise

- Salt and pepper noise : là dạng nhiễu đôi khi xuất hiện trên ảnh. Nó còn được gọi là nhiễu xung động. Nhiễu này có thể do tín hiệu hình ảnh bị nhiễu đột ngột và sắc nét. Nó tự thể hiện dưới dạng các pixel màu trắng và đen thưa thớt.



Salt and pepper noise

- Poisson noise : là một loại nhiễu có thể được mô hình hóa bằng quy trình Poisson. Trong điện tử, nhiễu bắt nguồn từ bản chất rời rạc của điện tích. Nó cũng xảy ra trong việc đếm photon trong các thiết bị quang học, nơi nhiễu Poisson có liên quan đến bản chất hạt của ánh sáng.

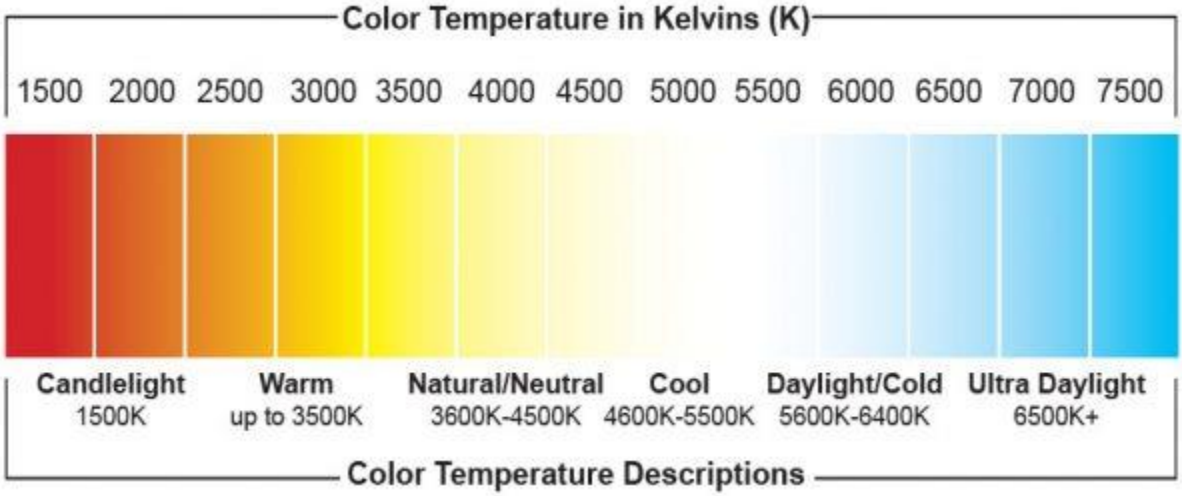
+ Quy trình Poisson : đặt theo tên nhà toán học người Pháp Siméon-Denis Poisson (1781 - 1840), là một quy trình ngẫu nhiên được định nghĩa theo sự xuất hiện của các biến cố.

Poisson noise

- Speckle noise : là một nhiễu dạng hạt vốn đã tồn tại và làm suy giảm chất lượng của radar chủ động, radar khẩu độ tổng hợp (SAR), siêu âm y tế và hình ảnh chụp cắt lớp kết hợp quang học.

Speckle noise

3. Thay đổi nhiệt độ màu ảnh



Lấy giá trị của từng kênh màu trong thang màu , sau đó chia nó cho 255 và lấy giá trị của từng kênh màu trong ảnh gốc nhân với giá trị RGB vừa tính được.

```python
# hàm đổi nhiệt độ cho màu
def convert_temp(image, temp):
    image = image.convert('RGB')
    r, g, b = kelvin_table[temp]
    matrix = ( r / 255.0, 0.0, 0.0, 0.0,
               0.0, g / 255.0, 0.0, 0.0,
               0.0, 0.0, b / 255.0, 0.0)
    print(matrix)
    return image.convert('RGB', matrix)
```

4. Hiệu ứng tạo ảnh vẽ bằng bút chì

Gồm 4 bước :

+ Chuyển ảnh gốc sang ảnh xám bằng hàm cvtColor của thư viện cv2
+ Đảo ngược vùng sáng tối của ảnh xám bằng hàm bitwise_not
+ Làm mờ ảnh thông qua GaussianBlur với kernel = 21x21
+ Lấy giá trị của ảnh xám chia cho 255 – giá trị ảnh làm mờ cuối cùng scale về 256

```python
# hàm effect vẽ bút chì
def effect_sketch(img):
    # chuyển sang dạng ảnh xám
    img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    # đảo ngược màu sắc của hình ảnh
    img_invert = cv2.bitwise_not(img_gray)

    # làm mờ ảnh
    img_smoothing = cv2.GaussianBlur(img_invert, (21, 21),sigmaX=0, sigmaY=0)

    # hình ảnh bút chì
    final_img = dodgeV2(img_gray, img_smoothing)

    return final_img
```

```python
# lấy bản phác thảo
def dodgeV2(x, y):
    return cv2.divide(x, 255 - y,scale= 256)
```

5. Hiệu ứng HDR

- Trong thuật ngữ nhiếp ảnh, dải tần nhạy sáng (dynamic range) là sự chênh lệch giữa phần sáng nhất và tối nhất của ảnh. HDR (high dynamic range) là quá trình làm tăng phạm vi nhạy sáng (tăng dải chênh lệch sáng - tối), được sử dụng để thể hiện cảnh vật chính xác hơn và tạo cảm giác ảnh nét hơn.
- Khi bạn chụp ảnh với chế độ HDR đã được bật, camera sẽ chụp nhiều ảnh liên tiếp với các giá trị phơi sáng khác nhau. Sau đó, phần mềm sẽ kết hợp chúng thành một ảnh duy nhất có thể giữ vững chi tiết từ những vùng tối và sáng nhất, cho ra ảnh có dải chênh lệch sáng – tối rộng nhất.
- Tạo hiệu ứng HDR thông qua 2 bước :

+ Tạo nhiều ảnh với các giá trị phơi sáng khác nhau thông qua hàm Brightness của thư viện Pillow.

- Hàm này sẽ tạo ra một ảnh đen – trắng thông qua tham số “enhance” được truyền vào sau đó kết hợp với ảnh gốc để giảm

hay tăng độ sáng cho ảnh. Tham số "enhance" càng lớn thì ảnh sẽ càng sáng.

+ Kết hợp các ảnh này lại thông qua hàm Mertens.

- Mertens Fushion : một kỹ thuật dùng để kết hợp chuỗi phơi sáng trong khung thành một hình ảnh chất lượng cao mà không cần chuyển đổi sang HDR trước. Chỉ cần quan tâm đến các futures(theo độ tương phản, độ bão hòa và độ phơi sáng) và được lấy một cách tự động. Tuy nhiên, nó không thể được coi là một kỹ thuật HDR thực sự vì không có hình ảnh HDR nào được tạo ra. Hình ảnh trông đẹp hơn trên màn hình, nhưng độ sâu bit thu được của hình ảnh bằng độ sâu đầu vào, không giống như trên ảnh HDR thực, nơi độ sâu bit lớn hơn cho phép lưu trữ các thay đổi cường độ chi tiết hơn. Tính linh hoạt là thế mạnh của nó, phương pháp này có thể được mở rộng để thực hiện xếp chồng tiêu điểm bằng cách sử dụng độ tương phản làm tiêu chí duy nhất.

```python
def hdr(img, rg = 1.8):
    # tạo list ảnh và độ sáng của nó
    times = []
    i = 0.5
    while(i <= rg):
        times.append(i)
        i = i + 0.1

    img_list = []
    for time in times:
        # điều chỉnh độ sáng của hình ảnh
        result = ImageEnhance.Brightness(img).enhance(time)
        img_temp = np.array(result)
        img_list.append(img_temp)
```

```python
def effect_hdr(imgs):
    # hợp nhất các ảnh phơi sáng bằng thuật toán Mertens
    merge_mertens = cv2.createMergeMertens()
    res_mertens = merge_mertens.process(imgs)

    # chuyển về định dạng chuẩn
    res_mertens_8bit = np.clip(res_mertens*255, 0, 255).astype('uint8')
    return res_mertens_8bit
```

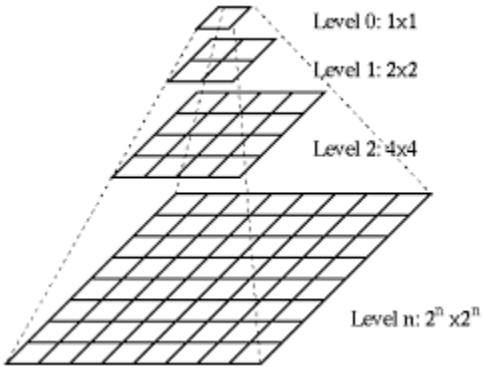
- Kết quả của việc kết hợp ảnh ảnh có độ sáng là 0.5 và 1

6. Image Pyramid

- Image Pyramid (kim tự tháp hình ảnh) : là một tập hợp các hình ảnh phát sinh từ một bức ảnh ban đầu, các hình ảnh này liên tục được lấy mẫu xuống (downsampled) đến khi đạt tới một điểm dừng nhất định.
- Có 2 loại :
    - Gaussian pyramid: được dùng để downsample ảnh (giảm độ phân giải của ảnh).
    - Laplacian pyramid: được dùng để tái tạo lại một hình ảnh được upsample từ một hình ảnh có độ phân giải thấp hơn trong kim tự tháp.

- Gaussian Pyramid:

- Kim tự tháp này là một tập các layer, trong đó hình ảnh ở layer cao hơn thì có kích thước nhỏ hơn



- Các layer được đánh số từ đáy lên đến đỉnh tháp, do đó layer thứ (i + 1) (kí hiệu G_i+1) sẽ nhỏ hơn layer thứ i (kí hiệu G_i).
- Để tạo layer thứ (i + 1) khi downsample, ta có các bước sau:
  - Thực hiện phép tính convolution của Gi với một Gaussian kernel (kernel này có giá trị lớn nhất bên trong và nhỏ dần khi ra ngoài biên, giá trị giảm dần này gọi là độ lệch chuẩn):

$$\frac{1}{16}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

  - Loại bỏ các hàng và cột số chẵn.
- Như vậy ảnh kết quả sẽ có kích thước chính xác bằng ¼ hình ảnh trước nó. Việc lặp lại quá trình này với ảnh đầu vào G0 (ảnh gốc) sẽ cho ra một kim tự tháp hoàn chỉnh.
- Để upsample một tấm hình, ta có các bước sau:
  - Tăng kích thước của hình ảnh lên gấp đôi kích thước ban đầu ở mỗi chiều, các hàng và cột số chẵn mới được gán giá trị 0.
  - Thực hiện convolution với kernel như trên (được nhân lên 4 lần) để xấp xỉ lại giá trị của các pixel rỗng.

7. **Source code :**
   Link Github :
   https://github.com/Hoangthang017/CS231.L11/tree/master/Do_an_cuoi_ki

```
# Làm nhỏ ảnh sử dụng Gaussian Pynamids
img_color = img_rgb
for _ in range(numDownSamples):
    img_color = cv2.pyrDown(img_color)
cv2.imshow("downcolor",img_color)
```

```
# phóng to ảnh trở lại kích thước gốc
for _ in range(numDownSamples):
    img_color = cv2.pyrUp(img_color)
cv2.imshow("upscaling",img_color)
```

- numDownSamples là số lần thực hiện làm nhỏ và phóng to ảnh , với một đơn vị ảnh sẽ thu nhỏ ¼ lần so với ảnh trước đó.

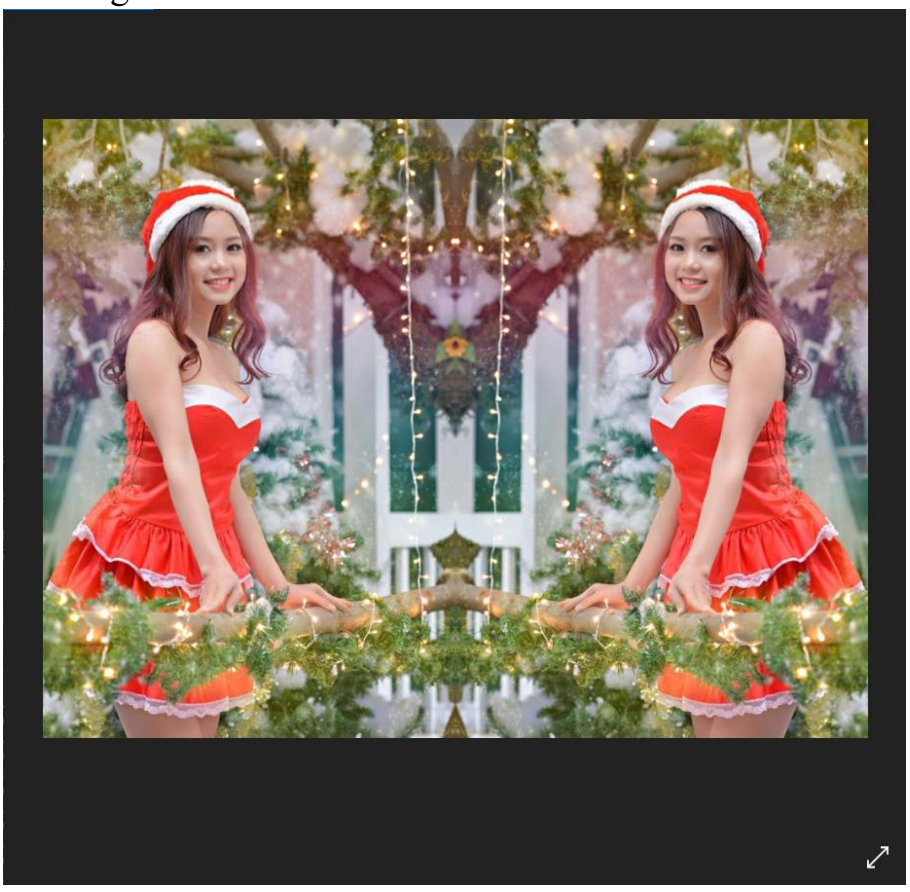
- Lý do dùng image pyrmid thay vì dùng hàm resize thì với pyrmid ảnh nhỏ đi nhưng chất lượng ảnh không hề giảm vẫn giữ được các cạnh trọng ảnh. Để ta có thể lấy các cạnh và làm đậm các cạnh này trong hiệu ứng cartoonizer.

CHƯƠNG 4 : DEMO

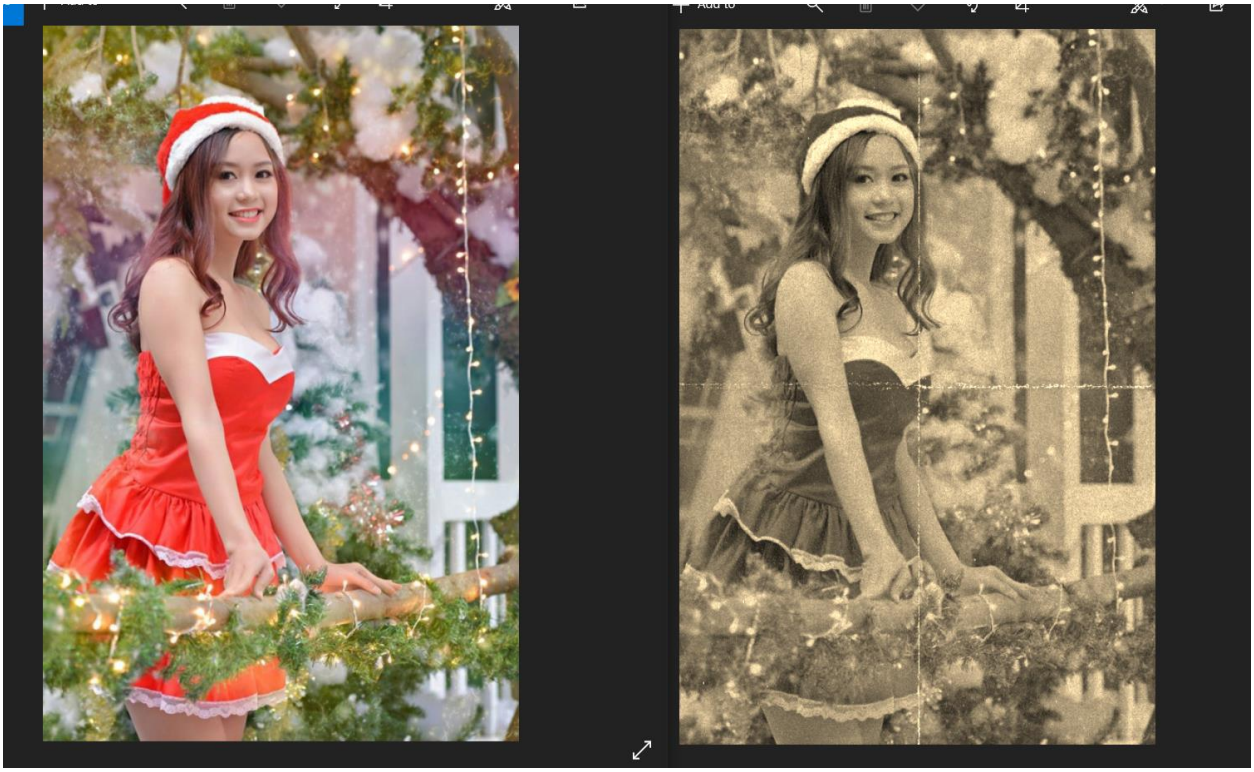
- cartoonizer :

- HDR :

- Gương :



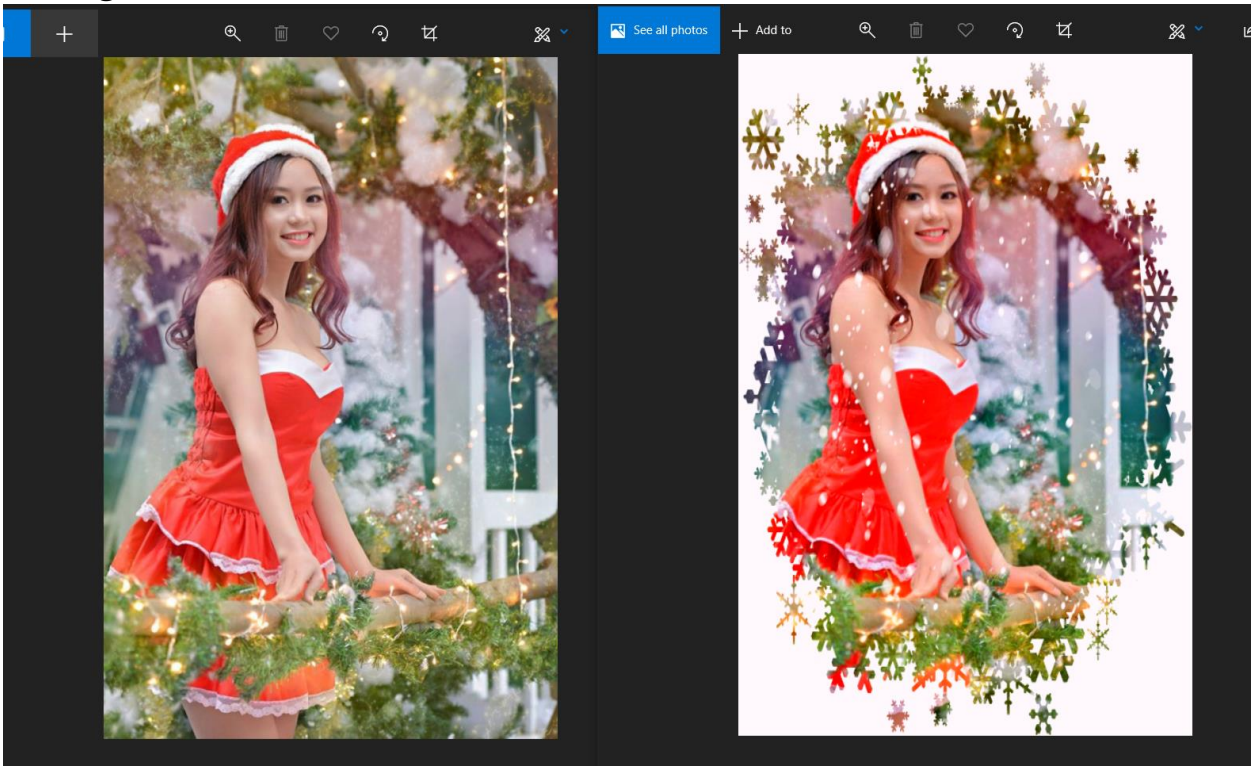
- Ảnh cũ :



- Ảnh vẽ bút chì :

- Mờ ảo :

- Giáng sinh :



CHƯƠNG 5 : TÀI LIỆU THAM KHẢO

[Gaussian Blur] : https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_filtering/py_filtering.html

[Noise]: https://en.wikipedia.org/wiki/Image_noise

[Table Color]: http://www.vendian.org/mncharity/dir3/blackbody/
[Image Pyradmid]: https://docs.opencv.org/master/d4/d1f/tutorial_pyramids.html

[Filter] : https://github.com/Tinker-S/SomeImageFilterWithPython