# Ex 3.1: Color balance

Write a simple application to change the color balance of an image by multiplying each color value by a different user-speciﬁed constant. If you want to get fancy, you can make this application interactive, with sliders.

1. Do you get different results if you take out the gamma transformation before or after doing the multiplication? Why or why not?

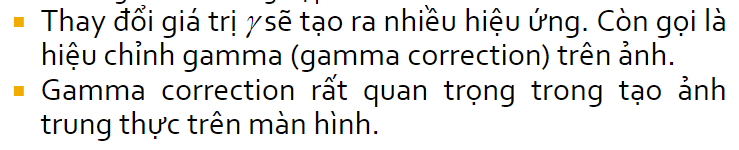
2. Take the same picture with your digital camera using different color balance settings (most cameras control the color balance from one of the menus). Can you recover what the color balance ratios are between the different settings? You may need to put your camera on a tripod and align the images manually or automatically to make this work. Alternatively, use a color checker chart (Figure 10.3b), as discussed in Sections 2.3 and 10.1.1.

3. If you have access to the RAW image for the camera, perform the demosaicing yourself (Section 10.3.1) or downsample the image resolution to get a “true” RGB image. Does your camera perform a simple linear mapping between RAW values and the color- balanced values in a JPEG? Some high-end cameras have a RAW+JPEG mode, which makes this comparison much easier.

4. Can you think of any reason why you might want to perform a color twist (Sec- tion 3.1.2) on the images? See also Exercise 2.9 for some related ideas.

Viết một ứng dụng đơn giản để thay đổi cân bằng màu của hình ảnh bằng cách nhân từng giá trị màu với một hằng số do người dùng chỉ định khác nhau. Nếu bạn muốn trở nên lạ mắt, bạn có thể làm cho ứng dụng này trở nên tương tác, bằng các thanh trượt.

1. Bạn có nhận được các kết quả khác nếu bạn thực hiện phép biến đổi gamma trước hoặc sau khi thực hiện phép nhân không? Tại sao hoặc tại sao không? có



2. Chụp cùng một bức ảnh bằng máy ảnh kỹ thuật số của bạn bằng cách sử dụng các cài đặt cân bằng màu khác nhau (hầu hết các máy ảnh kiểm soát cân bằng màu từ một trong các menu). Bạn có thể khôi phục tỷ lệ cân bằng màu giữa các cài đặt khác nhau không? Bạn có thể cần đặt máy ảnh của mình lên giá ba chân và căn chỉnh hình ảnh theo cách thủ công hoặc tự động để thực hiện công việc này. Ngoài ra, sử dụng biểu đồ kiểm tra màu (Hình 10.3b), như đã thảo luận trong Phần 2.3 và 10.1.1.

3. Nếu bạn có quyền truy cập vào hình ảnh RAW cho máy ảnh, hãy tự thực hiện việc khử dấu hiệu (Mục 10.3.1) hoặc giảm độ phân giải hình ảnh để có được hình ảnh RGB “đúng”. Máy ảnh của bạn có thực hiện ánh xạ tuyến tính đơn giản giữa các giá trị RAW và các giá trị cân bằng màu trong JPEG không? Một số máy ảnh cao cấp có chế độ RAW + JPEG, giúp việc so sánh này dễ dàng hơn nhiều.

4. Bạn có thể nghĩ ra lý do nào khiến bạn muốn thực hiện chỉnh màu (Phần 3.1.2) trên hình ảnh không? Xem thêm Bài tập 2.9 để biết một số ý liên quan.

Vì sự sai khác giữa mắt người và camera về màu sắc(tuyến tính – phi tuyến). Cách thức phi tuyến tính mà con người cảm nhận được ánh sáng và màu sắc

Mục tiêu quan trọng của việc điều chỉnh này là hiển thị các màu cụ thể - đặc biệt là các màu trung tính - một cách chính xác

# Ex 3.2: Compositing and reﬂections Tổng hợp và chỉnh sửa lại

Section 3.1.3 describes the process of compositing an alpha-matted image on top of another. Answer the following questions and optionally validate them experimentally:

1. Most captured images have gamma correction applied to them. Does this invalidate the basic compositing equation (3.8); if so, how should it be ﬁxed?

2. The additive (pure reﬂection) model may have limitations. What happens if the glass is tinted, especially to a non-gray hue? How about if the glass is dirty or smudged? How could you model wavy glass or other kinds of refractive objects?

Phần 3.1.3 mô tả quá trình ghép một hình ảnh có lớp nền alpha lên trên một hình ảnh khác. Trả lời các câu hỏi sau và tùy ý xác nhận chúng bằng thực nghiệm:

1. Hầu hết các hình ảnh đã chụp đều được áp dụng hiệu chỉnh gamma. Điều này có làm mất hiệu lực của phương trình tổng hợp cơ bản (3.8); nếu vậy, nó nên được xử lý như thế nào?

2. Mô hình phụ gia (nguyên chất) có thể có những hạn chế. Điều gì xảy ra nếu kính bị ngả màu, đặc biệt là đối với màu không phải màu xám? Làm thế nào về việc nếu kính bị bẩn hoặc bị nhòe? Làm thế nào bạn có thể mô hình thủy tinh gợn sóng hoặc các loại vật thể khúc xạ khác?

Phương thức PIL.Image.composite () tạo ảnh tổng hợp bằng cách trộn ảnh bằng mặt nạ trong suốt.

Mask image.

mode must be one of the following three types.

1: 1 bit image (binary image)

L: 8-bit grayscale image

RGBA: Image with alpha channel

<https://note.nkmk.me/en/python-pillow-composite/>

# Ex 3.3: Blue screen matting

Set up a blue or green background, e.g., by buying a large piece of colored posterboard. Take a picture of the empty background, and then of the back- ground with a new object in front of it. Pull the matte using the difference between each colored pixel and its assumed corresponding background pixel, using one of the techniques described in Section 3.1.3) or by Smith and Blinn (1996).

Thiết lập nền xanh lam hoặc xanh lá cây, ví dụ: bằng cách mua một tấm áp phích màu lớn. Chụp ảnh nền trống, sau đó là nền sau với một đối tượng mới phía trước. Kéo mờ bằng cách sử dụng sự khác biệt giữa mỗi pixel màu và pixel nền tương ứng giả định của nó, sử dụng một trong các kỹ thuật được mô tả trong Phần 3.1.3) hoặc của Smith và Blinn (1996).

<https://github.com/pymatting/pymatting>

# Ex 3.5: Photo effects

Write a variety of photo enhancement or effects ﬁlters: contrast, so- larization (quantization), etc. Which ones are useful (perform sensible corrections) and which ones are more creative (create unusual images)?

Viết nhiều loại hiệu ứng hoặc cải tiến ảnh: độ tương phản, so-larization (lượng tử hóa), v.v. Cái nào hữu ích (thực hiện các chỉnh sửa hợp lý) và cái nào sáng tạo hơn (tạo ra những hình ảnh khác thường)?

# Ex 3.6: Histogram equalization

Compute the gray level (luminance) histogram for an im- age and equalize it so that the tones look better (and the image is less sensitive to exposure settings). You may want to use the following steps:

1. Convert the color image to luminance (Section 3.1.2).

2. Compute the histogram, the cumulative distribution, and the compensation transfer function (Section 3.1.4).

3. (Optional) Try to increase the “punch” in the image by ensuring that a certain fraction of pixels (say, 5%) are mapped to pure black and white.

4. (Optional) Limit the local gain f′(I) in the transfer function. One way to do this is to limit f (I) < γI or f′(I) < γ while performing the accumulation (3.9), keeping any unaccumulated values “in reserve”. (I’ll let you ﬁgure out the exact details.)

5. Compensate the luminance channel through the lookup table and re-generate the color image using color ratios (2.116).

6. (Optional) Color values that are clipped in the original image, i.e., have one or more saturated color channels, may appear unnatural when remapped to a non-clipped value. Extend your algorithm to handle this case in some useful way.

Nhiệm vụ này thực hiện để làm cho sự khác biệt giữa hai hình ảnh dễ so sánh và sử dụng trong các ấn phẩm in.

Tính toán biểu đồ mức xám (độ chói) để biết độ tuổi và cân bằng nó để tông màu trông đẹp hơn (và hình ảnh ít nhạy cảm hơn với cài đặt phơi sáng). Bạn có thể muốn sử dụng các bước sau:

1. Chuyển đổi hình ảnh màu sang độ chói (Mục 3.1.2).

2. Tính toán biểu đồ, phân phối tích lũy và hàm truyền bù (Phần 3.1.4).

3. (Tùy chọn) Cố gắng tăng “cú đấm” trong hình ảnh bằng cách đảm bảo rằng một phần nhỏ pixel nhất định (ví dụ, 5%) được ánh xạ thành màu đen và trắng thuần túy.

4. (Tùy chọn) Giới hạn độ lợi cục bộ f ′ (I) trong hàm truyền. Một cách để làm điều này là giới hạn f (I) <γI hoặc f ′ (I) <γ trong khi thực hiện tích lũy (3.9), giữ mọi giá trị chưa tích lũy “ở trạng thái dự trữ”. (Tôi sẽ cho bạn biết chi tiết chính xác.)

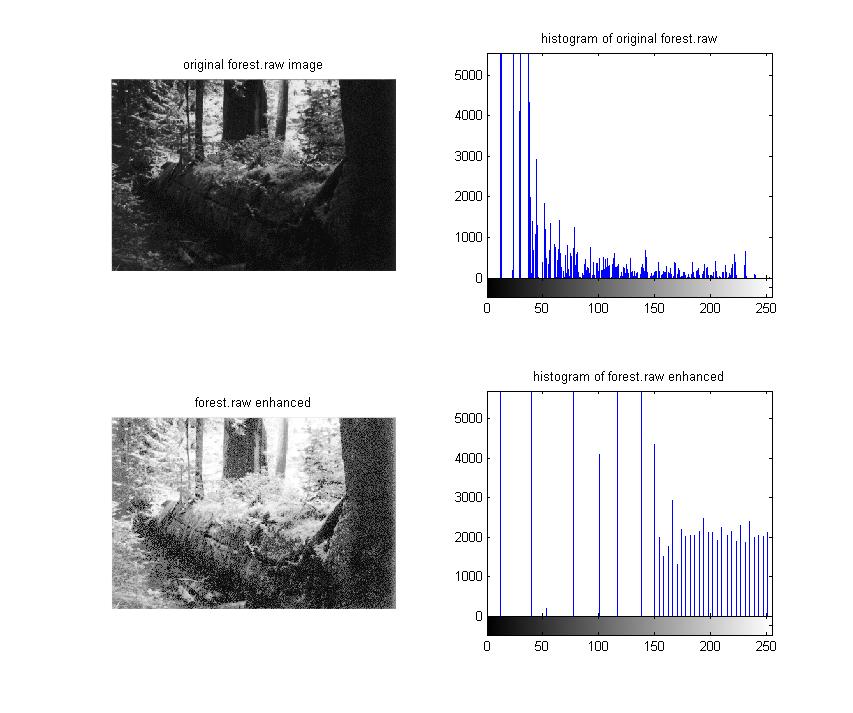
5. Bù kênh độ chói thông qua bảng tra cứu và tạo lại hình ảnh màu bằng cách sử dụng tỷ lệ màu (2.116).

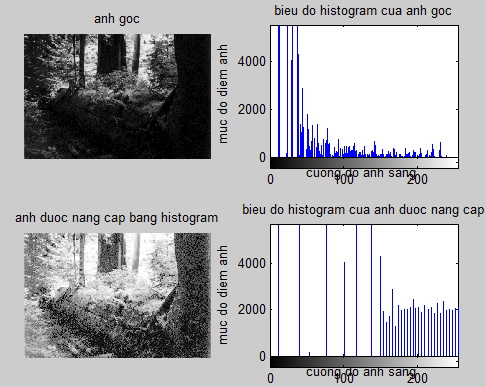
6. (Tùy chọn) Các giá trị màu được cắt bớt trong ảnh gốc, tức là có một hoặc nhiều kênh màu bão hòa, có thể xuất hiện không tự nhiên khi ánh xạ lại thành giá trị không được cắt bớt. Mở rộng thuật toán của bạn để xử lý trường hợp này theo một số cách hữu ích.

Ảnh được nâng cấp nhờ phương pháp histogram sáng màu hơn so với khi không nâng cấp. Do đó quan sát ảnh được dễ dàng hơn.

Có thể nhận thấy trong biểu đồ histogram thì điểm tối nhất được tập trung phía bên trái còn điểm sáng nhất sẽ tập trung phía bên phải. Nên ảnh đầu tiên sẽ có nhiều điểm ảnh phía bên trái còn ảnh sau nâng cấp sẽ tập trung điểm ảnh phía bên phải







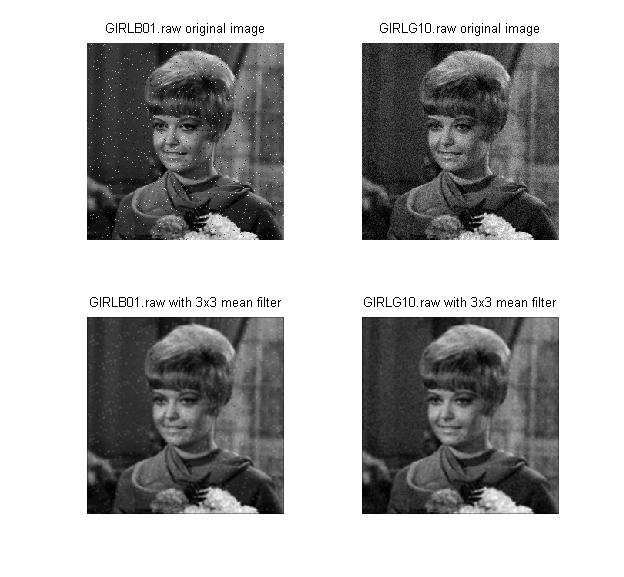
# Ex 3.9: Separable ﬁlters

Implement convolution with a separable kernel. The input should be a grayscale or color image along with the horizontal and vertical kernels. Make sure you support the padding mechanisms developed in the previous exercise. You will need this functionality for some of the later exercises. If you already have access to separable ﬁltering in an image processing package you are using (such as IPL), skip this exercise.

* (Optional) Use Pietro Perona’s (1995) technique to approximate convolution as a sum of a number of separable kernels. Let the user specify the number of kernels and report back some sensible metric of the approximation ﬁdelity.

Thực hiện tích chập với một hạt nhân có thể phân tách. Đầu vào phải là hình ảnh màu xám hoặc màu cùng với các nhân ngang và dọc. Đảm bảo rằng bạn hỗ trợ các cơ chế đệm được phát triển trong bài tập trước. Bạn sẽ cần chức năng này cho một số bài tập sau này. Nếu bạn đã có quyền truy cập để phân tách trong gói xử lý hình ảnh mà bạn đang sử dụng (chẳng hạn như IPL), hãy bỏ qua bài tập này.

• (Tùy chọn) Sử dụng kỹ thuật Pietro Perona’s (1995) để ước tính tích chập dưới dạng tổng của một số hạt nhân có thể phân tách. Cho phép người dùng chỉ định số lượng hạt nhân và báo cáo lại một số số liệu hợp lý của độ trễ gần đúng.



<https://www.cc.gatech.edu/classes/AY2015/cs4475_summer/documents/smoothing_separable.py>

# Ex 3.11: Sharpening, blur, and noise removal

Implement some softening, sharpening, and non-linear diffusion (selective sharpening or noise removal) ﬁlters, such as Gaussian, median, and bilateral (Section 3.3.1), as discussed in Section 3.4.4.

Take blurry or noisy images (shooting in low light is a good way to get both) and try to improve their appearance and legibility.

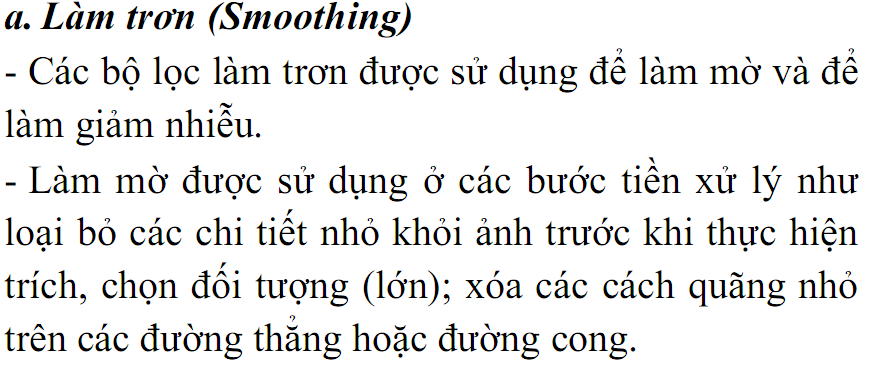
Ví dụ 3.11: Làm sắc nét, làm mờ và loại bỏ nhiễu Thực hiện một số làm mềm, làm sắc nét và khuếch tán phi tuyến tính (làm sắc nét chọn lọc hoặc loại bỏ nhiễu) ﬁ lters, chẳng hạn như Gaussian, trung bình và hai bên (Phần 3.3.1), như được thảo luận trong Phần 3.4 .4.

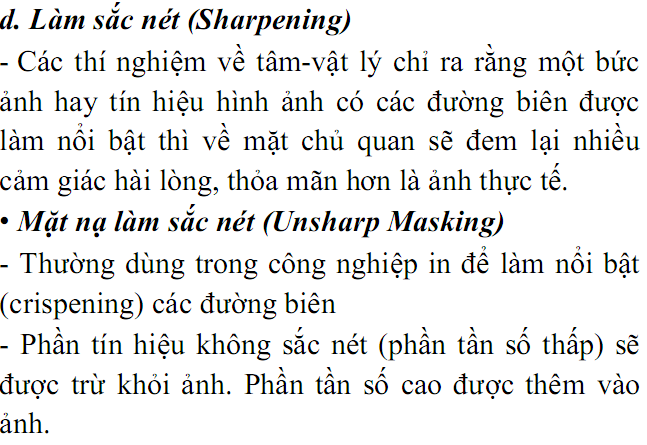
Chụp ảnh mờ hoặc nhiễu (chụp trong điều kiện ánh sáng yếu là một cách tốt để có được cả hai) và cố gắng cải thiện hình thức và độ rõ ràng của chúng.

Bộ lọc .BoxBlur () tương tự như bộ lọc được mô tả trong phần trước giới thiệu về nhân chập. Đối số là bán kính của bộ lọc làm mờ hộp. Trong phần trước thảo luận về hạt nhân, bộ lọc làm mờ hộp mà bạn đã sử dụng là bộ lọc 3x3

<https://realpython.com/image-processing-with-the-python-pillow-library/>

<https://www.geeksforgeeks.org/python-opencv-smoothing-and-blurring/>







# Ex 3.13: Distance transform

Implement some (raster-scan) algorithms for city block and Euclidean distance transforms. Can you do it without peeking at the literature (Danielsson 1980; Borgefors 1986)? If so, what problems did you come across and resolve?

Later on, you can use the distance functions you compute to perform feathering during image stitching (Section 9.3.2

Triển khai một số thuật toán (raster-scan) cho các phép biến đổi khối thành phố và khoảng cách Euclid. Bạn có thể làm điều đó mà không cần nhìn vào tài liệu (Danielsson 1980; Borgefors 1986) không? Nếu vậy, bạn đã gặp và giải quyết những vấn đề gì?

Sau đó, bạn có thể sử dụng các chức năng khoảng cách mà bạn tính toán để thực hiện chỉnh sửa lông trong khi ghép ảnh (Phần 9.3.2)

Distance transform sử dụng ảnh nhị phân, với mỗi pixel được thay thế bằng khoảng cách của nó tới pixel background gần nhất.

<https://docs.opencv.org/3.4/d2/dbd/tutorial_distance_transform.html>

# Ex 3.20: Pyramid blending Kim tự tháp pha trộn

Write a program that takes as input two color images and a binary mask image and produces the Laplacian pyramid blend of the two images.

1. Construct the Laplacian pyramid for each image.

2. Construct the Gaussian pyramid for the two mask images (the input image and its complement).

3. Multiply each Laplacian image by its corresponding mask and sum the images (see Figure 3.43).

4. Reconstruct the ﬁnal image from the blended Laplacian pyramid.

Generalize your algorithm to input n images and a label image with values 1 ...n (the value 0 can be reserved for “no input”). Discuss whether the weighted summation stage (step 3) needs to keep track of the total weight for renormalization, or whether the math just works out. Use your algorithm either to blend two differently exposed image (to avoid under- and over-exposed regions) or to make a creative blend of two different scenes.

Viết một chương trình lấy hai hình ảnh màu đầu vào và một hình ảnh mặt nạ nhị phân và tạo ra hỗn hợp kim tự tháp Laplacian của hai hình ảnh.

1. Dựng kim tự tháp Laplacian cho mỗi hình.

2. Dựng hình chóp Gaussian cho hai hình mặt nạ (hình đầu vào và phần bù của nó).

3. Nhân mỗi hình Laplacian với mặt nạ tương ứng của nó và tính tổng các hình ảnh (xem Hình 3.43).

4. Dựng lại hình từ kim tự tháp Laplacian pha trộn.

Tổng quát hóa thuật toán của bạn để nhập n hình ảnh và một hình ảnh nhãn có giá trị 1 ... n (giá trị 0 có thể được dành cho "không có đầu vào"). Thảo luận xem giai đoạn tổng kết có trọng số (bước 3) có cần theo dõi tổng trọng số để chuẩn hóa lại hay không, hay liệu phép toán chỉ giải ra. Sử dụng thuật toán của bạn để pha trộn hai hình ảnh được phơi sáng khác nhau (để tránh các vùng bị phơi sáng dưới và quá sáng) hoặc để tạo sự kết hợp sáng tạo giữa hai cảnh khác nhau.

<https://pages.cs.wisc.edu/~csverma/CS766_09/ImageMosaic/imagemosaic.html>

Thông thường, chúng tôi đã từng làm việc với một hình ảnh có kích thước không đổi. Nhưng trong một số trường hợp, chúng ta cần làm việc với (giống nhau) các hình ảnh ở độ phân giải khác nhau. Ví dụ: trong khi tìm kiếm thứ gì đó trong một hình ảnh, chẳng hạn như khuôn mặt, chúng tôi không chắc đối tượng sẽ hiện diện trong hình ảnh đã nói ở kích thước nào. Trong trường hợp đó, chúng ta sẽ cần tạo một tập hợp các hình ảnh giống nhau với các độ phân giải khác nhau và tìm kiếm đối tượng trong tất cả chúng. Tập hợp các hình ảnh có độ phân giải khác nhau này được gọi là Hình kim tự tháp (vì khi chúng được giữ trong một ngăn xếp với hình ảnh có độ phân giải cao nhất ở dưới cùng và hình ảnh có độ phân giải thấp nhất ở trên cùng, nó trông giống như một kim tự tháp).

Có hai loại Kim tự tháp hình ảnh. 1) Kim tự tháp Gaussian và 2) Kim tự tháp Laplacian

Để tạo một kim tự tháp, chúng ta cần giảm mẫu cho hình ảnh nguồn cho đến khi đạt được một số điểm dừng mong muốn.

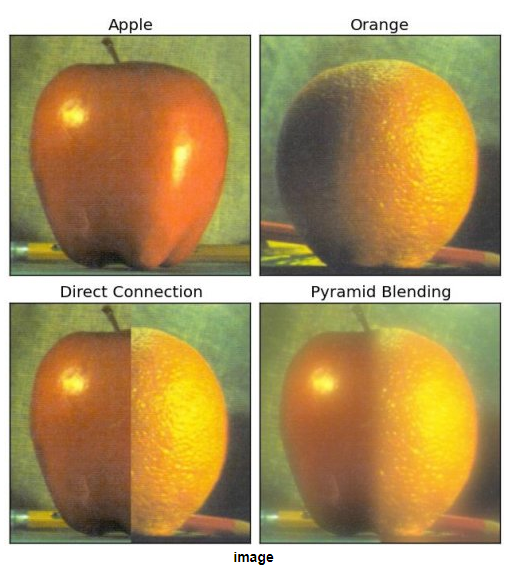
Tải hai hình ảnh táo và cam

Tìm Kim tự tháp Gauss cho táo và cam (trong ví dụ cụ thể này, số cấp là 6)

Từ Kim tự tháp Gauss, tìm Kim tự tháp Laplacian của chúng

Bây giờ nối nửa trái táo và nửa phải cam vào mỗi cấp của Kim tự tháp Laplacian

Cuối cùng từ kim tự tháp hình ảnh chung này, hãy dựng lại hình ảnh ban đầu



# Ex 3.28: Image denoising Làm giảm hình ảnh

Implement at least two of the various image denoising tech- niques described in this chapter and compare them on both synthetically noised image se- quences and real-world (low-light) sequences. Does the performance of the algorithm de- pend on the correct choice of noise level estimate? Can you draw any conclusions as to which techniques work better?

Ví dụ 3.28: Làm giảm hình ảnh

Thực hiện ít nhất hai trong số các kỹ thuật làm giảm hình ảnh khác nhau được mô tả trong chương này và so sánh chúng trên cả chuỗi hình ảnh tổng hợp và chuỗi trong thế giới thực (ánh sáng yếu). Hiệu suất của thuật toán có phụ thuộc vào sự lựa chọn chính xác của ước tính mức ồn không? Bạn có thể rút ra bất kỳ kết luận nào về kỹ thuật nào hoạt động tốt hơn không?

nonLocalMeans Thực hiện thuần túy không cục bộ có nghĩa là làm giảm giá trị mà không có bất kỳ đơn giản hóa nào, và do đó nó không nhanh.

Non-local Means Denoising algorithm Thuật toán Denoising trung bình không cục bộ

Việc sử dụng nâng cao các chức năng này có thể là làm biến dạng hình ảnh có màu theo các không gian màu khác nhau theo cách thủ công

Phương pháp này dựa trên một nguyên tắc đơn giản: thay thế màu của một pixel bằng trung bình cộng các màu của các pixel tương tự. Nhưng các pixel tương tự nhất với một pixel nhất định không có lý do gì để gần nhau cả. Do đó, rất khó để quét một phần lớn hình ảnh để tìm kiếm tất cả các pixel thực sự giống với pixel mà người ta muốn khử.

Khử nhiễu hình ảnh là loại bỏ tiếng ồn từ một hình ảnh bị nhiễu, để khôi phục hình ảnh thực. Tuy nhiên, vì nhiễu, cạnh và kết cấu là các thành phần tần số cao, nên rất khó để phân biệt chúng trong quá trình khử nhiễu và hình ảnh bị khử có thể chắc chắn bị mất một số chi tiết

<https://docs.opencv.org/4.x/d1/d79/group__photo__denoise.html>

<https://www.askpython.com/python/examples/denoising-images-in-python>

<https://www.geeksforgeeks.org/python-denoising-of-colored-images-using-opencv/>

