**TP3** This exercise aims to focus on brightness and contrast manipulation on image and video exploiting OpenCV functions. You will find theoretical explanation and code to complete.

## 1. BRIGHTNESS AND CONTRAST

A general image processing operator is a function that takes one or more input images and produces an output image. Image transforms can be seen as:

- Point operators (pixel transforms)
- Neighborhood (area-based) operators

In Pixel Transforms, each output pixel's value depends on only the corresponding input pixel value (plus, potentially, some globally collected information or parameters). Examples of such operators include brightness and contrast adjustments as well as color correction and transformations.

An image must have the proper brightness and contrast for easy viewing. ***Brightness*** refers to the overall lightness or darkness of the image. ***Contrast*** is the difference in brightness between objects or regions.

Changing brightness of an image is a commonly used point operation. In this operation, the value of each and every pixels in an image should be increased/decreased by a constant. To change the brightness of a video, the same operation should be performed on each frame in the video.

If you want to increase the brightness of an image, you have to add some positive constant value to each and every pixel in the image.

$$new\_image(i,j) = image(i,j) + c \tag{1}$$

If you want to decrease the brightness of an image, you have to subtract some positive constant value from each and every pixel in the image.

$$new\_image(i,j) = image(i,j) - c \tag{2}$$

Let's assume the data type of the image is CV_8U. (i.e. - Pixels in the image is 8 bit unsigned. Please refer OpenCV C++ API for more information.) Hence the valid value range of every pixels in the image should be 0 - 255.

| | | | |
|---|---|---|---|
| 12 | 23 | 84 | 122 |
| 123 | 34 | 92 | 200 |
| 23 | 45 | 29 | 73 |

Original Image

Say, you want to increase the brightness of the original image by 60. Therefore you should add 60 to each pixels in the original image. You must make sure that pixel values in the output image should not exceed the maximum allowable limit by adding 60 to the original image. If it exceeds the maximum limit, you must assign the maximum value instead of the correct value.

Here is the output image of which the brightness is increased by 60. You may have already noticed that the pixel value at (3, 1) location is 255, although it should be 260 after adding 60 to the 200. It is because the maximum allowable pixel value of this image is 255.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 12 + 60 | 23 + 60 | 84 + 60 | 122 + 60 | = | 72 | 83 | 144 | 182 |
| 123 + 60 | 34 + 60 | 92 + 60 | 200 + 60 | | 183 | 94 | 152 | 255 |
| 23 + 60 | 45 + 60 | 29 + 60 | 73 + 60 | | 83 | 105 | 89 | 133 |

Image of which the brightness is increased by 60

Say, you want to decrease the brightness of the original image by 20. Therefore you should subtract 20 from each pixels in the original image. You must make sure that pixel values in the output image should not go below the minimum allowable limit after subtracting 20 from the original image. If it goes below the minimum limit, you must assign the minimum value instead of the correct value.

Here is the output image of which the brightness is decreased by 20. You may have already noticed that the pixel value at (0, 0) location is 0, although it should be -8 after subtracting 20 from 12. It is because the minimum allowable pixel value is 0 for images with unsigned data types.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 12 - 20 | 23 - 20 | 84 - 20 | 122 - 20 | = | 0 | 3 | 64 | 102 |
| 123 - 20 | 34 - 20 | 92 - 20 | 200 - 20 | | 103 | 14 | 72 | 180 |
| 23 - 20 | 45 - 20 | 29 - 20 | 73 - 20 | | 3 | 25 | 9 | 53 |

Image of which brightness is decreased by 20

Changing contrast of an image is also a commonly used point operation. In this operation, the value of each and every pixels in an image should be multiplied by a positive constant which is not equal to one. To change the contrast of a video, the same operation should be performed on each frame in the video.

In order to increase the contrast of an image, each and every pixel in the image should be multiplied by a positive constant larger that one.

$$new\_image(i,j) = image(i,j) * c \qquad c > 1 \tag{3}$$

In order to decrease the contrast of an image, each and every pixel in the image should be multiplied by a positive constant smaller that one.

$$new\_image(i,j) = image(i,j) * c \qquad 0 < c < 1 \tag{4}$$

There are more advance methods to enhance the contrast of an image using histogram (such as histogram equalisation, that you will study in Computer Vision courses).

As before this is your original image and so assume the data type of the image is CV_8U with so a valid value range of every pixels in the image should be 0 - 255.

| | | | |
|---|---|---|---|
| 144 | 245 | 132 | 54 |
| 10 | 62 | 81 | 84 |
| 99 | 106 | 29 | 7 |

Original Image

You want to increase the contrast of the original image by a factor of 2. Therefore you should multiply each pixels in the original image by 2. You must make sure that pixel values in the output image should not exceed the maximum allowable limit after the multiplication. If it exceeds the maximum limit, you must assign the maximum value instead of the correct value.

Here is the image of which the contrast is increased by a factor of 2. You may have already noticed that the pixel value at (0, 0) location is 255, although it should be 288 after multiplying 144 by 2. It is because the maximum allowable pixel value of this image is 255.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 144 * 2 | 245 * 2 | 132 * 2 | 54 * 2 | = | 255 | 255 | 255 | 108 |
| 10 * 2 | 62 * 2 | 81 * 2 | 84 * 2 | | 20 | 124 | 162 | 168 |
| 99 * 2 | 106 * 2 | 29 * 2 | 7 * 2 | | 198 | 212 | 58 | 14 |

Image of which the contrast is increased by factor of 2

Say, you want to halve the contrast of the original image. Therefore you should multiply each pixels in the original image by 0.5. Here is the image of which the contrast is decreased.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 144 * 0.5 | 245 * 0.5 | 132 * 0.5 | 54 * 0.5 | = | 72 | 123 | 66 | 27 |
| 10 * 0.5 | 62 * 0.5 | 81 * 0.5 | 84 * 0.5 | | 5 | 31 | 41 | 42 |
| 99 * 0.5 | 106 * 0.5 | 29 * 0.5 | 7 * 0.5 | | 50 | 53 | 15 | 4 |

Image of which the contrast is decreased

Brightness and contrast adjustments

The described operation of point processes, namely multiplication and addition with a constant, can be formulated as:

$$g(x) = \alpha f(x) + \beta \tag{5}$$

where

- the parameters $\alpha > 0$ and $\beta$ are often called the gain and bias parameters and they are said to control contrast and brightness respectively.
- f(x) can be seen as the source image pixels
- g(x) as the output image pixels.

Then, more conveniently we can write the expression as:

$$g(i, j) = \alpha \cdot f(i, j) + \beta \tag{6}$$

where i and j indicates that the pixel is located in the i-th row and j-th column.

2. **Implementing contrast and brightness**

   *Complete the program bright_ contr_ student.cpp following these steps*:

   - Create parameters to save $\alpha$ and $\beta$ to be entered by the user.
   - Load an image and save it in a Mat object.

- Since you will make some transformations to this image, you need to create another Mat object to store it. Also, we want this to have the following features: initial pixel values equal to zero, same size and type as the original image; to do this you can use Mat::zeros, which returns a zero array of specified size and type (that are the same as the initial image).
- To perform the operation $g(i,j) = \alpha \cdot f(i,j) + \beta$ you have to access to each pixel in image and since you are operating with BGR images, you will have three values per pixel (B, G and R) to access separately. To execute this step, complete adequately the three `for` loops in which the previous operation is calculated as follows:

  $$new\_image.at < Vec3b > (y,x)[c] = saturate\_cast < uchar > (alpha*(image.at < Vec3b > (y,x)[c]) + beta) \quad (7)$$

  In here we are using the syntax $image.at < Vec3b > (y,x)[c]$ to access each pixel in the images (y is the row, x is the column and c is R, G or B). We use $saturate\_cast$ to make sure the values are valid as operation (6) can give values out of range or not integers.
- Finally create windows and show the images, the usual way.

3. **Changing brightness of an image**

   Instead of using the `for` loops to access each pixel, you can use the simple command `void cv::Mat::convertTo (OutputArray m, int rtype, double alpha = 1, double beta = 0 ) const`. The method converts source pixel values to the target data type, applying $saturate\_cast <>$ at the end to avoid possible overflows:

   $$m(x,y) = saturate\_cast < rType > (\alpha(*this)(x,y) + \beta) \quad (8)$$

   in which

   - m is the output matrix; if it does not have a proper size or type before the operation, it is reallocated.
   - rtype in the desired output matrix type or, rather, the depth since the number of channels are the same as the input has; if rtype is negative, the output matrix will have the same type as the input.
   - alpha is the optional scale factor.
   - beta is the optional delta added to the scaled values.

   Using `convertTo` let work faster then with the previous method giving the same results.

   *Complete the program brightness_students.m using convertTo following the instruction in the code: in this case we want just to see effects in changing brightness. Use"eco.jpg" and "van_gogh.jpg".*

4. **Change brightness of a video**

   Remembering that a frame of a video is simply an image, change the brightness of the video "chaplin.mp4" completing the program bright_video_students.cpp.

5. **Changing contrast of an image**

   In this case change the contrast of the images "eco.jpg" and "van_gogh.jpg" completing the program contrast_image_students.cpp

6. **Changing contrast of a video**

   This time change the brightness of the video "chaplin.mp4" completing the program contrast_video_students.cpp. After completing this exercise make some general considerations about brightness and contrast.

7. **Rotate a video**

In the previous lesson you saw how to rotate an image. Exploit those information to rotate the video "chaplin.mp4" completing the program rotate_video_students.cpp. We want you to use a track bar to change the rotating angle dynamically. The function `int createTrackbar(const string& trackbarname, const string& winname, int* value, int count, TrackbarCallback onChange = 0, void userdata = 0)` creates a trackbar and attached that trackbar to a specified window

Here are the functions parameters explanation:

- trackbarname - The name of the trackbar
- winname - The name of the window to which the trackbar is attached
- value - This integer, pointed by this pointer, holds the value associated with the position of the trackbar
- count - The maximum value of the trackbar. The minimum value is always zero.
- onChange - This function will be called everytime the position of the trackbar is changed. The prototype of this function should be "FunctionName(int, void*)". The "int" value is the value associate with the position of the trackbar. And "void*" is any pointer value which you pass as the "userdata" (See the next parameter).
- userdata - This pointer variable will be passed as the second parameter of the above function