

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN
BỘ MÔN THI GIÁC MÁY TÍNH VÀ ĐIỀU KHIỂN
HỌC THÔNG MINH



XỬ LÝ ẢNH SỐ VÀ VIDEO SỐ
BÁO CÁO THỰC HÀNH GIỮA KỲ
PHÉP BIẾN ĐỔI MÀU

Phần 1. Thông tin của nhóm

Mã số sinh viên	Họ và tên	Chức vụ
18120018	Nguyễn Hoàng Đức	Nhóm trưởng
18120061	Lê Nhựt Nam	Thành viên
18120062	Nguyễn Hoàng Nam	Thành viên
18120533	Dương Đoàn Bảo Sơn	Thành viên
18120543	Trần Đại Tài	Thành viên

Phần 2. Phân công

Phân phân công cho thực hành lab 01 – Phép biến đổi màu

Họ tên	Nhiệm vụ	Số điểm
Nguyễn Hoàng Đức	Tính histogram của ảnh màu, ảnh xám, Chuyển ảnh từ hệ màu HSV sang hệ màu RGB	3
Lê Nhựt Nam	Vẽ histogram của ảnh màu, ảnh xám	3
Nguyễn Hoàng Nam	Tăng giảm độ sáng của ảnh, Cân bằng histogram ảnh màu, ảnh xám	3
Dương Đoàn Bảo Sơn	Chuyển ảnh màu sang ảnh xám, Chuyển ảnh xám sang ảnh màu, Chuyển ảnh từ hệ màu RGB sang hệ màu HSV	3
Trần Đại Tài	So sánh ảnh dựa vào histogram, Tăng giảm độ tương phản của ảnh	3

Phần 3. Nội dung thực hiện

A. Nội dung của bài thực hành 01 – Phép biến đổi màu

Câu	Yêu cầu	Tên câu lệnh	Tham số câu lệnh	Điểm	Mức độ hoàn thành
1a	Chuyển ảnh màu sang ảnh xám	--rgb2gray	Không có	1	100%
1b	Chuyển ảnh xám sang ảnh màu	--gray2rgb	Không có	1	100%

2a	Chuyển ảnh từ hệ màu RGB sang hệ màu HSV	--rgb2HSV	Không có	1	100%
2b	Chuyển ảnh từ hệ màu HSV sang hệ màu RGB	--HSV2RGB	Không có	1	100%
3	Tăng giảm độ sáng của ảnh	--bright	b: độ sáng	1	100%
4	Tăng giảm độ tương phản của ảnh	--contrast	c: độ tương phản	1	100%
5	Tính histogram của ảnh màu, ảnh xám	--hist	Không có	2	100%
6	Cân bằng histogram ảnh màu, ảnh xám	--equalhist	Không có	2	100%
7	Vẽ histogram của ảnh màu, ảnh xám	--drawhist	Không có	3	100%
8	So sánh ảnh dựa vào histogram	--compare	Không có	2	100%
Tổng				15	

Trình bày báo cáo: Lê Nhựt Nam

B. Giải quyết vấn đề

1. Chuyển ảnh màu sang ảnh xám

Áp dụng theo công thức

$$gray = \alpha_1 \cdot blue + \alpha_2 \cdot green + \alpha_3 \cdot red$$

$$\begin{cases} \alpha_1 = 0.11 \\ \alpha_2 = 0.59 \\ \alpha_3 = 0.3 \end{cases}$$

2. Chuyển ảnh xám sang ảnh màu

Lần lược gán giá trị ở kênh màu ở ảnh xám cho 3 kênh màu của ảnh màu

3. Chuyển ảnh từ hệ màu RGB sang hệ màu HSV

$$X_{\max} := \max(R, G, B) =: V$$

$$X_{\min} := \min(R, G, B) = V - C$$

$$C := X_{\max} - X_{\min} = 2(V - L)$$

$$L := \min(R, G, B) = \frac{X_{\max} - X_{\min}}{2} V - \frac{C}{2}$$

Có giá trị hue được xác định như sau

$$H := \begin{cases} 0, & \text{if } C = 0 \\ 60^0 \cdot \left(0 + \frac{G-B}{C} \right), & \text{if } V = R \\ 60^0 \cdot \left(2 + \frac{G-B}{C} \right), & \text{if } V = G \\ 60^0 \cdot \left(4 + \frac{G-B}{C} \right), & \text{if } V = B \end{cases}$$

$$S_V := \begin{cases} 0, & \text{if } V = 0 \\ \frac{C}{V}, & \text{if } V \neq 0 \end{cases}$$

Nguồn: https://en.wikipedia.org/wiki/HSL_and_HSV

4. Chuyển ảnh từ hệ màu HSV sang hệ màu RGB

Sử dụng HSV to RGB Alternative

Với hue $H \in [0, 360^0]$, saturation $S = S_V \in [0, 1]$, value $V \in [0, 1]$

Định nghĩa hàm $f(n)$ như sau

$$f(n) = V - VS \max(0, \min(k, 4-k, 1))$$

Trong đó, $k, n \in \mathbb{R}_{\geq 0}$

$$k = \left(0 + \frac{H}{60^0} \right) \bmod 6$$

Trả ra bộ giá trị (R, G, B) nằm trong khoảng $[0, 1]$, như sau

$$(R, G, B) = (f(5), f(3), f(1))$$

Nguồn: https://en.wikipedia.org/wiki/HSL_and_HSV#HSV_to_RGB_alternative

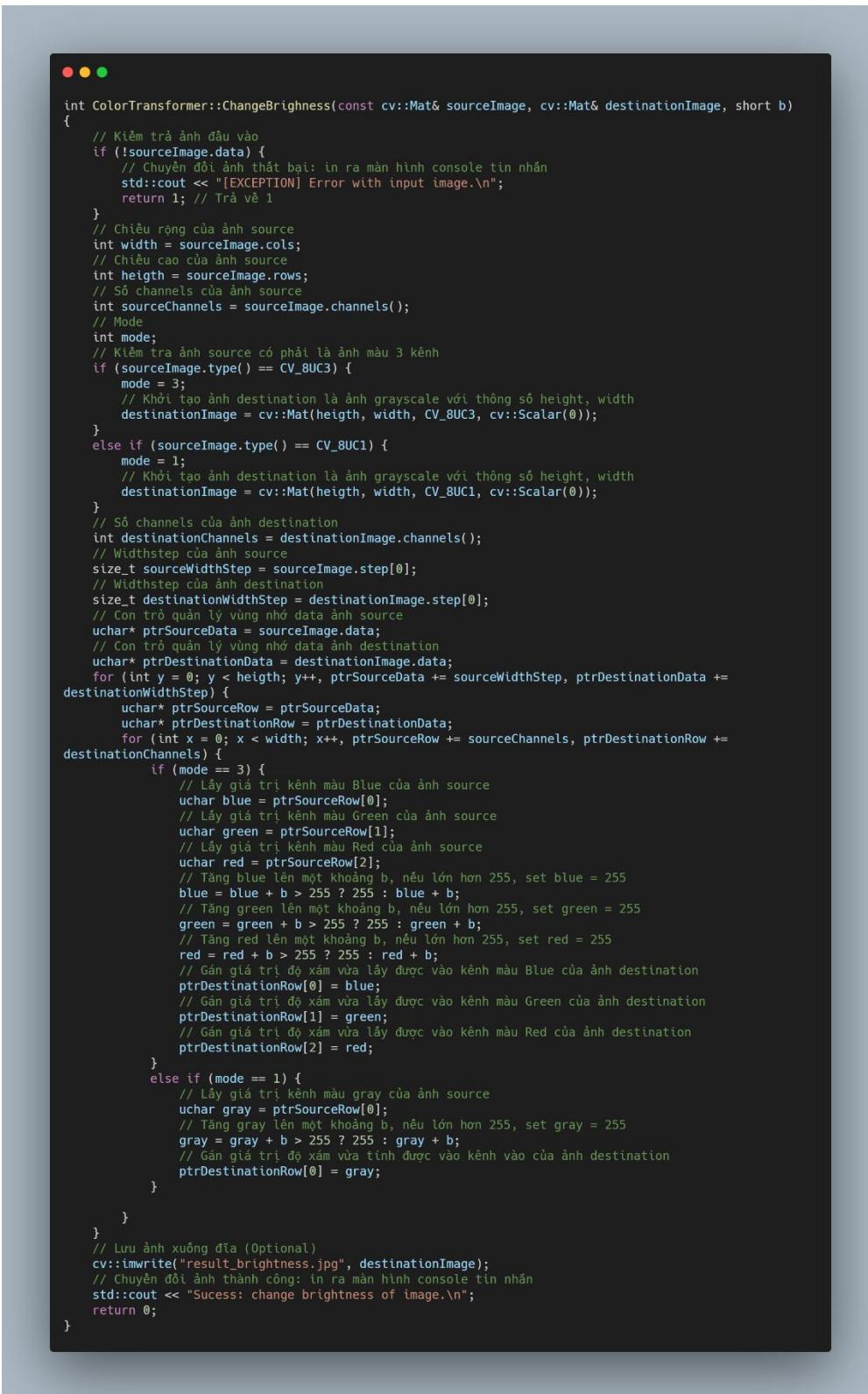
5. Tăng giảm độ sáng của ảnh

Dùng phép biến đổi tuyến tính

$$g(x, y) = f(x, y) + b$$

Với ảnh xám, có một kênh màu, tăng/giảm giá trị b (brightness)

Với ảnh màu, có 3 kênh (blue, green, red), mỗi kênh tăng/giảm giá trị b (brightness)



```
int ColorTransformer::ChangeBrightness(const cv::Mat& sourceImage, cv::Mat& destinationImage, short b)
{
    // Kiểm tra ảnh đầu vào
    if (!sourceImage.data) {
        // Chuyển đổi ảnh thất bại: in ra màn hình console tin nhắn
        std::cout << "[EXCEPTION] Error with input image.\n";
        return 1; // Trả về 1
    }
    // Chiều rộng của ảnh source
    int width = sourceImage.cols;
    // Chiều cao của ảnh source
    int height = sourceImage.rows;
    // Số channels của ảnh source
    int sourceChannels = sourceImage.channels();
    // Mode
    int mode;
    // Kiểm tra ảnh source có phải là ảnh màu 3 kênh
    if (sourceImage.type() == CV_8UC3) {
        mode = 3;
        // Khởi tạo ảnh destination là ảnh grayscale với thông số height, width
        destinationImage = cv::Mat(height, width, CV_8UC3, cv::Scalar(0));
    }
    else if (sourceImage.type() == CV_8UC1) {
        mode = 1;
        // Khởi tạo ảnh destination là ảnh grayscale với thông số height, width
        destinationImage = cv::Mat(height, width, CV_8UC1, cv::Scalar(0));
    }
    // Số channels của ảnh destination
    int destinationChannels = destinationImage.channels();
    // Widthstep của ảnh source
    size_t sourceWidthStep = sourceImage.step[0];
    // Widthstep của ảnh destination
    size_t destinationWidthStep = destinationImage.step[0];
    // Còn trả quản lý vùng nhớ data ảnh source
    uchar* ptrSourceData = sourceImage.data;
    // Còn trả quản lý vùng nhớ data ảnh destination
    uchar* ptrDestinationData = destinationImage.data;
    for (int y = 0; y < height; y++, ptrSourceData += sourceWidthStep, ptrDestinationData += destinationWidthStep) {
        uchar* ptrSourceRow = ptrSourceData;
        uchar* ptrDestinationRow = ptrDestinationData;
        for (int x = 0; x < width; x++, ptrSourceRow += sourceChannels, ptrDestinationRow += destinationChannels) {
            if (mode == 3) {
                // Lấy giá trị kênh màu Blue của ảnh source
                uchar blue = ptrSourceRow[0];
                // Lấy giá trị kênh màu Green của ảnh source
                uchar green = ptrSourceRow[1];
                // Lấy giá trị kênh màu Red của ảnh source
                uchar red = ptrSourceRow[2];
                // Tăng blue lên một khoảng b, nếu lớn hơn 255, set blue = 255
                blue = blue + b > 255 ? 255 : blue + b;
                // Tăng green lên một khoảng b, nếu lớn hơn 255, set green = 255
                green = green + b > 255 ? 255 : green + b;
                // Tăng red lên một khoảng b, nếu lớn hơn 255, set red = 255
                red = red + b > 255 ? 255 : red + b;
                // Gán giá trị độ xám vừa lấy được vào kênh màu Blue của ảnh destination
                ptrDestinationRow[0] = blue;
                // Gán giá trị độ xám vừa lấy được vào kênh màu Green của ảnh destination
                ptrDestinationRow[1] = green;
                // Gán giá trị độ xám vừa lấy được vào kênh màu Red của ảnh destination
                ptrDestinationRow[2] = red;
            }
            else if (mode == 1) {
                // Lấy giá trị kênh màu gray của ảnh source
                uchar gray = ptrSourceRow[0];
                // Tăng gray lên một khoảng b, nếu lớn hơn 255, set gray = 255
                gray = gray + b > 255 ? 255 : gray + b;
                // Gán giá trị độ xám vừa tính được vào kênh vào của ảnh destination
                ptrDestinationRow[0] = gray;
            }
        }
    }
    // Lưu ảnh xuống đĩa (Optional)
    cv::imwrite("result_brightness.jpg", destinationImage);
    // Chuyển đổi ảnh thành công: in ra màn hình console tin nhắn
    std::cout << "Success: change brightness of image.\n";
    return 0;
}
```

6. Tăng giảm độ tương phản của ảnh

Dùng phép biến đổi tuyến tính

$$g(x, y) = c.f(x, y)$$

Với ảnh xám, có một kênh màu, tăng/giảm giá trị c lần (contrast)

Với ảnh màu, có 3 kênh (blue, green, red), mỗi kênh tăng/giảm giá trị c lần (contrast)

7. Tính histogram của ảnh màu, ảnh xám

Trường hợp đầu vào là ảnh xám

Mảng histogram là 1 mảng chứa 256 phần tử



```
// Nếu ảnh đầu vào là ảnh GrayScale
if (sourceImage.channels() == 1) {
    // Khởi tạo ảnh histogram matrix với thông số height = 1, width = 256
    histMatrix = cv::Mat(1, 256, CV_32S, cv::Scalar(0));

    // Số channels của ảnh destination
    int histMatrixChannels = histMatrix.channels();

    // Widthstep của ảnh source
    size_t sourceWidthStep = sourceImage.step[0];

    // Widthstep của ảnh destination
    size_t histMatrixWidthStep = histMatrix.step[0];

    // Con trỏ quản lý vùng nhớ data ảnh source
    uchar* ptrSourceData = sourceImage.data;

    // Con trỏ quản lý vùng nhớ data ảnh destination
    int* ptrHistMatrixData = histMatrix.ptr<int>(0);

    for (int y = 0; y < height; y++, ptrSourceData += sourceWidthStep) {
        uchar* ptrSourceRow = ptrSourceData;
        for (int x = 0; x < width; x++, ptrSourceRow += sourceChannels) {

            uchar gray = ptrSourceRow[0];
            ptrHistMatrixData[gray] += 1;
        }
    }
    std::cout << "Success: Calculated the histogram of image.\n";
    return 1;
}
```

Trường hợp đầu vào là ảnh màu

```
// Nếu ảnh đầu vào là ảnh RGB
else if (sourceImage.channels() == 3) {
    // Khởi tạo ảnh histogram matrix với thông số height = 3, width = 256
    histMatrix = cv::Mat(3, 256, CV_32S, cv::Scalar(0));
    // Số channels của ảnh destination
    int histMatrixChannels = histMatrix.channels();
    // Widthstep của ảnh source
    size_t sourceWidthStep = sourceImage.step[0];
    // Widthstep của ảnh destination
    size_t histMatrixWidthStep = histMatrix.step[0];
    // Con trỏ quản lý vùng nhớ data ảnh source
    uchar* ptrSourceData = sourceImage.data;
    // Con trỏ quản lý vùng nhớ data ảnh destination
    int* blueHistMatrixData = histMatrix.ptr<int>(0);
    int* greenHistMatrixData = histMatrix.ptr<int>(1);
    int* redHistMatrixData = histMatrix.ptr<int>(2);

    for (int y = 0; y < heighth; y++, ptrSourceData += sourceWidthStep) {
        uchar* ptrSourceRow = ptrSourceData;
        for (int x = 0; x < width; x++, ptrSourceRow += sourceChannels) {
            // Lấy giá trị kênh màu Blue của ảnh source
            uchar blue = ptrSourceRow[0];
            // Lấy giá trị kênh màu Green của ảnh source
            uchar green = ptrSourceRow[1];
            // Lấy giá trị kênh màu Red của ảnh source
            uchar red = ptrSourceRow[2];
            // Blue
            blueHistMatrixData[blue] += 1;
            // Green
            greenHistMatrixData[green] += 1;
            // Red
            redHistMatrixData[red] += 1;
        }
    }
    std::cout << "Success: Calculated the histogram of image.\n";
    return 1;
}
```

8. Cân bằng histogram ảnh màu, ảnh xám

```

int ColorTransformer::HistogramEqualization(const cv::Mat& sourceImage, cv::Mat& destinationImage)
{
    // Kiểm tra ảnh đầu vào
    if (!sourceImage.data) {
        // Chuyển đổi ảnh thất bại: in ra màn hình console tin nhắn
        std::cout << "[EXCEPTION] Error with input image.\n";
        return 1; // Trả về 1
    }
    // Chiều rộng của ảnh source
    int width = sourceImage.cols;
    // Chiều cao của ảnh source
    int height = sourceImage.rows;
    // Diện tích area của ảnh
    int area = width * height;
    // Số channels của ảnh source
    int sourceChannels = sourceImage.channels();
    // Mode
    int mode;
    // Mảng chứa blue histogram có 256 phần tử
    int blueHistogram[256];
    // Mảng chứa green histogram có 256 phần tử
    int greenHistogram[256];
    // Mảng chứa red histogram có 256 phần tử
    int redHistogram[256];
    // Mảng chứa gray histogram có 256 phần tử
    int grayHistogram[256];
    // Vòng lặp khởi tạo giá trị cho các mảng trên có giá trị bằng 0
    for (int i = 0; i < 256; i++) {
        blueHistogram[i] = 0;
        greenHistogram[i] = 0;
        redHistogram[i] = 0;
        grayHistogram[i] = 0;
    }
    // Check the source image is RGB type?
    if (sourceImage.type() == CV_8UC3) {
        mode = 3;
        // Khởi tạo ảnh destination là ảnh grayscale với thông số height, width
        destinationImage = cv::Mat(height, width, CV_8UC3, cv::Scalar(0));
    }
    else if (sourceImage.type() == CV_8UC1) {
        mode = 1;
        // Khởi tạo ảnh destination là ảnh grayscale với thông số height, width
        destinationImage = cv::Mat(height, width, CV_8UC1, cv::Scalar(0));
    }
    // Số channels của ảnh destination
    int destinationChannels = destinationImage.channels();
    // Widthstep của ảnh source
    size_t sourceWidthStep = sourceImage.step[0];
    // Widthstep của ảnh destination
    size_t destinationWidthStep = destinationImage.step[0];
    // Con trỏ quản lý vùng nhớ data ảnh source
    uchar* ptrSourceData = sourceImage.data;
    // Con trỏ quản lý vùng nhớ data ảnh destination
    uchar* ptrDestinationData = destinationImage.data;
    // Tính histogram
    for (int y = 0; y < height; y++, ptrSourceData += sourceWidthStep) {
        uchar* ptrSourceRow = ptrSourceData;
        for (int x = 0; x < width; x++, ptrSourceRow += sourceChannels) {
            if (mode == 3) {
                // Lấy giá trị kênh màu Blue của ảnh source
                uchar blue = ptrSourceRow[0];
                blueHistogram[blue]++;
                // Lấy giá trị kênh màu Green của ảnh source
                uchar green = ptrSourceRow[1];
                greenHistogram[green]++;
                // Lấy giá trị kênh màu Red của ảnh source
                uchar red = ptrSourceRow[2];
                redHistogram[red]++;
            }
            else if (mode == 1) {
                // Lấy giá trị kênh màu gray của ảnh source
                uchar gray = ptrSourceRow[0];
                grayHistogram[gray]++;
            }
        }
    }
    // Con trỏ ptrSourceData nám phần data của sourceImage
    ptrSourceData = sourceImage.data;
}

```

Phần cân bằng lược đồ nếu là ảnh xám

```
// Nếu là ảnh grayscale: Cân bằng lược đồ xám
if (mode == 1) {
    int grayEqualization[256];
    // T[0] = H[0]
    grayEqualization[0] = grayHistogram[0];
    // T[p] = T[p-1] + H[p]
    for (int i = 1; i < 256; i++) {
        grayEqualization[i] = grayEqualization[i - 1] + grayHistogram[i];
    }
    // Chuẩn hóa: T về đoạn [0, nG=255]
    // T[p] = round((nG - 1/ NM)T[p])
    for (int i = 0; i < 256; i++) {
        grayEqualization[i] = round(255 * grayEqualization[i] / area);
    }
    // Tạo ảnh kết quả
    for (int y = 0; y < height; y++, ptrSourceData += sourceWidthStep, ptrDestinationData += destinationWidthStep) {
        uchar* ptrSourceRow = ptrSourceData;
        uchar* ptrDestinationRow = ptrDestinationData;

        for (int x = 0; x < width; x++, ptrSourceRow += sourceChannels, ptrDestinationRow += destinationChannels) {
            // Lấy giá trị kênh màu Gray của ảnh source
            uchar gray = ptrSourceRow[0];
            // Gán giá trị độ xám vừa tính được vào kênh vào của ảnh destination
            ptrDestinationRow[0] = grayEqualization[gray];
        }
    }
}
```

Phần cân bằng lược đồ nếu là ảnh màu

```

// Nếu là ảnh màu 3 kênh: Cân bằng lược đồ màu
else if (mode == 3) {
    int blueEqualization[256];
    int greenEqualization[256];
    int redEqualization[256];
    // T[0] = H[0]
    blueEqualization[0] = blueHistogram[0];
    greenEqualization[0] = greenHistogram[0];
    redEqualization[0] = redHistogram[0];
    // T[p] = T[p-1] + H[p]
    for (int i = 1; i < 256; i++) {
        blueEqualization[i] = blueEqualization[i - 1] + blueHistogram[i];
        greenEqualization[i] = greenEqualization[i - 1] + greenHistogram[i];
        redEqualization[i] = redEqualization[i - 1] + redHistogram[i];
    }
    // Chuẩn hóa: T về đoạn [0, nG=255]
    // T[p] = round((nG - 1/ NM)T[p])
    for (int i = 0; i < 256; i++) {
        blueEqualization[i] = round(255 * blueEqualization[i] / area);
        greenEqualization[i] = round(255 * greenEqualization[i] / area);
        redEqualization[i] = round(255 * redEqualization[i] / area);
    }
    // Tạo ảnh kết quả
    for (int y = 0; y < height; y++, ptrSourceData += sourceWidthStep, ptrDestinationData += destinationWidthStep) {
        uchar* ptrSourceRow = ptrSourceData;
        uchar* ptrDestinationRow = ptrDestinationData;

        for (int x = 0; x < width; x++, ptrSourceRow += sourceChannels, ptrDestinationRow += destinationChannels) {
            // Lấy giá trị kênh màu Blue của ảnh source
            uchar blue = ptrSourceRow[0];
            // Lấy giá trị kênh màu Green của ảnh source
            uchar green = ptrSourceRow[1];
            // Lấy giá trị kênh màu Red của ảnh source
            uchar red = ptrSourceRow[2];
            // Gán giá trị độ xám vừa lấy được vào kênh màu Blue của ảnh destination
            ptrDestinationRow[0] = blueEqualization[blue];
            // Gán giá trị độ xám vừa lấy được vào kênh màu Green của ảnh destination
            ptrDestinationRow[1] = greenEqualization[green];
            // Gán giá trị độ xám vừa lấy được vào kênh màu Red của ảnh destination
            ptrDestinationRow[2] = redEqualization[red];
        }
    }
}
// Lưu ảnh xuống đĩa (Optional)
cv::imwrite("result_histogram_equalization.jpg", destinationImage);
// Chuyển đổi ảnh thành công: in ra màn hình console tin nhắn
std::cout << "Sucess: equalize histograms of image.\n";
return 0;
}

```

9. Vẽ histogram của ảnh màu, ảnh xám

Nếu ảnh đầu vào là ảnh grayscale

Vẽ một biểu đồ histogram có chiều cao 300 pixel, chiều rộng 256 pixel - ứng với mảng histogram tính được

```
// Vẽ histogram cho ảnh grayscale
if (hist.rows == 1)
{
    // Con trỏ quản lý vùng data của ma trận histogram
    uchar* ptrHistMatrixData = hist.data;
    // Khai báo giá trị max, gán giá trị này bằng ptrHistMatrixData[0]
    uchar max = *ptrHistMatrixData;
    // Tìm giá trị max trong mảng histogram
    for (int i = 0; i < hist.cols; i++)
    {
        max = max < *(ptrHistMatrixData + i) ? *(ptrHistMatrixData + i) : max;
    }
    // Chuẩn hóa các giá trị trong mảng histogram về đoạn [0, 300]
    for (int i = 0; i < hist.cols; i++)
    {
        *(ptrHistMatrixData + i) = *(ptrHistMatrixData + i) * 300 / max;
    }
    // Khởi tạo histogram image, height = 300, width = 256, màu đen
    histImage = cv::Mat(300, 256, CV_8UC1, cv::Scalar(0));
    // Với mỗi giá trị trong mảng histogram sẽ là một đường thẳng từ điểm giá trị của giá trị màu
    // đó đến chỉ số của giá trị màu đó, độ thick = 2
    for (int j = 0; j < histImage.cols; j++)
    {
        line(histImage, cv::Point(j, histImage.rows - 1 - *(ptrHistMatrixData + j)),
              cv::Point(j, histImage.rows - 1), Color::white, 2);
    }
    return 1;
}
```

Nếu ảnh đầu vào là ảnh màu

```

// Vẽ histogram cho ảnh màu
else if (hist.rows == 3)
{
    // Con trỏ quản lý vùng data của ma trận histogram
    uchar* ptrHistMatrixData = hist.data;
    // Khai báo giá trị max, gán giá trị này bằng ptrHistMatrixData[0]
    uchar max = *ptrHistMatrixData;
    uchar* ptrSourceData = ptrHistMatrixData;
    // Tìm giá trị max trong mảng histogram
    for (int i = 0; i < hist.rows; i++, ptrSourceData += hist.step[0])
    {
        uchar* ptrSourceRow = ptrSourceData;
        for (int j = 0; j < hist.cols; j++)
        {
            max = max < *(ptrSourceRow + j) ? *(ptrSourceRow + j) : max;
        }
    }
    // Chuẩn hóa các giá trị trong mảng histogram về đoạn [0, 300]
    for (int i = 0; i < hist.rows; i++, ptrSourceData += hist.step[0])
    {
        uchar* ptrSourceRow = ptrSourceData;
        for (int j = 0; j < hist.cols; j++)
        {
            *(ptrHistMatrixData + i) = *(ptrHistMatrixData + i) * 300 / max;
        }
    }
    // Khởi tạo histogram image, height = 300, width = 256 * 3, màu đen
    histImage = cv::Mat(300, 256 * 3, CV_8UC3, cv::Scalar(0));
    size_t histMatrixWidthStep = hist.step[0];
    // Con trỏ quản lý mảng blue
    uchar* blueHistMatrixData = ptrHistMatrixData;
    // Con trỏ quản lý mảng green
    uchar* greenHistMatrixData = ptrHistMatrixData + histMatrixWidthStep;
    // Con trỏ quản lý mảng red
    uchar* redHistMatrixData = ptrHistMatrixData + histMatrixWidthStep + histMatrixWidthStep;
    // Với mỗi giá trị trong mảng histogram sẽ là một đường thẳng từ điểm giá trị của giá trị màu
    // đó đến chỉ số của giá trị màu đó, độ thick = 2
    for (int j = 0; j < hist.cols; j++)
    {
        // vẽ đường màu xanh dương
        line(histImage, cv::Point(j, histImage.rows - 1 - *(blueHistMatrixData + j)),
              cv::Point(j, histImage.rows - 1), Color::blue, 2);
        // vẽ đường màu xanh lá cây
        line(histImage, cv::Point(256 + j, histImage.rows - 1 - *(greenHistMatrixData + j)),
              cv::Point(256 + j, histImage.rows - 1), Color::green, 2);
        // vẽ đường màu đỏ
        line(histImage, cv::Point(512 + j, histImage.rows - 1 - *(redHistMatrixData + j)),
              cv::Point(512 + j, histImage.rows - 1), Color::red, 2);
    }
    return 1;
}

```

10. So sánh ảnh dựa vào histogram

So sánh 2 ảnh grayscale

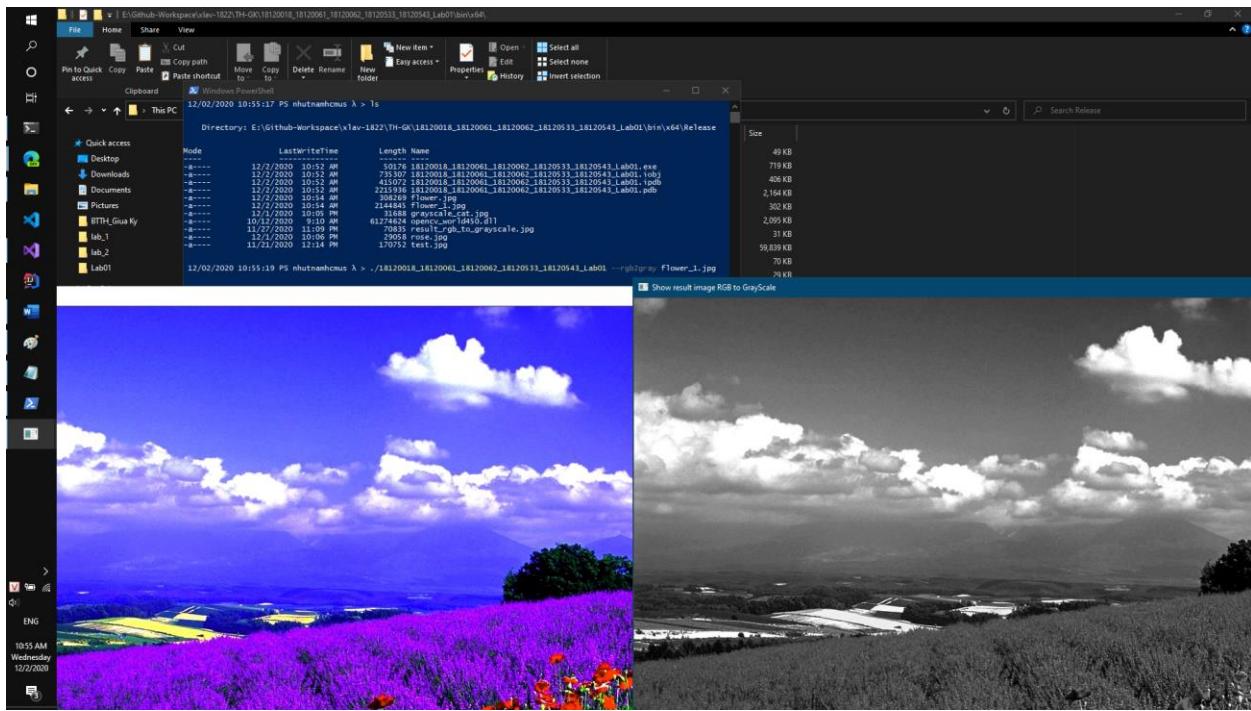
So sánh 2 ảnh màu

Dựa trên khoảng cách Euclidean giữa 2 điểm ảnh của 2 histogram

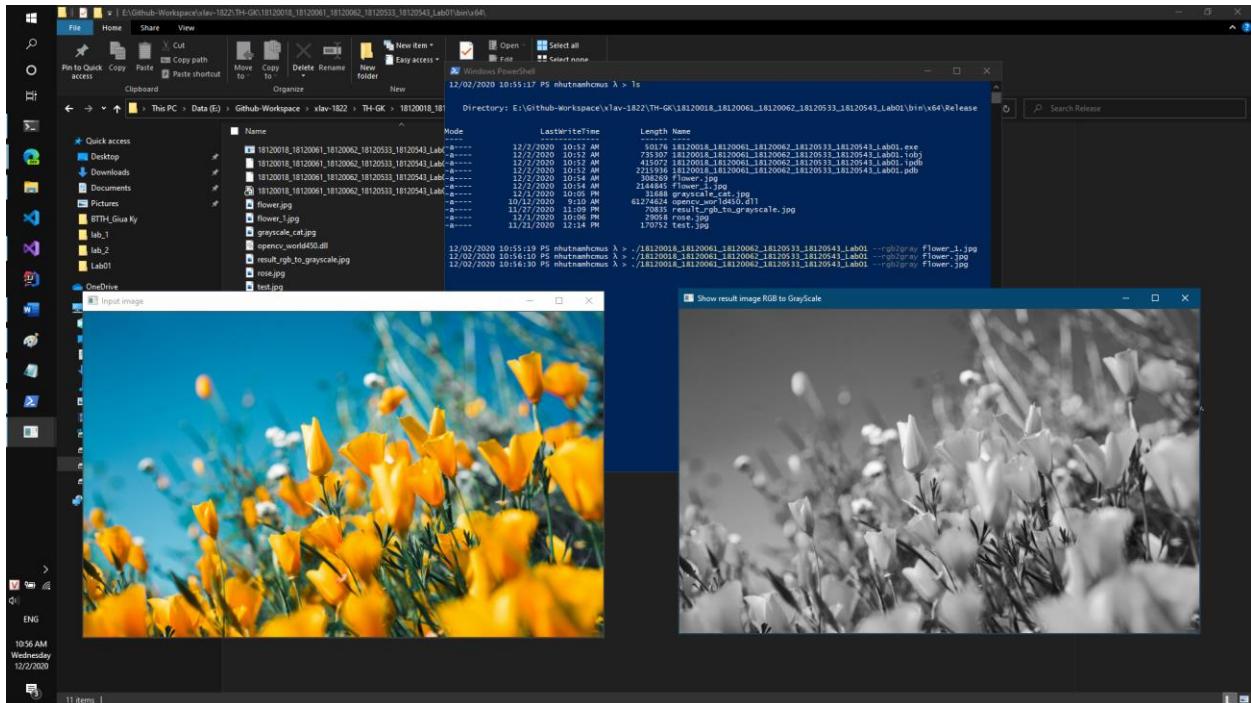
C. Demo/ Kiểm thử các yêu cầu

1. Câu 1a: Chuyển ảnh màu sang ảnh xám

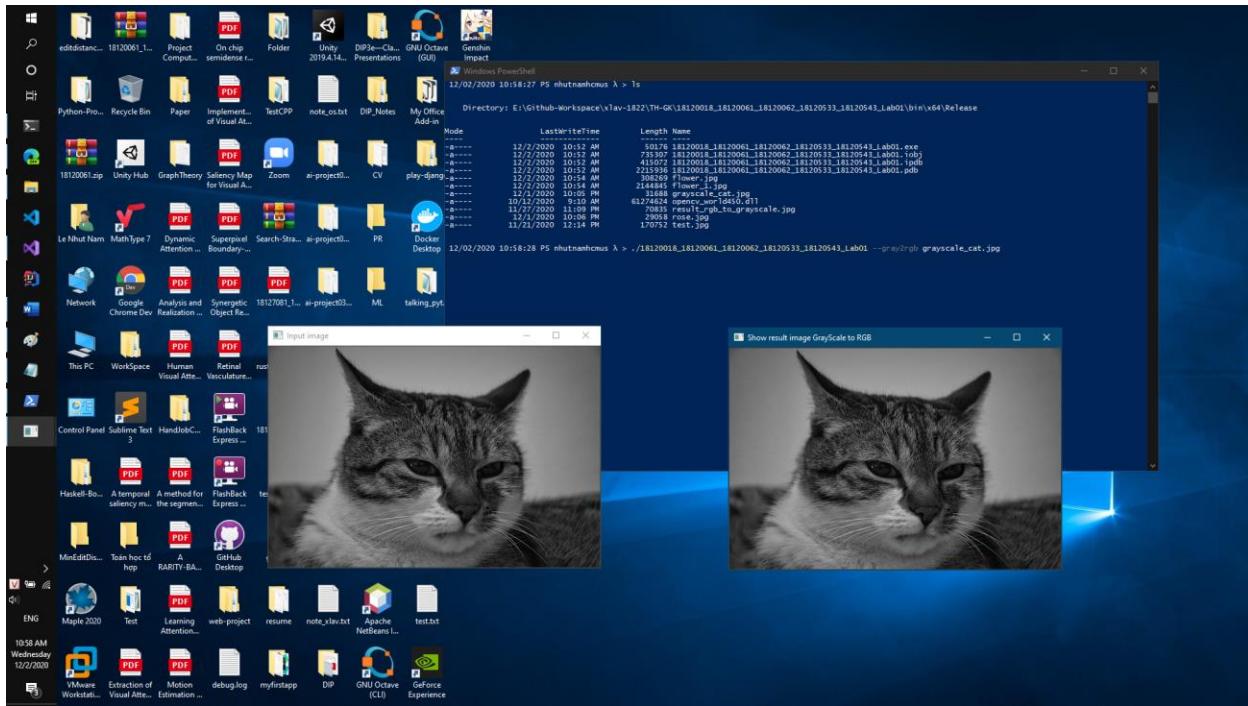
./18120018_18120061_18120062_18120533_18120543_Lab01 --rgb2gray flower_1.jpg



./18120018_18120061_18120062_18120533_18120543_Lab01 --rgb2gray flower_1.jpg



2. Câu 1b: Chuyển ảnh xám sang ảnh màu



3. Câu 2a: Chuyển ảnh từ hệ màu RGB sang hệ màu HSV
4. Câu 2b: Chuyển ảnh từ hệ màu HSV sang hệ màu RGB

Bước kiểm thử chung cho câu 2a, 2b: Input là một ảnh RGB, sau đó gọi hàm converter từ RGB sang HSV, rồi lấy ảnh output đó chuyển ngược về HSV

```

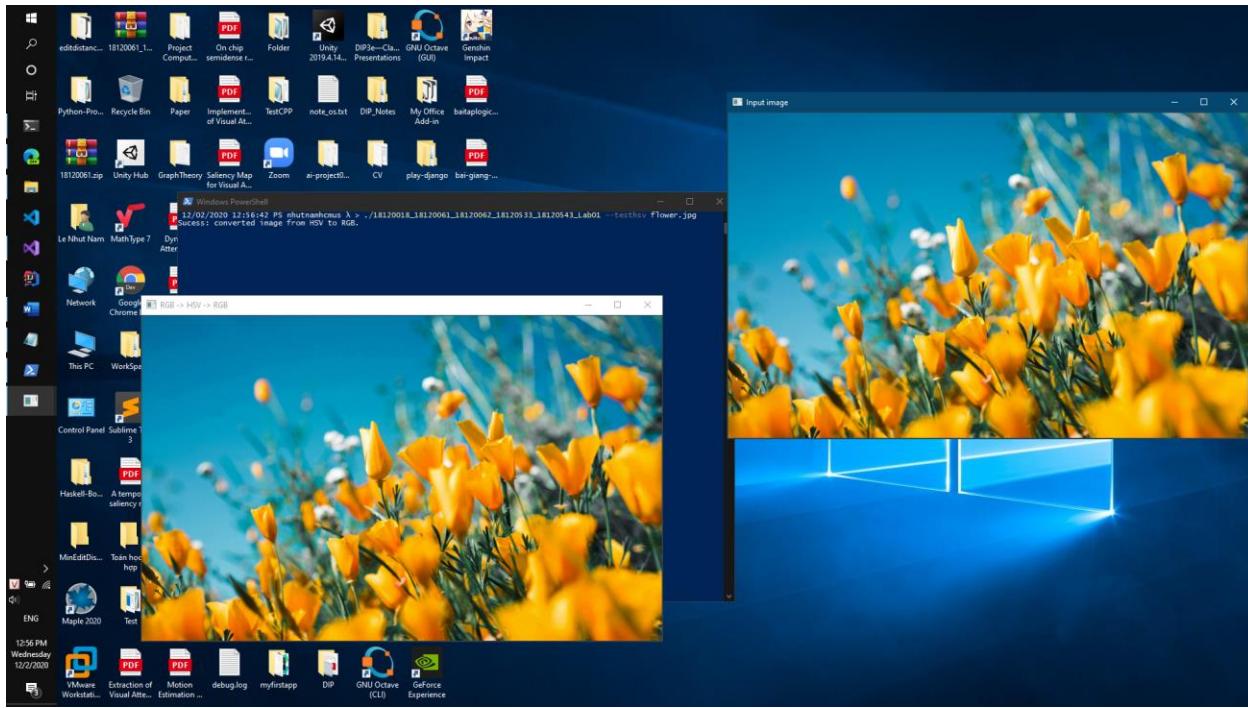
else if (__str_cmp__(argv[1], "--testHSV")) {
    // Đọc ảnh (image) đầu vào
    cv::Mat input_image = cv::imread(argv[2], cv::IMREAD_ANYCOLOR);

    cv::Mat temp;
    // Tính histogram của ảnh
    cv::Mat output_image;
    converter.Convert(input_image, temp, 2);
    converter.Convert(temp, output_image, 3);
    // Dispaly ảnh ra màn hình
    cv::namedWindow("Input image", cv::WINDOW_AUTOSIZE);
    cv::imshow("Input image", input_image);

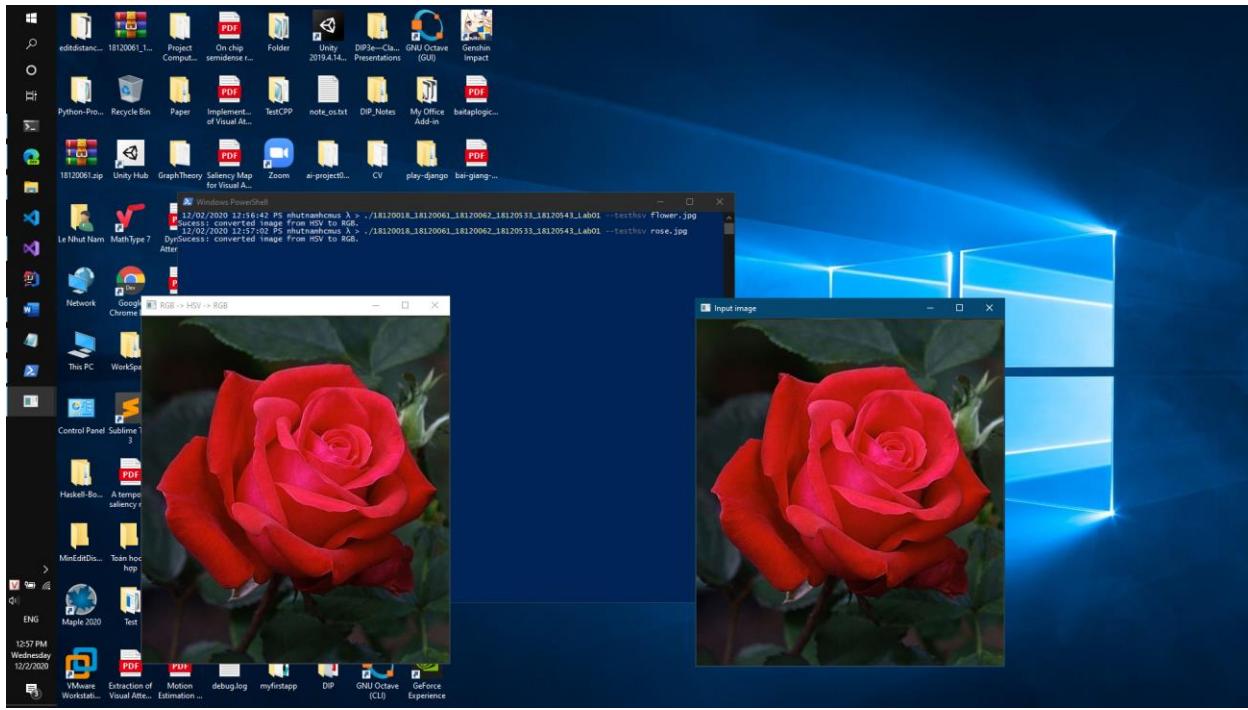
    cv::namedWindow("RGB -> HSV -> RGB", cv::WINDOW_AUTOSIZE);
    cv::imshow("RGB -> HSV -> RGB", output_image);
    cv::waitKey(0);
}

```

./18120018_18120061_18120062_18120533_18120543_Lab01 --testHSV flower.jpg



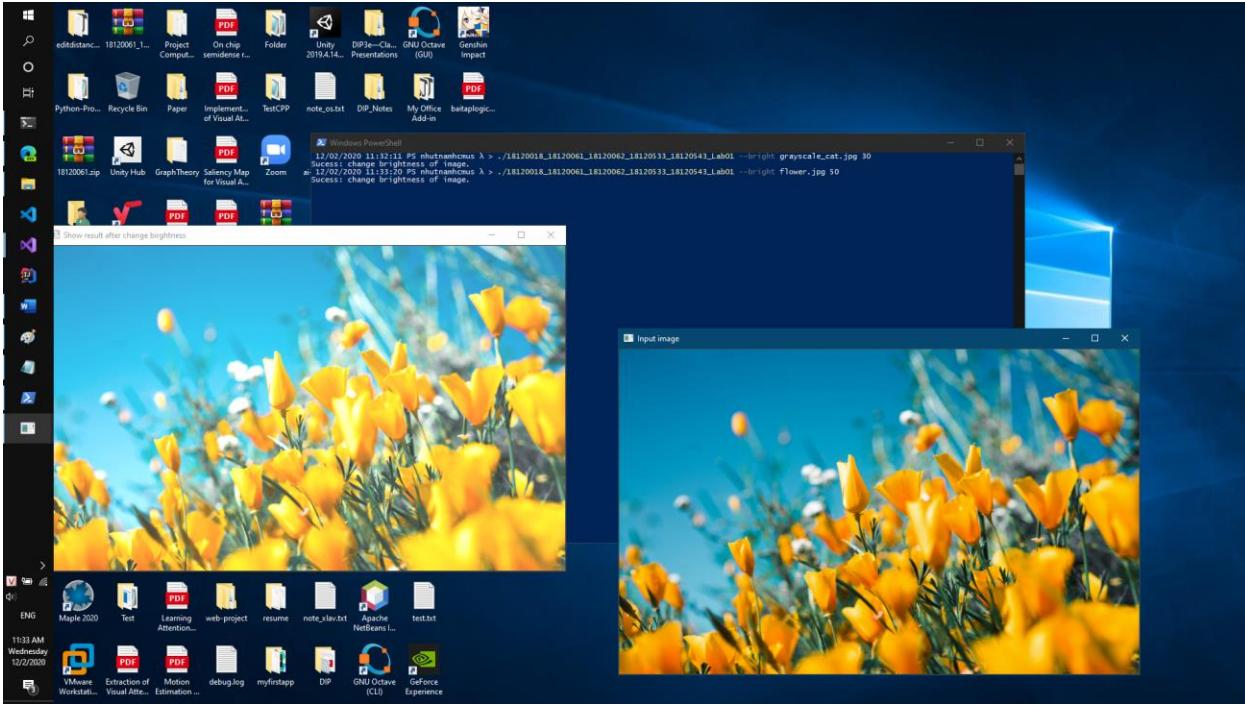
./18120018_18120061_18120062_18120533_18120543_Lab01 --testhsv rose.jpg



5. Câu 3: Tăng giảm độ sáng của ảnh

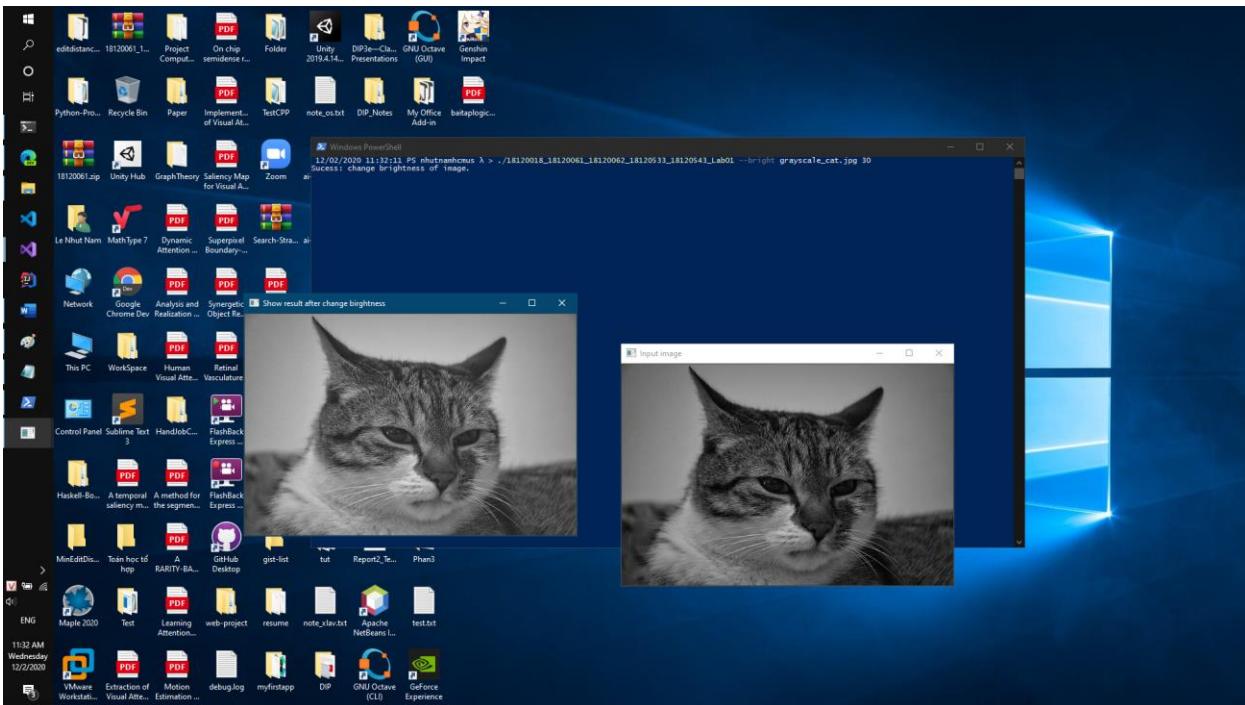
Ảnh màu

./18120018_18120061_18120062_18120533_18120543_Lab01 --bright flower.jpg 50



Ảnh xám

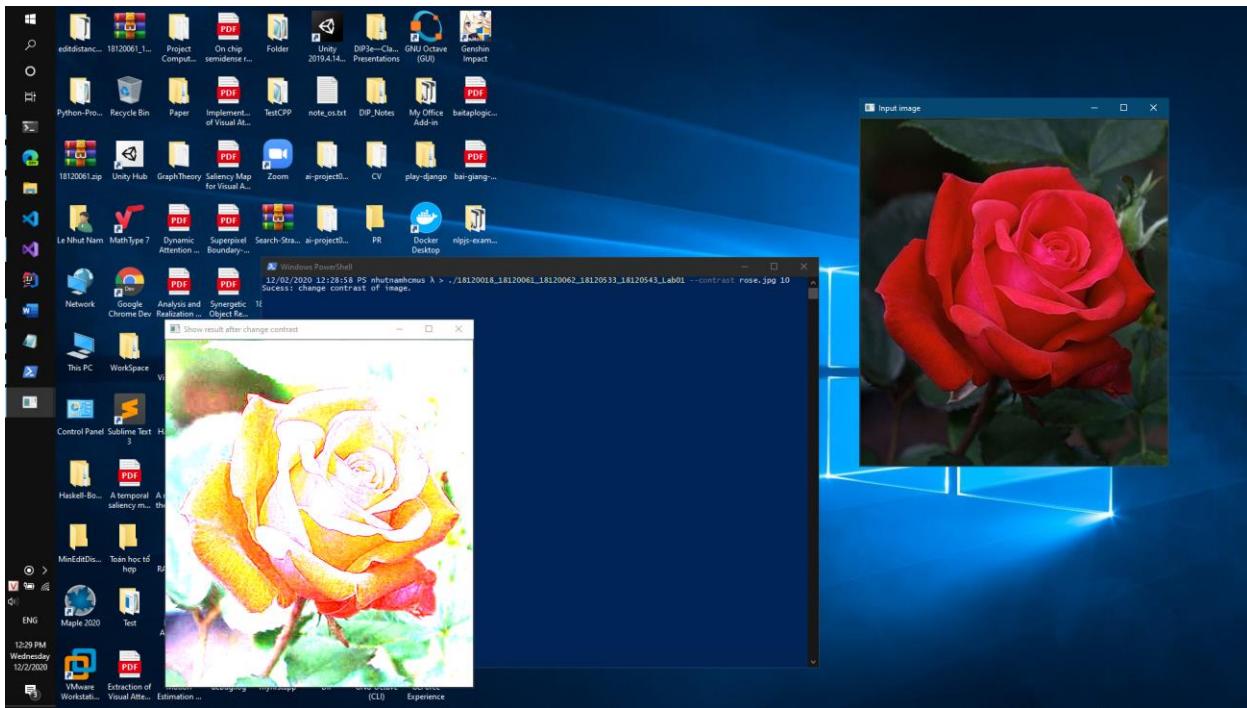
`./18120018_18120061_18120062_18120533_18120543_Lab01 --bright
grayscale_cat.jpg 30`



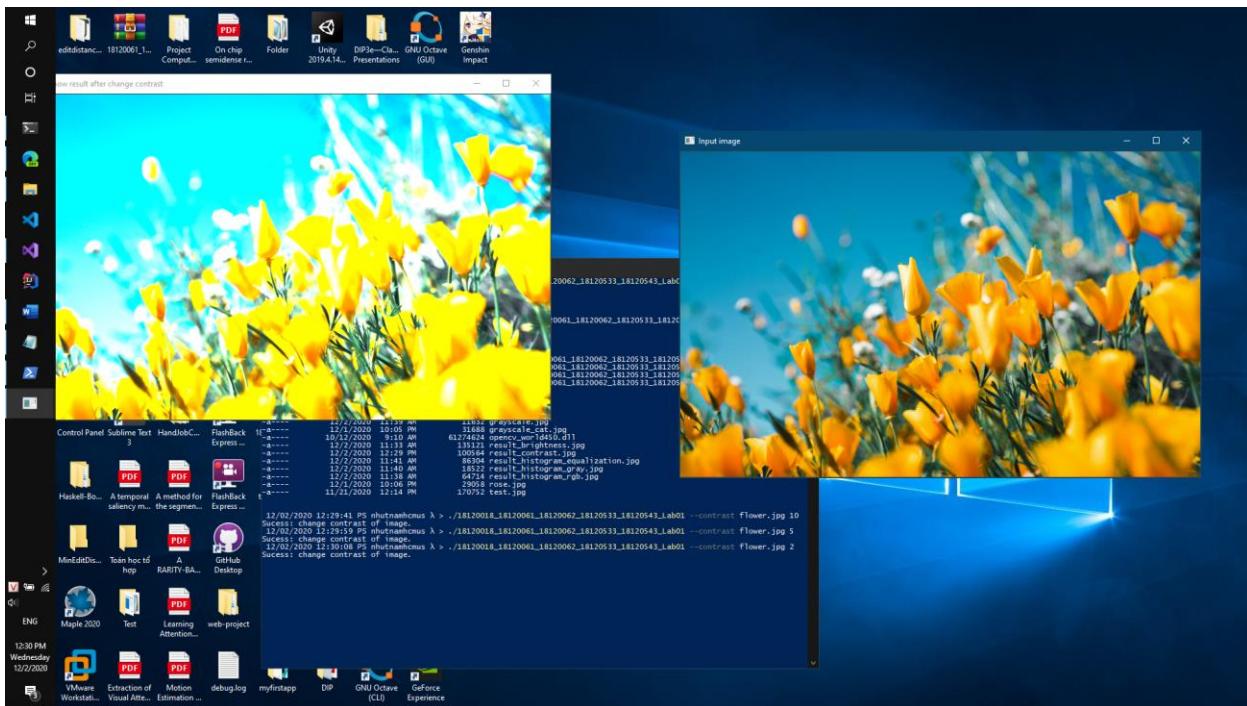
6. Câu 4: Tăng giảm độ tương phản của ảnh

Ảnh màu

./18120018_18120061_18120062_18120533_18120543_Lab01 --contrast rose.jpg 10

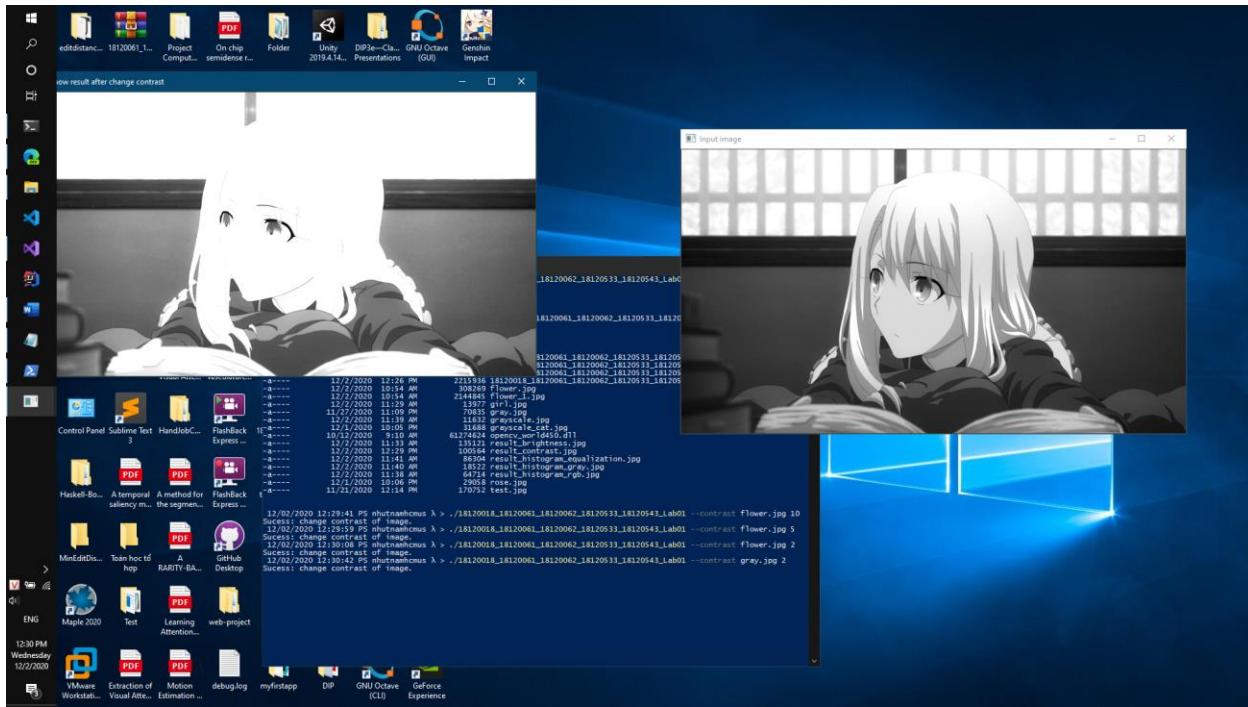


./18120018_18120061_18120062_18120533_18120543_Lab01 --contrast flower.jpg 2

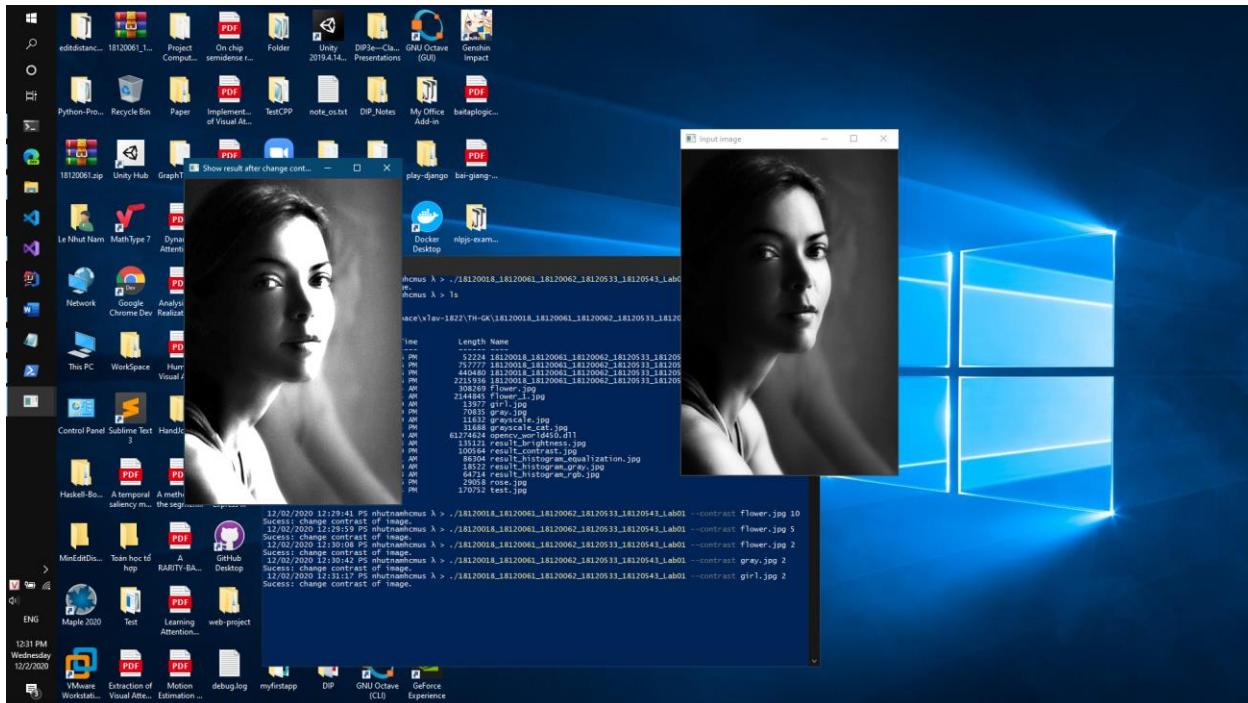


Ảnh xám

./18120018_18120061_18120062_18120533_18120543_Lab01 --contrast gray.jpg 2



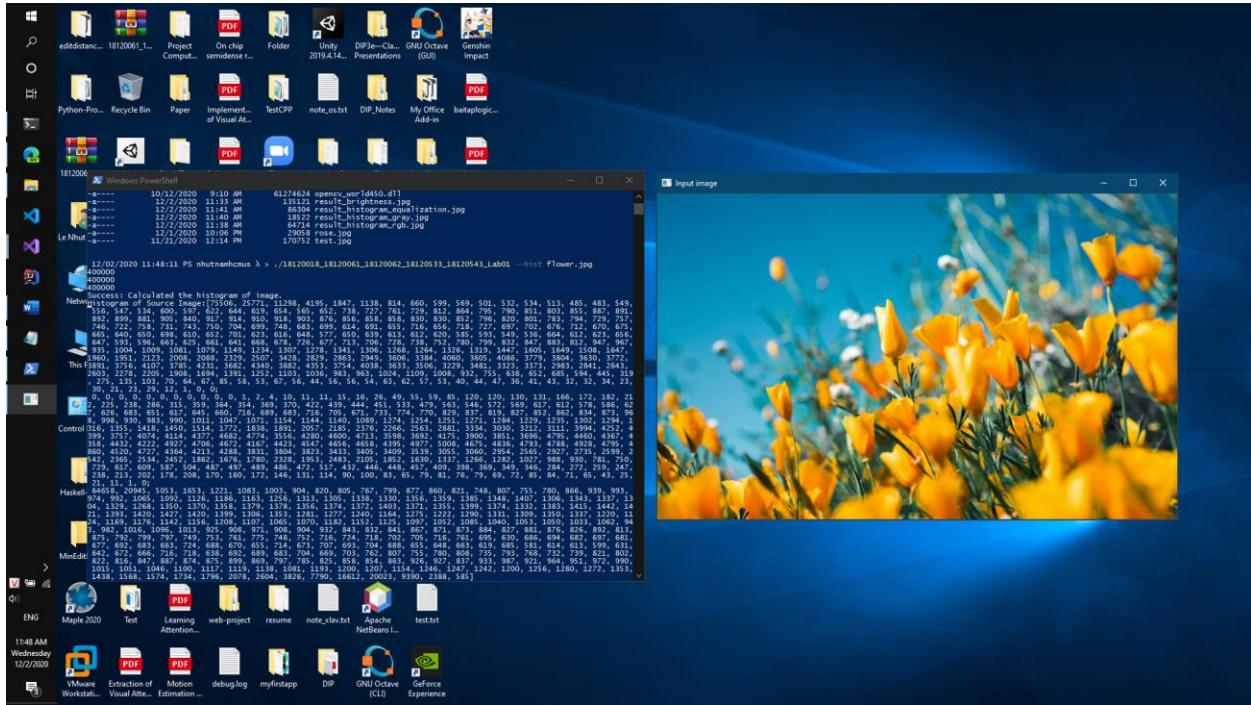
./18120018_18120061_18120062_18120533_18120543_Lab01 --contrast girl.jpg 2



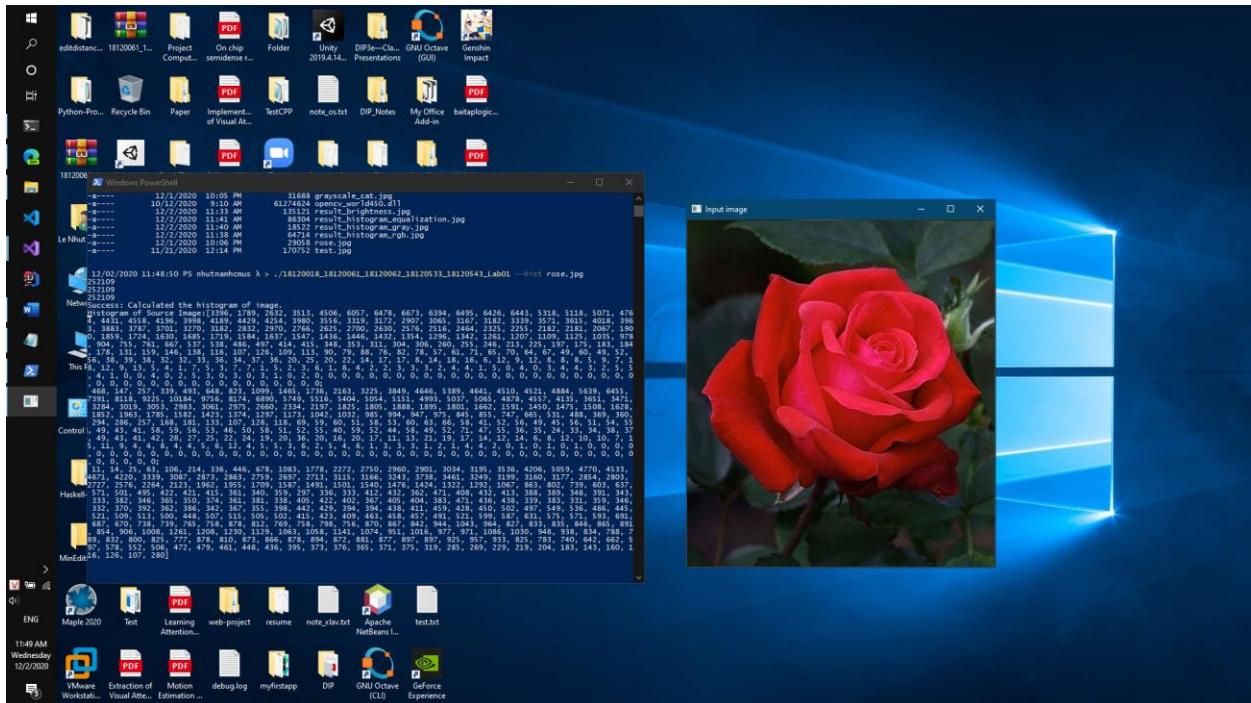
7. Tính histogram của ảnh màu, ảnh xám

Ảnh màu

./18120018_18120061_18120062_18120533_18120543_Lab01 --hist flower.jpg

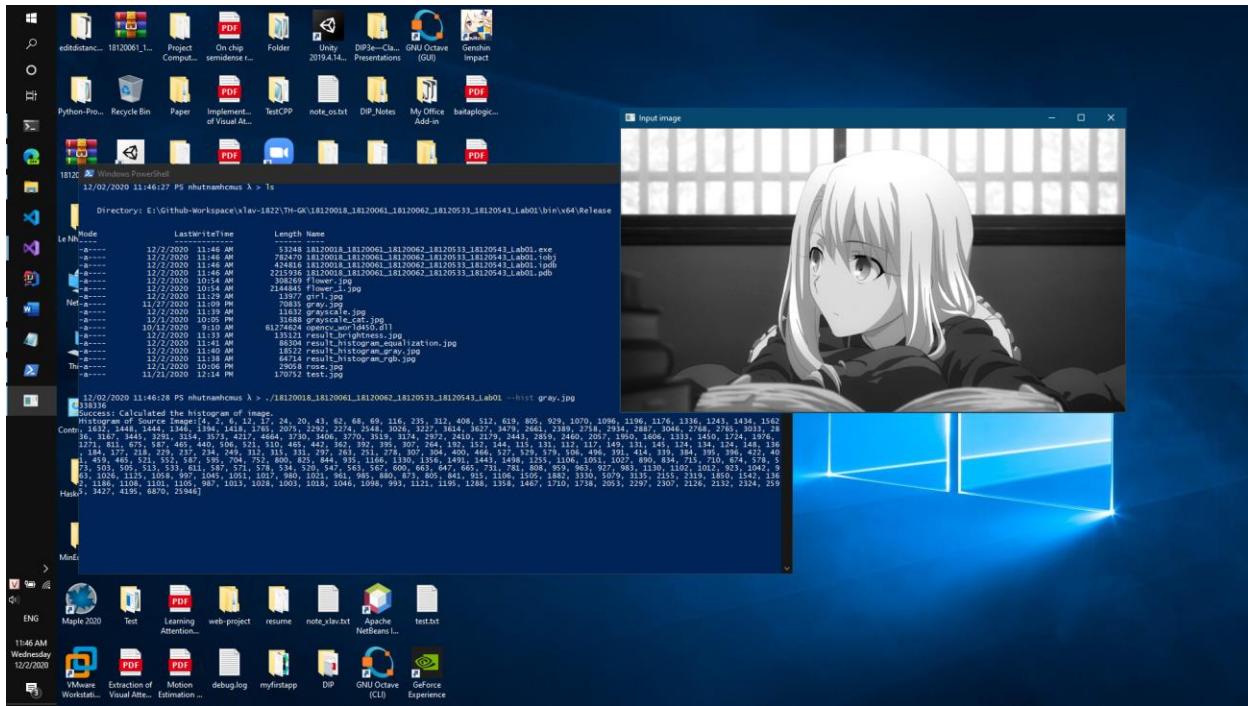


./18120018_18120061_18120062_18120533_18120543_Lab01 --hist rose.jpg

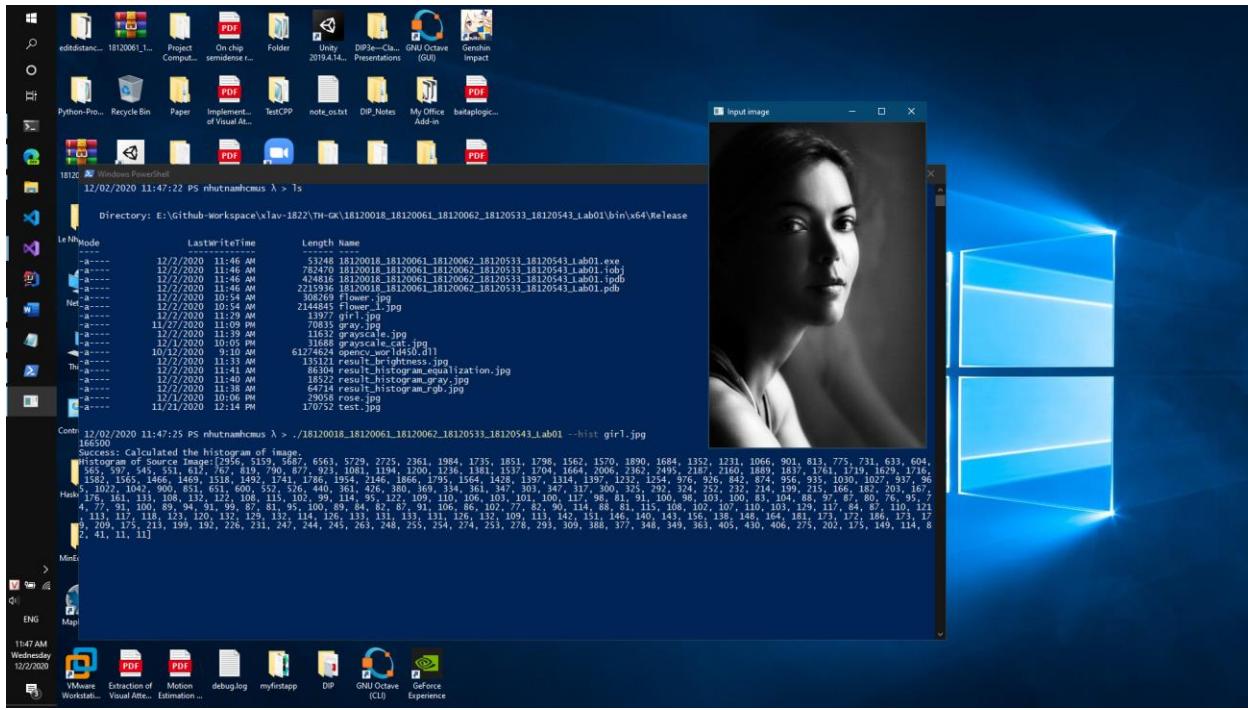


Ảnh xám

./18120018_18120061_18120062_18120533_18120543_Lab01 --hist gray.jpg

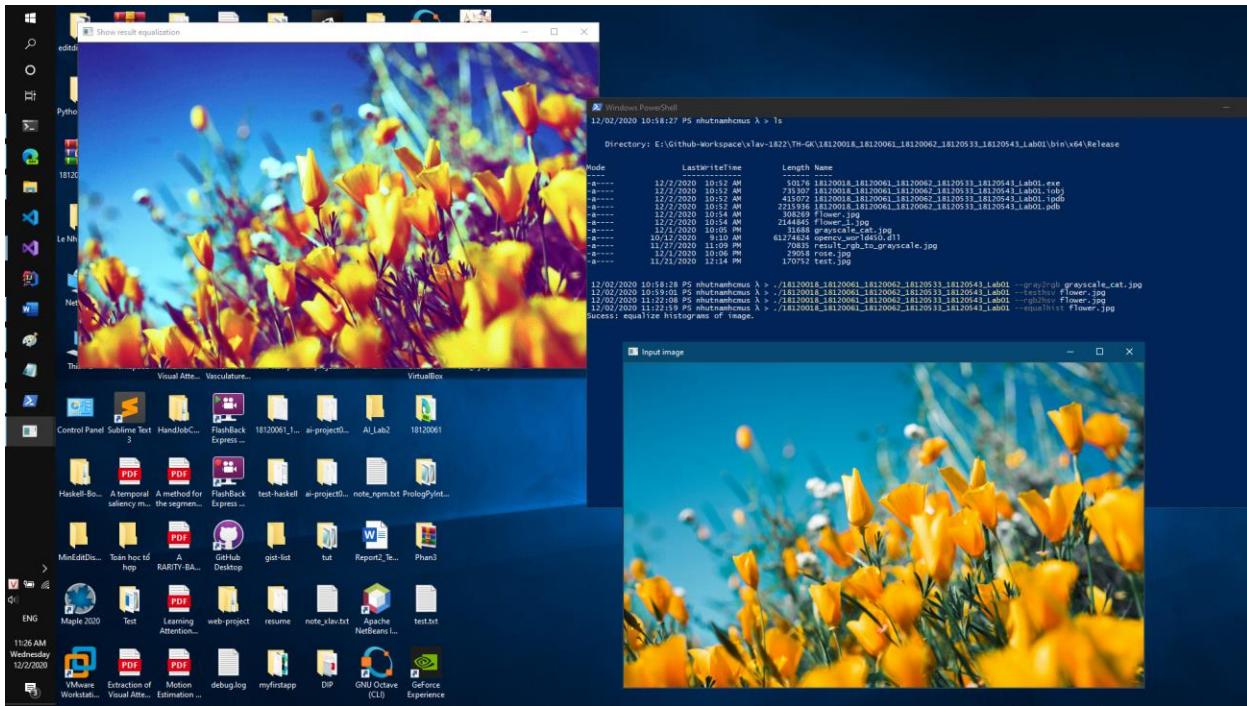


./18120018_18120061_18120062_18120533_18120543_Lab01 --hist girl.jpg

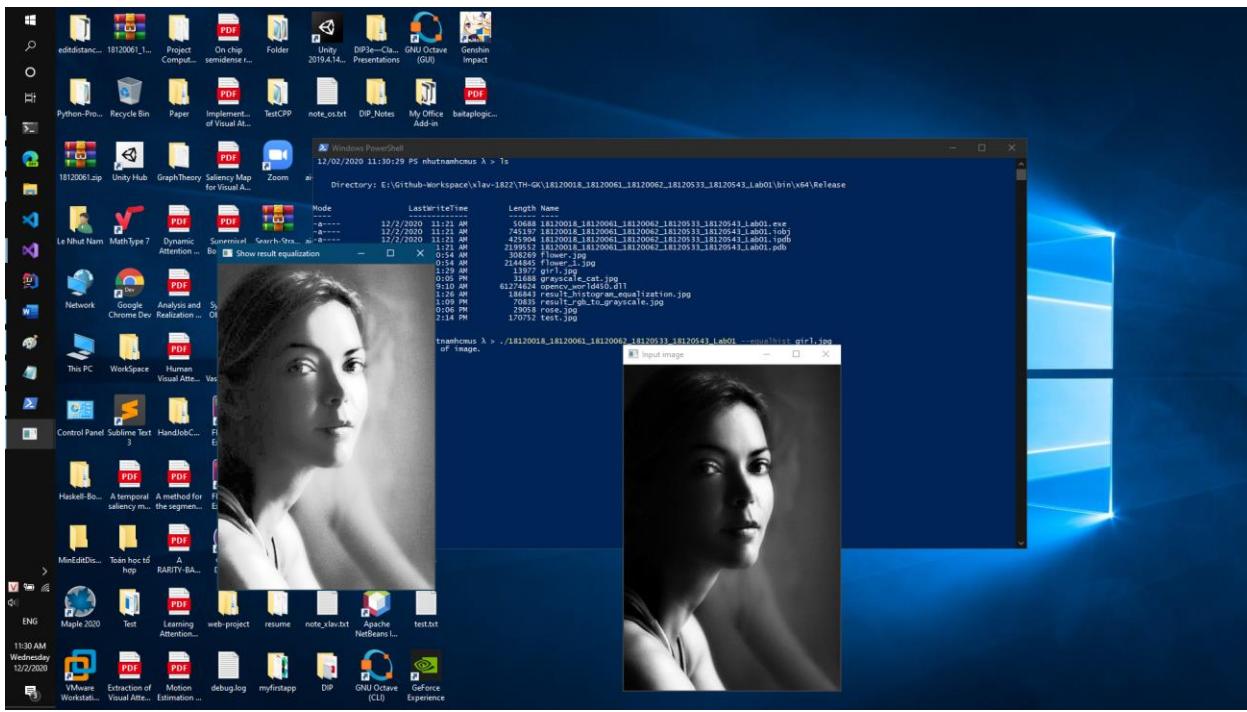


8. Câu 6: Cân bằng histogram ảnh màu, ảnh xám

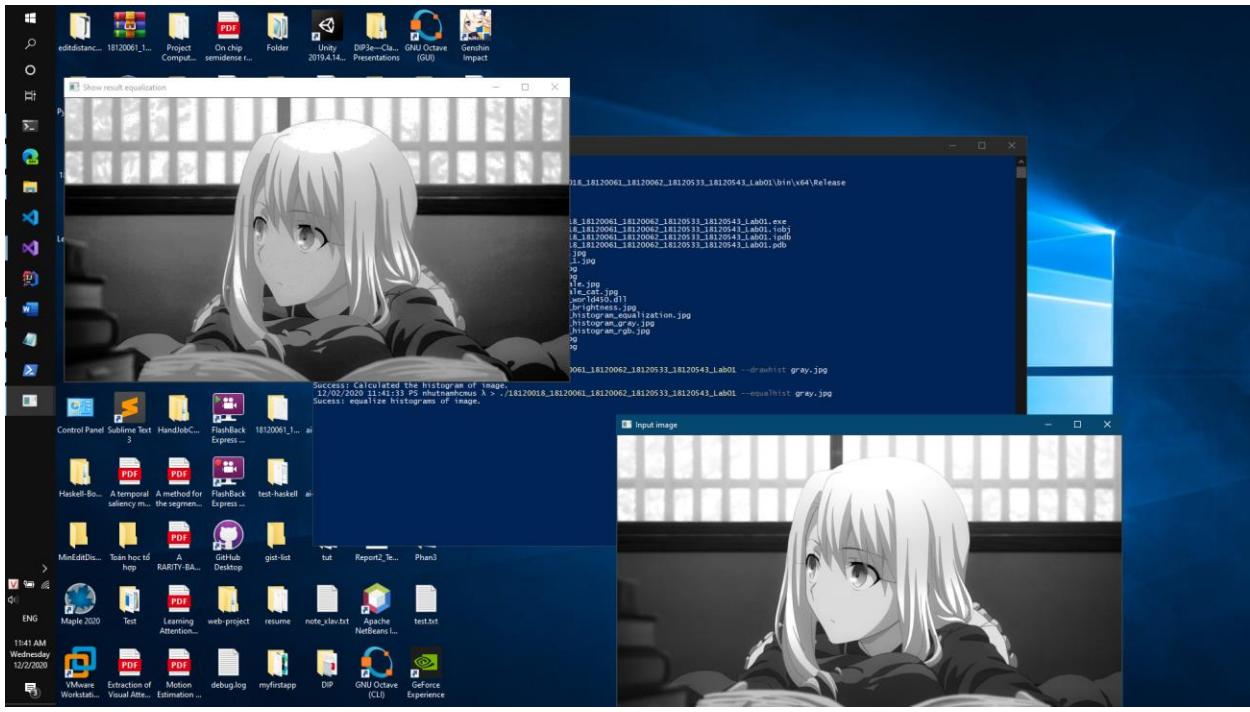
Ảnh màu



Ảnh xám



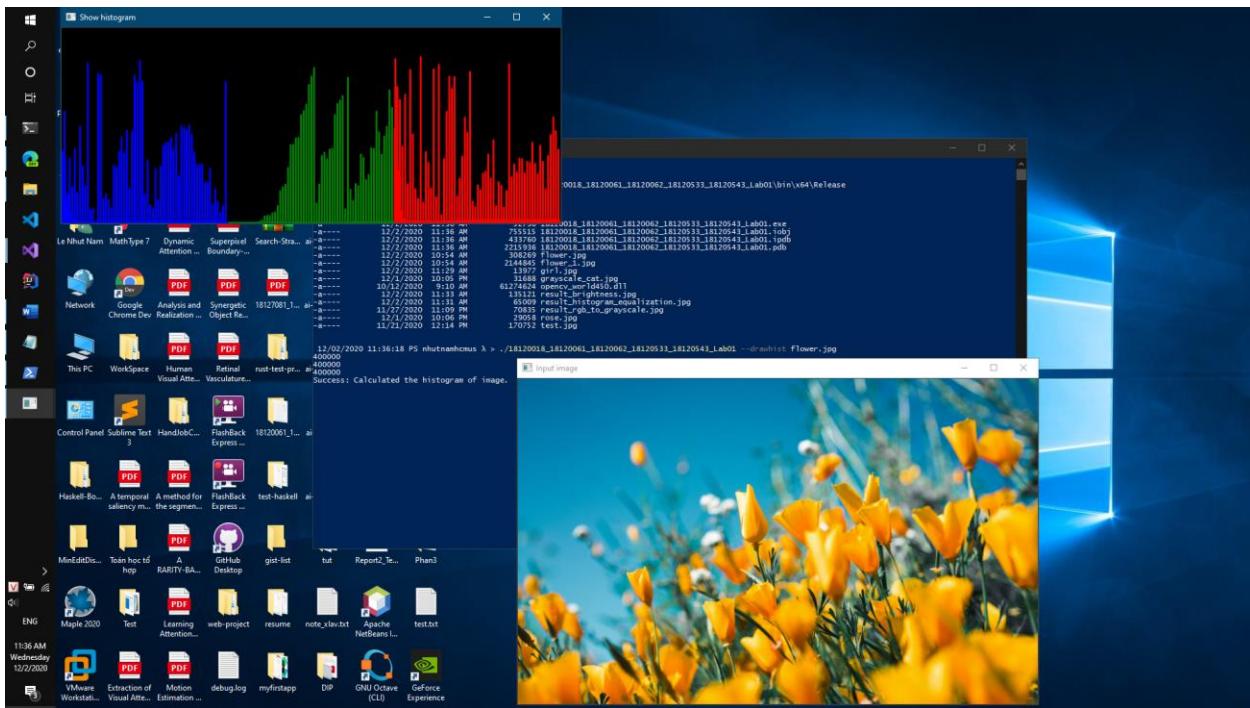
`./18120018_18120061_18120062_18120533_18120543_Lab01 --equalhist gray.jpg`



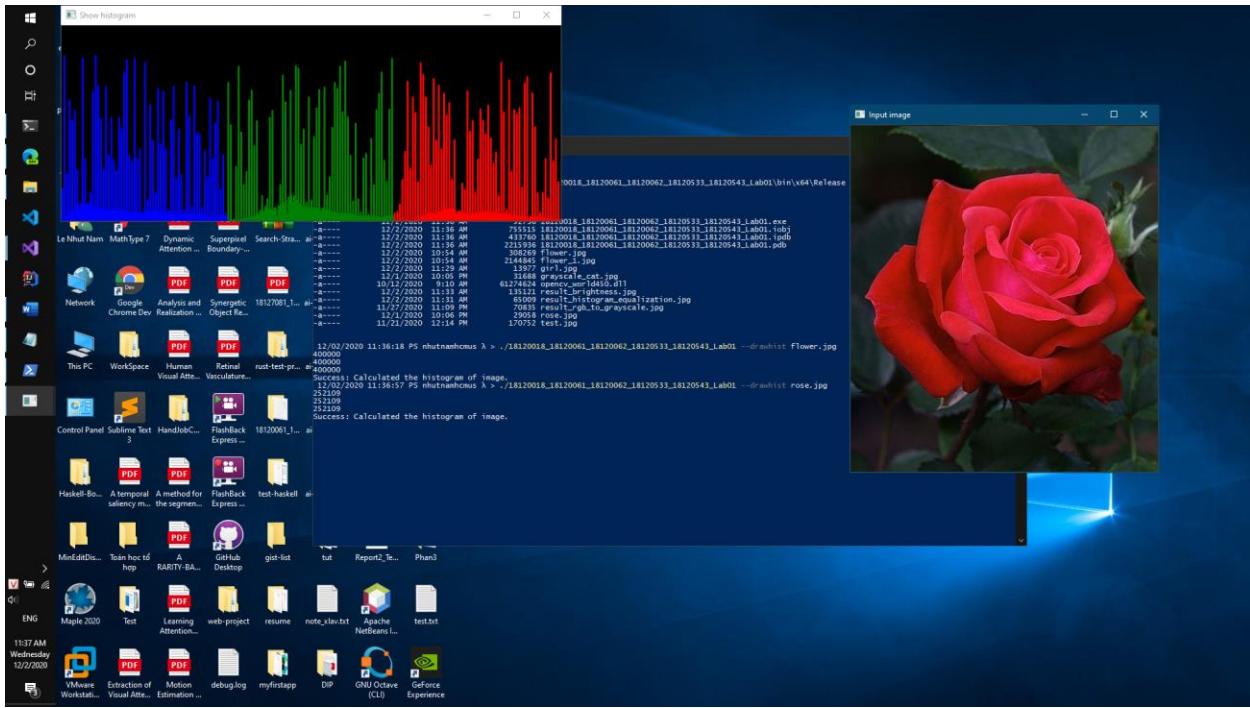
9. Câu 7: Vẽ histogram của ảnh màu, ảnh xám

Ảnh màu

./18120018_18120061_18120062_18120533_18120543_Lab01 --drawhist flower.jpg

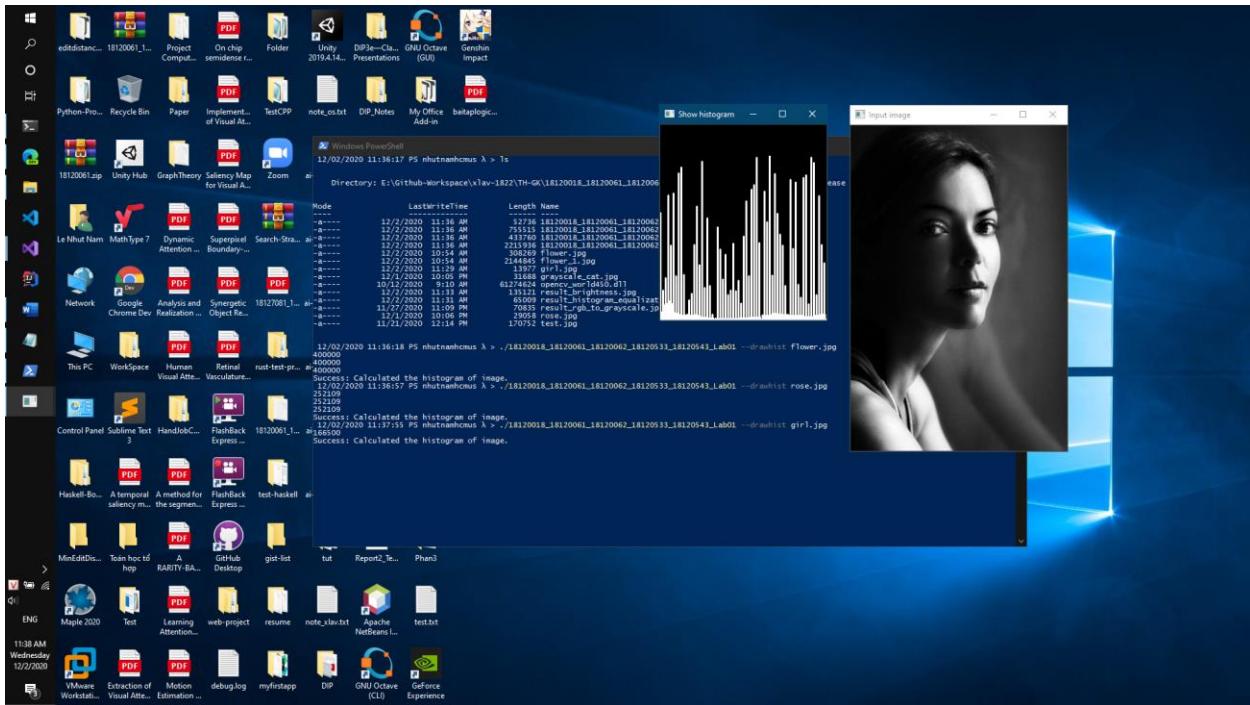


./18120018_18120061_18120062_18120533_18120543_Lab01 --drawhist rose.jpg

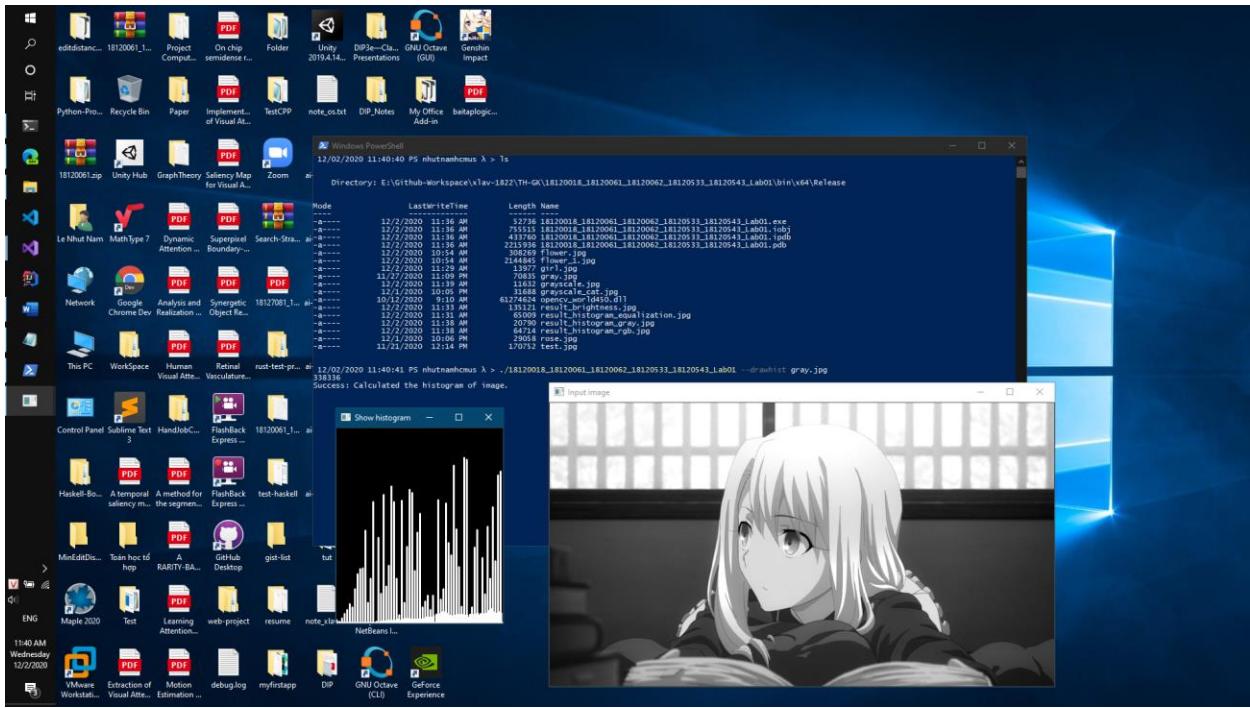


Ảnh xám

`/18120018_18120061_18120062_18120533_18120543_Lab01 --drawhist girl.jpg`



`/18120018_18120061_18120062_18120533_18120543_Lab01 --drawhist gray.jpg`

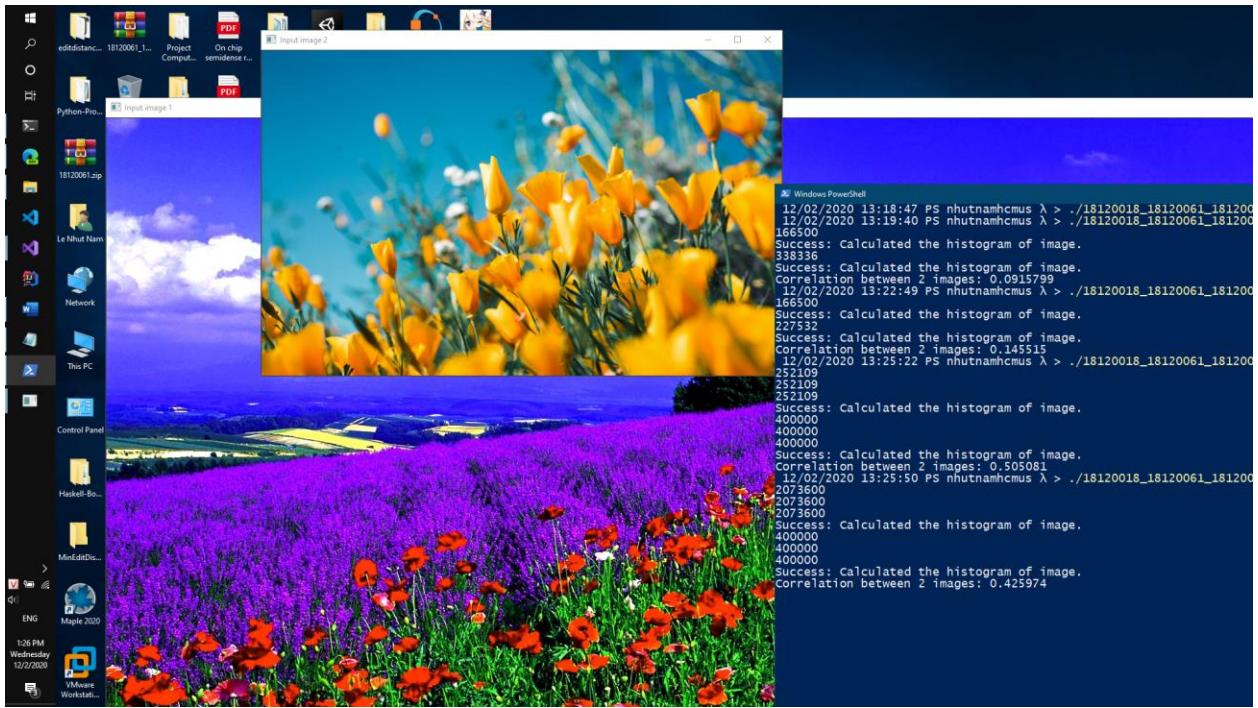


10. Câu 8: So sánh ảnh dựa vào histogram

Ảnh màu

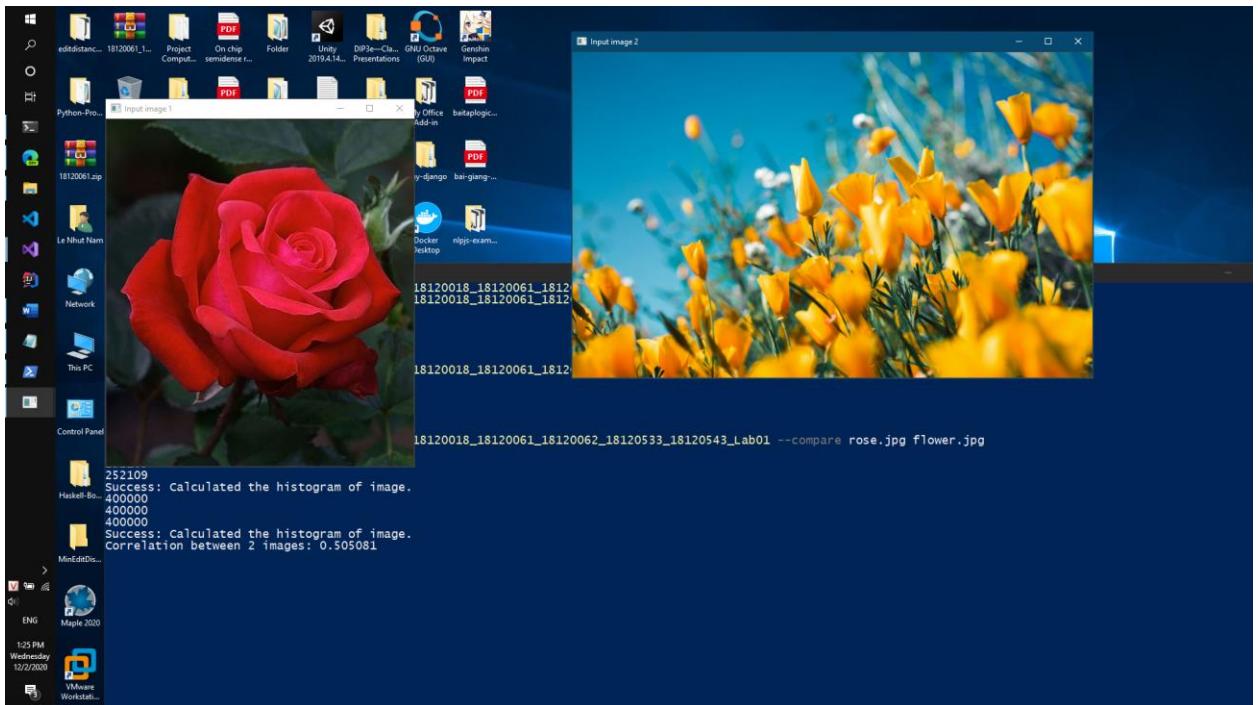
```
./18120018_18120061_18120062_18120533_18120543_Lab01 --compare flower_1.jpg
flower.jpg
```

Correlation between 2 images: 0.425974



./18120018_18120061_18120062_18120533_18120543_Lab01 --compare rose.jpg
flower.jpg

Correlation between 2 images: 0.505081



Ảnh xám

```
./18120018_18120061_18120062_18120533_18120543_Lab01 --compare girl.jpg  
gray.jpg
```

Correlation between 2 images: 0.0915799



```
./18120018_18120061_18120062_18120533_18120543_Lab01 --compare girl.jpg  
mother.png
```

Correlation between 2 images: 0.145515

