实验一：图像灰度变换

学号：SA22225286 姓名：孟寅磊 日期：20220921

实验内容

> 1. 利用OpenCV读取图像
>
>   具体内容：用OpenCV打开图像，并在窗口中显示。
> 2. 灰度图像二值化处理
>
>   具体内容：设置并调整阈值对图像进行二值化处理。
> 3. 灰度图像的对数变换
>
>   具体内容：设置并调整r值对图像进行对数变换。
> 4. 灰度图像的伽马变换
>
>   具体内容：设置并调整γ值对图像进行伽马变换。
> 5. 彩色图像的补色变换
>
>   具体内容：对彩色图像进行补色变换。

实验完成情况

> ☑ 利用OpenCV读取图像

使用`imgproc`模块中的`imread`函数读取图像，成功返回`Mat`矩阵对象，失败返回空矩阵。
使用`imgproc`模块中的`imshow`函数显示图像。

```
int main(int argc, char **argv) {
    Mat image1 = imread("images/bird.jpg");
    if (image1.empty()) {
        cout << "Could not read the image." << endl;
        return 1;
    }
    imshow("Image", image1);
    waitKey(0);
    return 0;
}
```

> ☑ 灰度图像二值化处理

遍历每个像素，判断其灰度值。当灰度值大于某一阈值时，置灰度值为255，小于等于阈值时置0。当目标和背景像素的灰度分布非常不同时，可对整个图像使用全局阈值。本次实验使用下面的迭代算法。

```
/*
 * 全局阈值的迭代算法：
 * 1. 为全局阈值T选择一个初始估计值(选用图像的平均灰度作为初始值最好)。
 * 2. 用T分割图像。这将产生两组像素：由灰度值大于T的所有像素组成的G1，由所有小于等于T的像素组成的G2。
 * 3. 对G1和G2中的像素分别计算平均灰度值m1和m2。
 * 4. 计算新的阈值T = (m1 + m2) / 2。
```

```
 * 5. 重复步骤2到4，直到连续两次迭代的值的差小于某个预定义的值delta为止。
 */
void global_threshold_binaryzation(Mat &src, Mat &dst) {
    dst = src.clone();
    const int GRAY_SCALE = 256;
    const double delta_T = 0.1;
    int histogram[GRAY_SCALE] = {0};
    // 直方图统计
    for (int i = 0; i < src.rows; ++i) {
        uchar *p = src.ptr<uchar>(i);
        for (int j = 0; j < src.cols; ++j)
            ++histogram[p[j]];
    }
    // 构造前缀和数组用于计算平均灰度值
    int cnt[GRAY_SCALE] = {histogram[0]};
    for (int i = 1; i < GRAY_SCALE; ++i)
        cnt[i] = cnt[i-1] + histogram[i];
    double sum[GRAY_SCALE] = {0.0};
    for (int i = 1; i < GRAY_SCALE; ++i)
        sum[i] = sum[i-1] + histogram[i] * i;
    // 迭代计算全局阈值
    double old_threshold = mean(src)[0];
    double new_threshold = 0.0;
    for (;;) {
        int t = static_cast<int>(old_threshold + 0.5);
        double m1 = sum[t] / cnt[t];
        double m2 = (sum[GRAY_SCALE-1] - sum[t]) /
                    (cnt[GRAY_SCALE-1] - cnt[t]);
        new_threshold = (m1 + m2) / 2;
        if (abs(new_threshold - old_threshold) < delta_T)
            break;
        else
            old_threshold = new_threshold;
    }
    // 根据计算出的阈值对图像进行二值化处理
    int threshold = static_cast<int>(new_threshold + 0.5);
    for (int i = 0; i < dst.rows; ++i) {
        uchar *p = dst.ptr<uchar>(i);
        for (int j = 0; j < dst.cols; ++j)
            p[j] = (p[j] > threshold) ? 255 : 0;
    }
}
```

> ☑ 灰度图像的对数变换

对数变换的通式为$s = log(1 + r)$，这个变换将输入中范围较窄的低灰度值映射为输出中范围较宽的灰度级，将输入中的高灰度值映射为输出中范围较窄的灰度级。我们使用这类变换来扩展图像中的暗像素值，同时压缩高灰度级值。

```cpp
void log_transform(Mat &src, Mat &dst, double c) {
    dst.create(src.size(), src.type());
    for (int i = 0; i < dst.rows; ++i)
        for (int j = 0; j < dst.cols; ++j)
            dst.at<uchar>(i, j) = saturate_cast<uchar>(c * log(1.0 +
src.at<uchar>(i, j)));
}
```

> ☑ 灰度图像的伽马变换

伽马变换的公式为$s = cr^{\gamma}$。主要用于图像的校正，对漂白的图片或者过黑的图片进行修正，也就是对灰度级过高或者灰度级过低的图片进行修正，增强对比度。

```cpp
void gamma_transform(Mat &src, Mat &dst, double c, double gamma) {
    dst.create(src.size(), src.type());
    for (int i = 0; i < src.rows; ++i)
        for (int j = 0; j < src.cols; ++j)
            dst.at<uchar>(i, j) = saturate_cast<uchar>(c * pow(src.at<uchar>
(i, j), gamma));
}
```

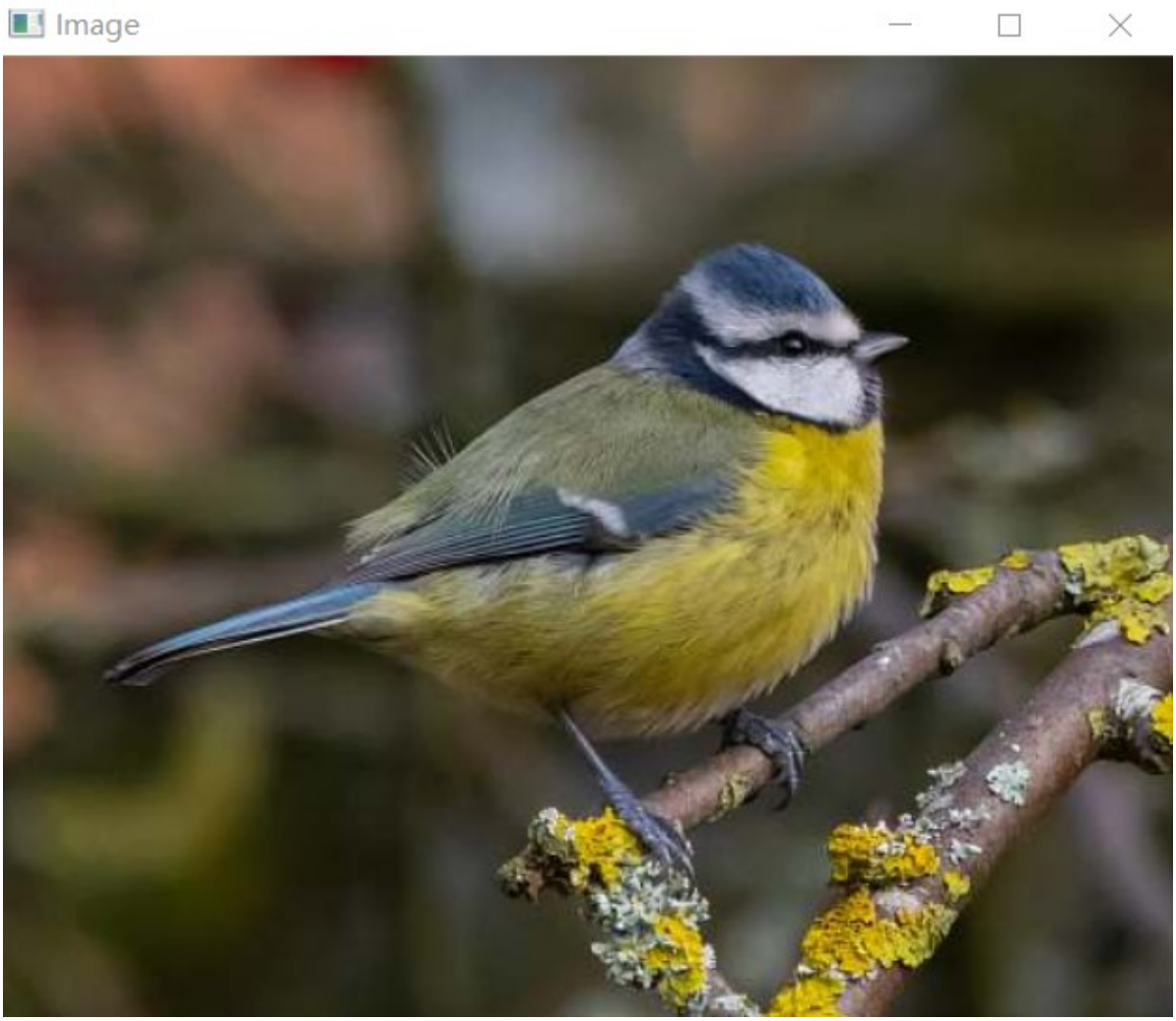
> ☑ 彩色图像的补色变换

在艾萨克·牛顿创建的彩色环中，两端对应的颜色是互补的。补色变换可用于增强彩色图像中各个暗色区域中的细节，尤其是在这些区域的尺寸较大时。

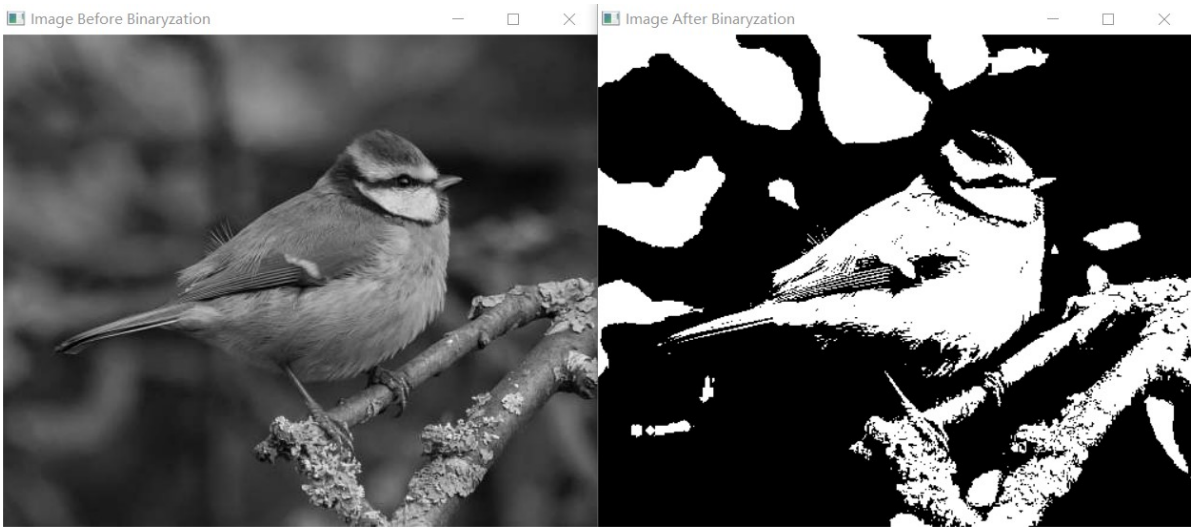
```cpp
void rgb_comp_transform(Mat &src, Mat &dst) {
    dst.create(src.size(), src.type());
    for (int i = 0; i < dst.rows; ++i) {
        for (int j = 0; j < dst.cols; ++j) {
            dst.at<Vec3b>(i, j)[0] = 255 - src.at<Vec3b>(i, j)[0];
            dst.at<Vec3b>(i, j)[1] = 255 - src.at<Vec3b>(i, j)[1];
            dst.at<Vec3b>(i, j)[2] = 255 - src.at<Vec3b>(i, j)[2];
        }
    }
}
```

实验结果

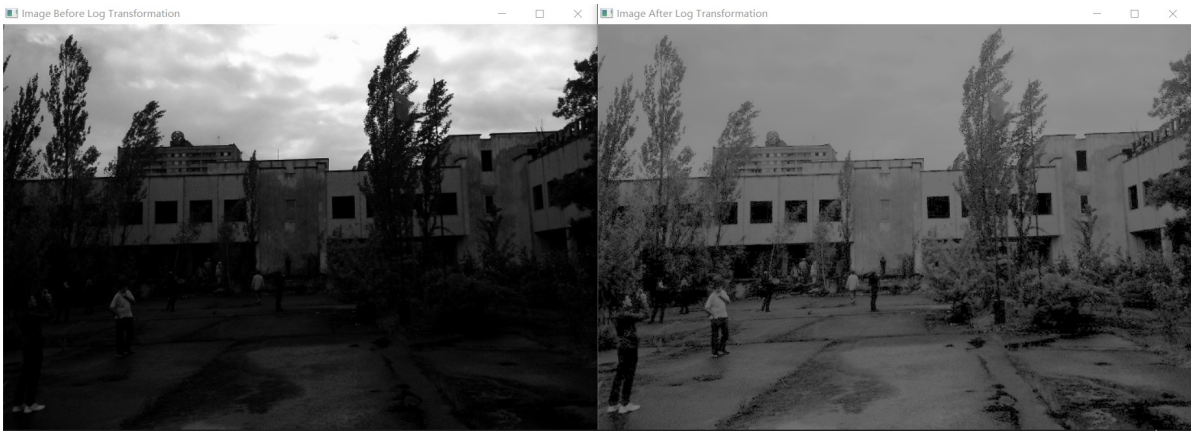
> 图像的显示



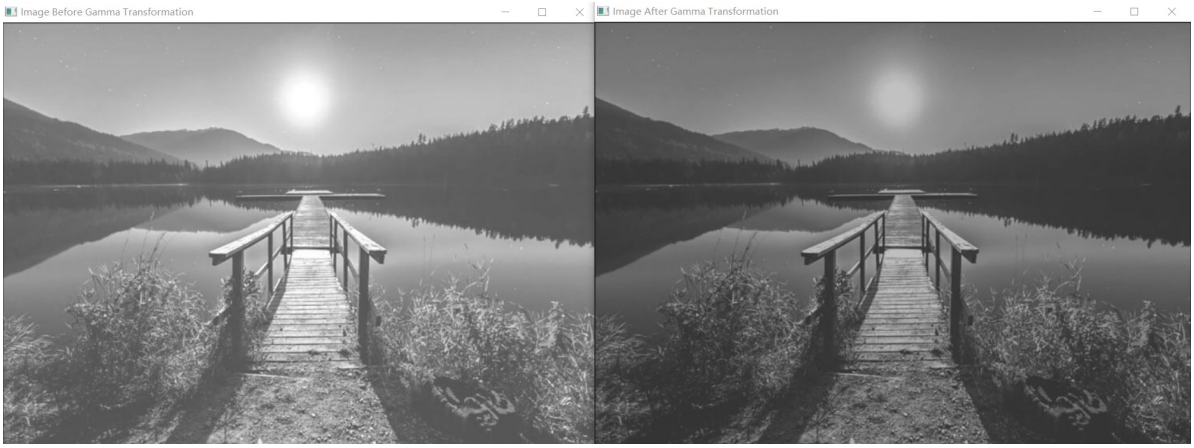
灰度图像的二值化处理



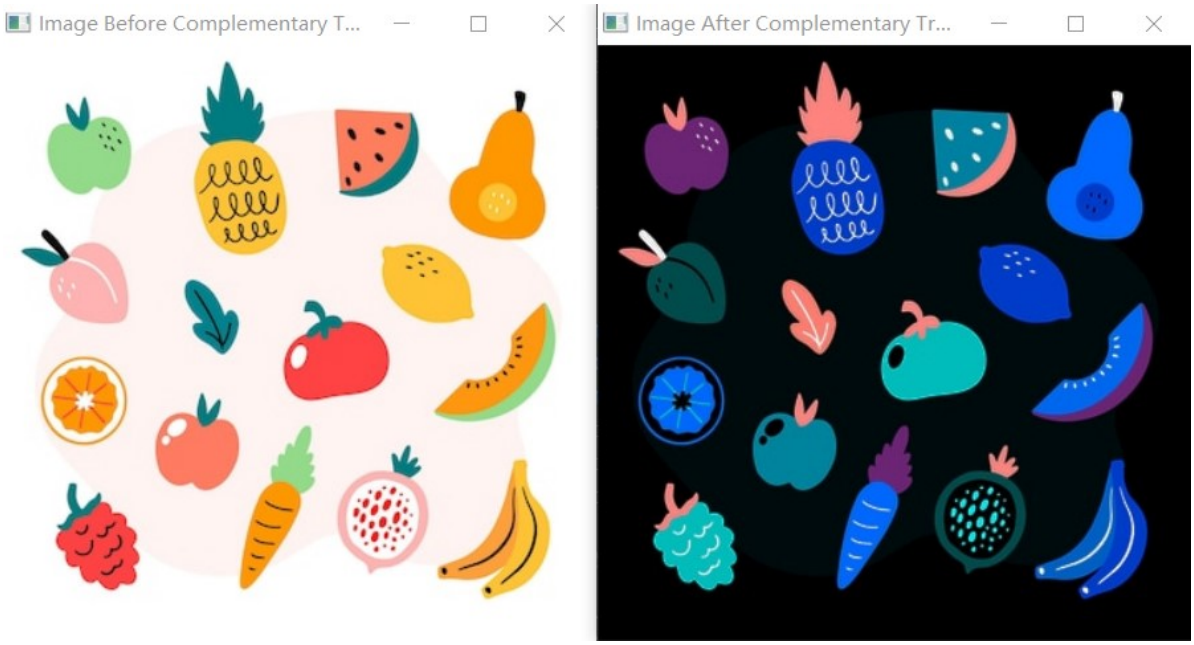
灰度图像的对数变换



> 灰度图像的伽马变换



> 彩色图像的补色变换



完整的源代码见附件`lab1.cpp`。