【图像处理】-015 空域滤波处理-高斯滤波

在讨论频域滤波时，我们讨论了高斯低通滤波器、高斯高通滤波器等，这里，我们将对空域中的高斯滤波器进行讨论。

1 理论依据

1.1 空间域中的高斯滤波器

高斯低通滤波器(GLPF)的数学表达式如下：

$$H(u,v)=1-e^{-D^2(u,v)/2\sigma^2} \tag{1}$$

通常讨论时，可以去截止频率D_0，表示形式如下：

$$H(u,v)=1-e^{-D^2(u,v)/2D_0^2} \tag{2}$$

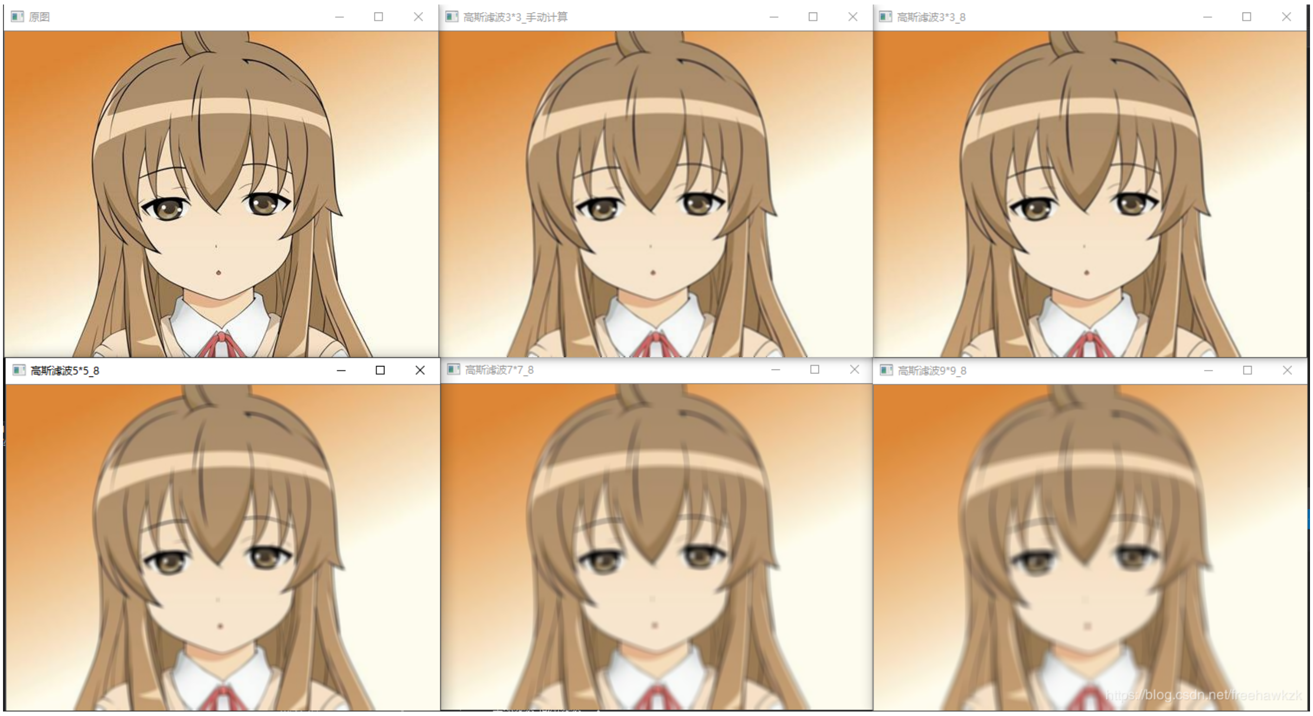
2 实现

```cpp
#include "../include/importOpenCV.h"
#include "../include/baseOps.h"
#include "../include/opencv400/opencv2/core.hpp"
#include <iostream>



int main()
{
    //将工作目录设置到EXE所在的目录。
    SetCurrentDirectoryToExePath();

    cv::Mat src = cv::imread("../images/71.jpg");
    cv::imshow("原图", src);

    cv::Mat gaussianFilter2;
    SGLPFParam param;
    param.r = 8;
    param.sz = cv::Size(3, 3);
    CreateGaussianLowpassFilter(param, gaussianFilter2);
    std::vector<cv::Mat> w;
    cv::split(gaussianFilter2, w);
    cv::Scalar ssum = cv::sum(w[0]);
    w[0] = w[0] / ssum.val[0];
    cv::Mat& w1 = w[0];
    cv::Mat output;
    src.copyTo(output);
    if (src.channels() == 3)
    {
        std::vector<cv::Mat> srcbgr;
        cv::split(src, srcbgr);
        std::vector<cv::Mat> dstbgr;
        cv::split(output, dstbgr);

        for (int i = 1; i < srcbgr[0].rows-1; i ++)
        {
            for (int j = 1; j < srcbgr[0].cols-1; j++)
            {
                dstbgr[0].at<uchar>(i, j) = (uchar)(srcbgr[0].at<uchar>(i - 1, j - 1)*w1.at<float>(0, 0) + srcbgr[0].at<uchar>(i - 1, j)*w1.at<float>(0, 1) + srcbgr[0].at<uchar>(i - 1, j + 1)*w1.at<float>(0, 2) + \
                                                    srcbgr[0].at<uchar>(i    , j - 1)*w1.at<float>(1, 0) + srcbgr[0].at<uchar>(i    , j)*w1.at<float>(1, 1) + srcbgr[0].at<uchar>(i    , j + 1)*w1.at<float>(1, 2) + \
                                                    srcbgr[0].at<uchar>(i + 1, j - 1)*w1.at<float>(2, 0) + srcbgr[0].at<uchar>(i + 1, j)*w1.at<float>(2, 1) + srcbgr[0].at<uchar>(i + 1, j + 1)*w1.at<float>(2, 2));

                dstbgr[1].at<uchar>(i, j) = (uchar)(srcbgr[1].at<uchar>(i - 1, j - 1)*w1.at<float>(0, 0) + srcbgr[1].at<uchar>(i - 1, j)*w1.at<float>(0, 1) + srcbgr[1].at<uchar>(i - 1, j + 1)*w1.at<float>(0, 2) + \
                                                    srcbgr[1].at<uchar>(i    , j - 1)*w1.at<float>(1, 0) + srcbgr[1].at<uchar>(i    , j)*w1.at<float>(1, 1) + srcbgr[1].at<uchar>(i    , j + 1)*w1.at<float>(1, 2) + \
                                                    srcbgr[1].at<uchar>(i + 1, j - 1)*w1.at<float>(2, 0) + srcbgr[1].at<uchar>(i + 1, j)*w1.at<float>(2, 1) + srcbgr[1].at<uchar>(i + 1, j + 1)*w1.at<float>(2, 2));

                dstbgr[2].at<uchar>(i, j) = (uchar)(srcbgr[2].at<uchar>(i - 1, j - 1)*w1.at<float>(0, 0) + srcbgr[2].at<uchar>(i - 1, j)*w1.at<float>(0, 1) + srcbgr[2].at<uchar>(i - 1, j + 1)*w1.at<float>(0, 2) + \
                                                    srcbgr[2].at<uchar>(i    , j - 1)*w1.at<float>(1, 0) + srcbgr[2].at<uchar>(i    , j)*w1.at<float>(1, 1) + srcbgr[2].at<uchar>(i    , j + 1)*w1.at<float>(1, 2) + \
                                                    srcbgr[2].at<uchar>(i + 1, j - 1)*w1.at<float>(2, 0) + srcbgr[2].at<uchar>(i + 1, j)*w1.at<float>(2, 1) + srcbgr[2].at<uchar>(i + 1, j + 1)*w1.at<float>(2, 2));
            }
        }

        cv::merge(dstbgr, output);
        cv::imshow("高斯滤波3*3_手动计算", output);
        cv::Mat dst1;
        cv::GaussianBlur(src, dst1, cv::Size(3, 3),8);
        cv::imshow("高斯滤波3*3_cv::blur", dst1);
        cv::GaussianBlur(src, dst1, cv::Size(5, 5),8);
        cv::imshow("高斯滤波5*5_cv::blur", dst1);
        cv::GaussianBlur(src, dst1, cv::Size(7, 7), 8);
        cv::imshow("高斯滤波7*7_cv::blur", dst1);
        cv::GaussianBlur(src, dst1, cv::Size(9, 9), 8);
        cv::imshow("高斯滤波9*9_cv::blur", dst1);
    }
    else
    {

        for (int i = 1; i < src.rows - 1; i++)
        {
            for (int j = 1; j < src.cols - 1; j++)
            {
                output.at<uchar>(i, j) = (uchar)(src.at<uchar>(i - 1, j - 1)*w1.at<float>(0, 0) + src.at<uchar>(i - 1, j)*w1.at<float>(0, 1) + src.at<uchar>(i - 1, j + 1)*w1.at<float>(0, 2) + \
                                                 src.at<uchar>(i    , j - 1)*w1.at<float>(1, 0) + src.at<uchar>(i    , j)*w1.at<float>(1, 1) + src.at<uchar>(i    , j + 1)*w1.at<float>(1, 2) + \
                                                 src.at<uchar>(i + 1, j - 1)*w1.at<float>(2, 0) + src.at<uchar>(i + 1, j)*w1.at<float>(2, 1) + src.at<uchar>(i + 1, j + 1)*w1.at<float>(2, 2));
            }
        }

        cv::Mat dst1;
        cv::GaussianBlur(src, dst1, cv::Size(3, 3), 8);
        cv::imshow("高斯滤波3*3_手动计算", output);
        cv::imshow("高斯滤波3*3_cv::blur", dst1);
    }

    cv::waitKey();
    return 0;
}
```

3 讨论

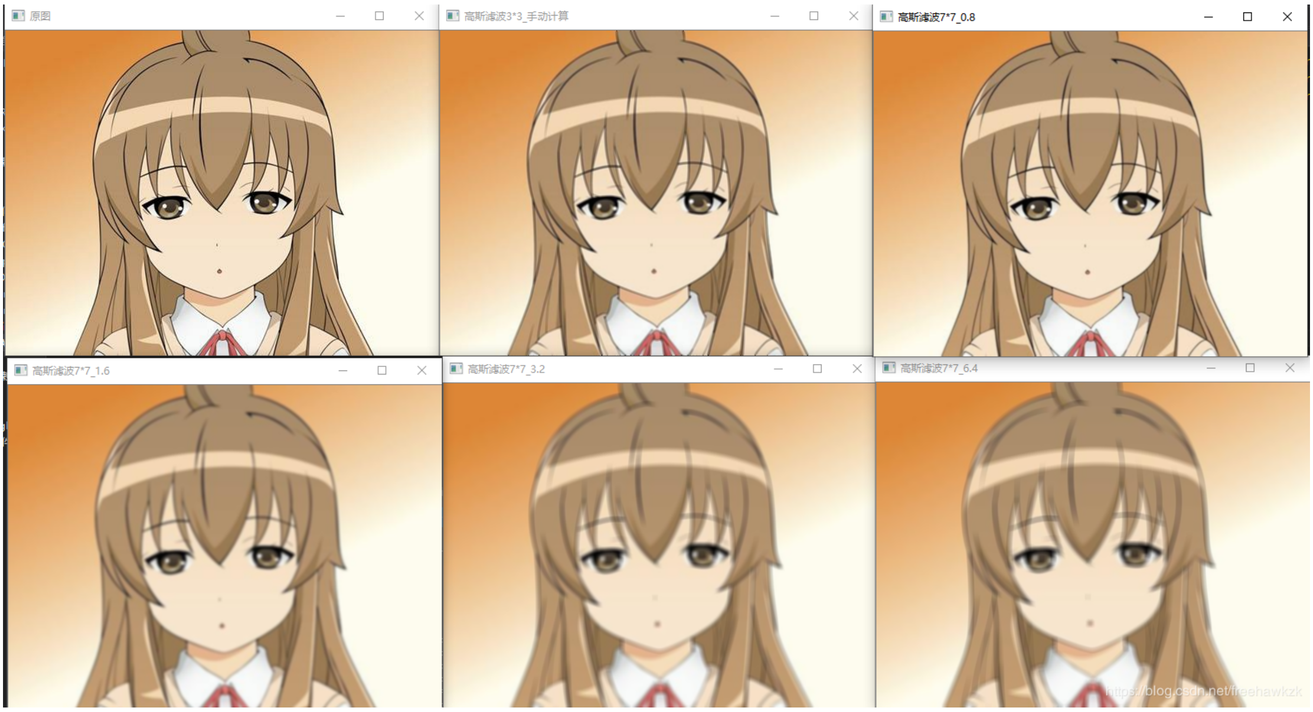
在OpenCV中，高斯滤波通过`GaussianBlur`函数来实现。高斯模糊的效果受高斯滤波器的尺寸和方差控制。

3.1 不同尺寸，相同方差



可以看出，在方差相同时，随着滤波器尺寸的增加，图像的模糊效果逐渐加重。

3.2 相同尺寸，不同方差



可以看出，在滤波器尺寸相同时，方差逐渐增加，模糊效果会加强。