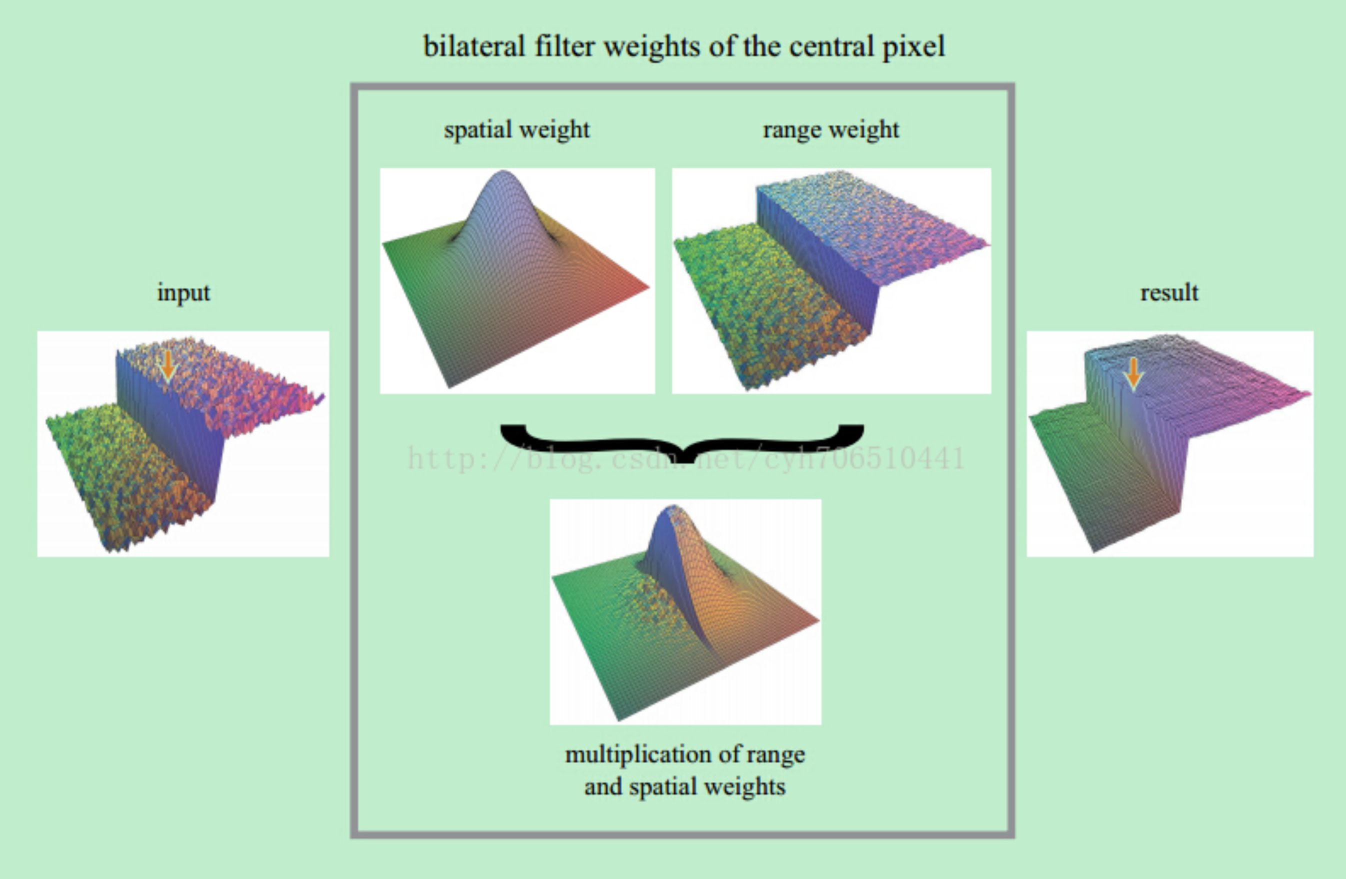
双边滤波原理与C++实现

一、原理

双边滤波（Bilateral filter）是一种可以去噪保边的滤波器。之所以可以达到此效果，是因为滤波器是由两个函数构成：一个函数是由几何空间距离决定滤波器系数，另一个由像素差值决定滤波器系数。

原理示意图如下：



双边滤波器中，输出像素的值依赖于邻域像素的值的加权组合，

权重系数w(i,j,k,l)取决于定义域核

和值域核

的乘积

$$w(i,j,k,l) = \exp\left(-\frac{(i-k)^2+(j-l)^2}{2\sigma_d^2} - \frac{\|f(i,j)-f(k,l)\|^2}{2\sigma_r^2}\right).$$

二、C++实现

2.1 OpenCV调用方法：

```
cvSmooth(m_iplImg, dstImg, CV_BILATERAL, 2 * r + 1, 0, sigma_r, sigma_d);
```

2.2 MATLAB版代码：

http://www.mathworks.com/matlabcentral/fileexchange/12191-bilateral-filtering/content/Bilateral%20Filtering/bfilter2.m

调用方法参见资料[1]

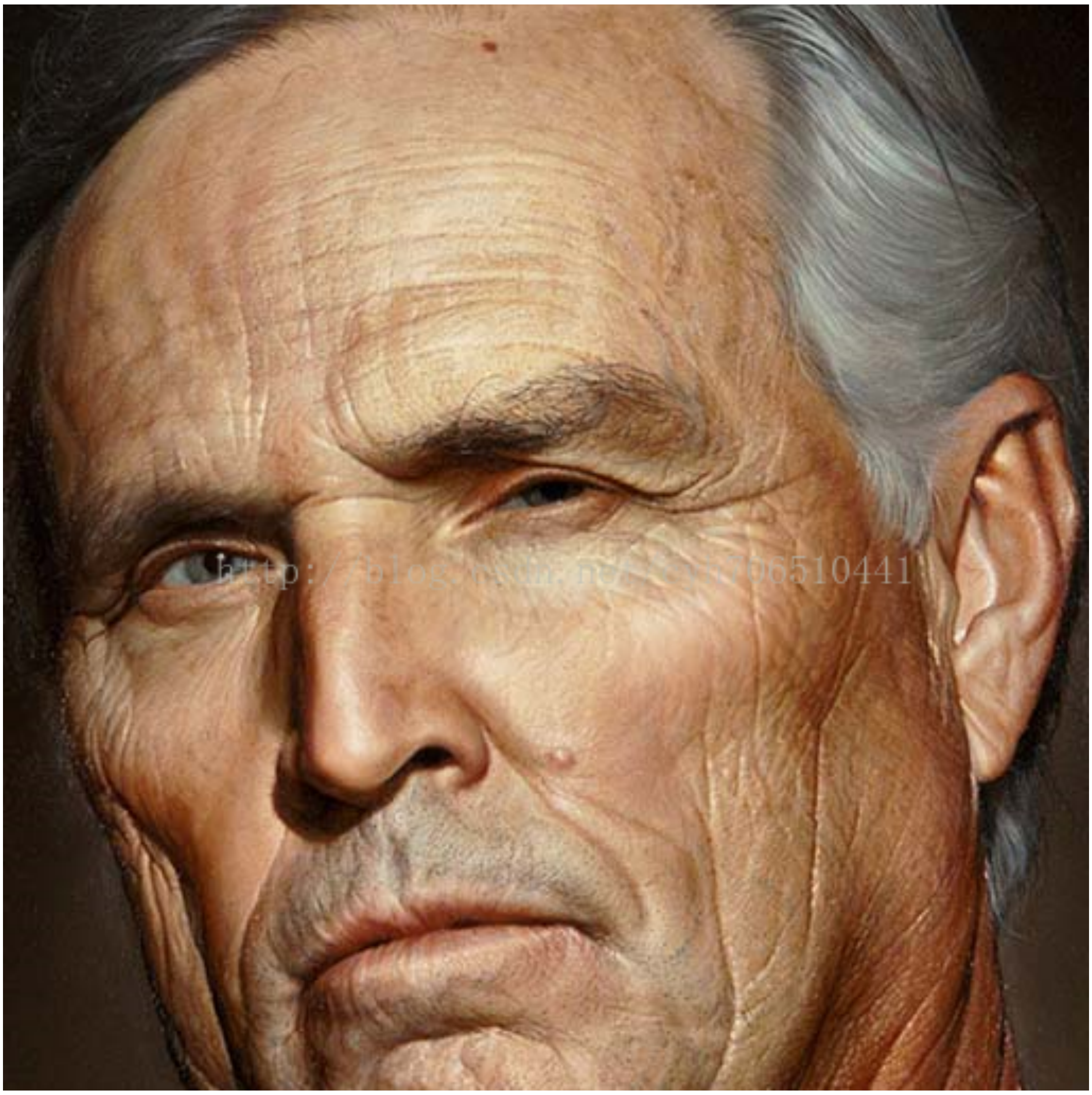
2.3 C++代码

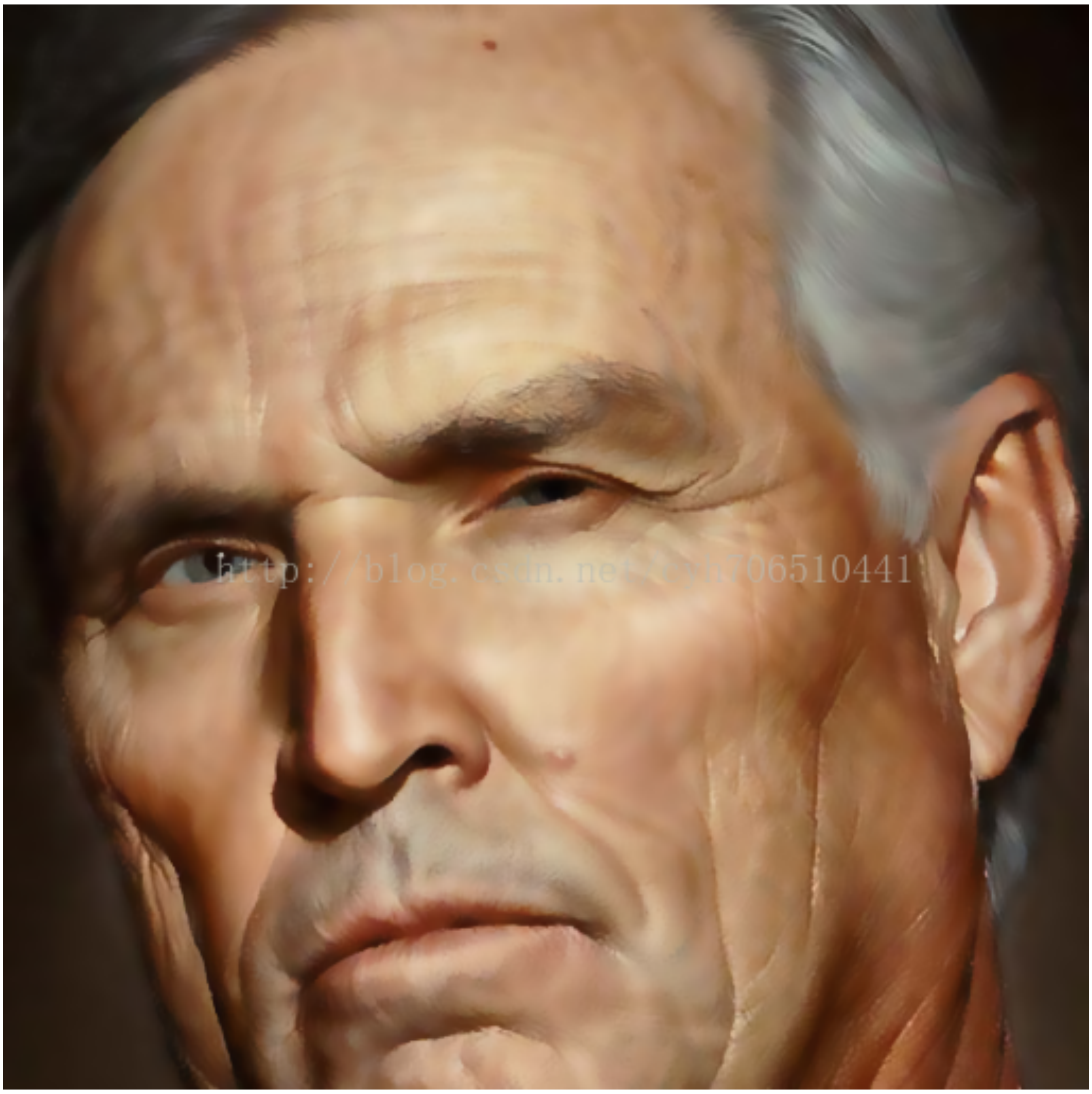
```
void CImageObj::Bilateral_Filter(int r, double sigma_d, double sigma_r)
{
    int i, j, m, n, k;
    int nx = m_width, ny = m_height;
    int w_filter = 2 * r + 1;

    double gaussian_d_coeff = -0.5 / (sigma_d * sigma_d);
    double gaussian_r_coeff = -0.5 / (sigma_r * sigma_r);

    double** d_metrix = NewDoubleMatrix(w_filter, w_filter);
    double r_metrix[256];


    double* img_tmp = new double[m_nChannels * nx * ny];
    for (i = 0; i < ny; i++)
```

```
        for (j = 0; j < nx; j++)
            for (k = 0; k < m_nChannels; k++)
            {
                img_tmp[i * m_nChannels * nx + m_nChannels * j + k] = m_imgData[i * m_nChannels * nx + m_nChannels * j + k];
            }


    for (i = -r; i <= r; i++)
        for (j = -r; j <= r; j++)
        {
            int x = j + r;
            int y = i + r;

            d_metrix[y][x] = exp((i * i + j * j) * gaussian_d_coeff);
        }


    for (i = 0; i < 256; i++)
    {
        r_metrix[i] = exp(i * i * gaussian_r_coeff);
    }


    for (i = 0; i < ny; i++)
        for (j = 0; j < nx; j++)
        {
            for (k = 0; k < m_nChannels; k++)
            {
                double weight_sum, pixcel_sum;
                weight_sum = pixcel_sum = 0.0;

                for (m = -r; m <= r; m++)
                    for (n = -r; n <= r; n++)
                    {
                        if (m*m + n*n > r*r) continue;

                        int x_tmp = j + n;
                        int y_tmp = i + m;

                        x_tmp = x_tmp < 0 ? 0 : x_tmp;
                        x_tmp = x_tmp > nx - 1 ? nx - 1 : x_tmp;
                        y_tmp = y_tmp < 0 ? 0 : y_tmp;
                        y_tmp = y_tmp > ny - 1 ? ny - 1 : y_tmp;

                        int pixcel_dif = (int)abs(img_tmp[y_tmp * m_nChannels * nx + m_nChannels * x_tmp + k] - img_tmp[i * m_nChannels * nx + m_nChannels * j + k]);
                        double weight_tmp = d_metrix[m + r][n + r] * r_metrix[pixcel_dif];

                        pixcel_sum += img_tmp[y_tmp * m_nChannels * nx + m_nChannels * x_tmp + k] * weight_tmp;
                        weight_sum += weight_tmp;
                    }

                pixcel_sum = pixcel_sum / weight_sum;
                m_imgData[i * m_nChannels * nx + m_nChannels * j + k] = (uchar)pixcel_sum;

            }

        }

    UpdateImage();
    DeleteDoubleMatrix(d_metrix, w_filter, w_filter);
    delete[] img_tmp;
}
```

性能方面，跟OpenCV处理速度有差距，有兴趣的，可以自己研究OpenCV版本的源代码

三、效果图





四、参考资料

资料[4]是MIT的学习资料，最全面，包括课件、论文、代码等，涵盖原理、改进、应用、与PDE的联系等等，最值得一看。

[2] 【OpenCV】邻域滤波：方框、高斯、中值、双边滤波
[3] Bilateral Filtering(双边滤波) for SSAO
[4] MIT学习资料