《微波遥感》课程集中实习

实习三 极化 SAR 信息提取

实习报告

目录

一、实习目的

利用全极化 SAR 数据进行滤波和极化目标分解，了解极化 SAR 数据的处理、地物的极化散射机理，加深对极化 SAR 遥感原理和应用的学习，巩固课堂相关内容的学习成果。

本程序报告主要囊括四部分：

1.利用 SNAP 将旧金山湾极化 SAR 数据转为 C3 文件；

2.编程实现该数据的 BOXCAR 均值极化滤波；

3.利用 SNAP 对滤波后的结果进行伪彩色合成，并对比滤波前后差异；

（注：将 hdr 和 config 文件拷入滤波后的文件夹中）

4.编程实现利用滤波后数据的 Freeman 三分量极化目标分解；

利用 SNAP 对分解的结果进行伪彩色合成，并利用分解结果对典型地物进行目视解译。（注：将 hdr 和 config 文件拷入分解后的文件夹中，根据分解结果文件名，修改 hdr 文件名）

二、实习原理

2.1 SAR 极化方式

2.1.1 极化方式

极化方式(Polarization）: H 水平极化；V 垂直极化，即电磁场的振动方向，卫星向地面发射信号时，所采用的无线电波的振动方向可以有多种方式，目前所使用的有：水平极化（H）：水平极化是指卫星向地面发射信号时，其无线电波的振动方向是水平方向。垂直极化（V）：垂直极化是指卫星向地面发射信号时，其无线电波的振动方向是垂直方向。使用 H 和 V 线性极化的雷达系统用一对符号表示发射和接收极化，因此可以具有以下通道—HH、VV、HV、VH。

雷达遥感系统常用四种极化方式：

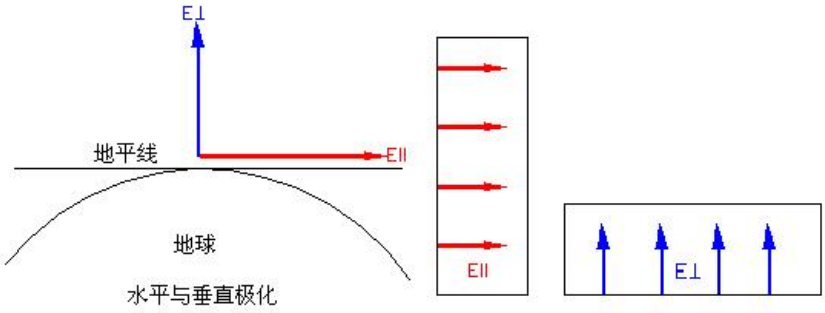
（1）HH-用于水平发送和水平接收

（2）VV-用于垂直发送和垂直接收

（3）HV-用于水平发送和垂直接收

（4）VH-用于垂直发送和水平接收

这些偏振组合中的前两个被称为相似偏振，因为发射和接收偏振是相同的。最后两个组合称为交叉极化，因为发送和接收极化彼此正交。



2.1.2 极化方式的选择

经验表明，对于海洋应用，L 波段的 HH 极化较敏感，而 C 波段是 VV 极化比较好；对于低散射率的草地和道路，水平极化使地物之间有较大的差异，所以，地形测绘用的星载

SAR 都使用水平极化；对粗糙度大于波长的陆地，HH 或 VV 无明显变化。不同极化下同一地物的回波强弱不同，图像的色调也不一样，增加了识别地物目标的信息。相同极化（HH ，VV）和交叉极化（HV ，VH）的信息比较，可以显著地增加雷达图像信息，而且，植被和其他不同地物的极化回波之间的信息差别比不同波段之间的差别更敏感。所以，多极化工作是 SAR 卫星发展方向之一。

所以，实际应用中可根据不同需求选择合适的极化方式，综合利用多种极化方式有利于地物分类精度的提高。

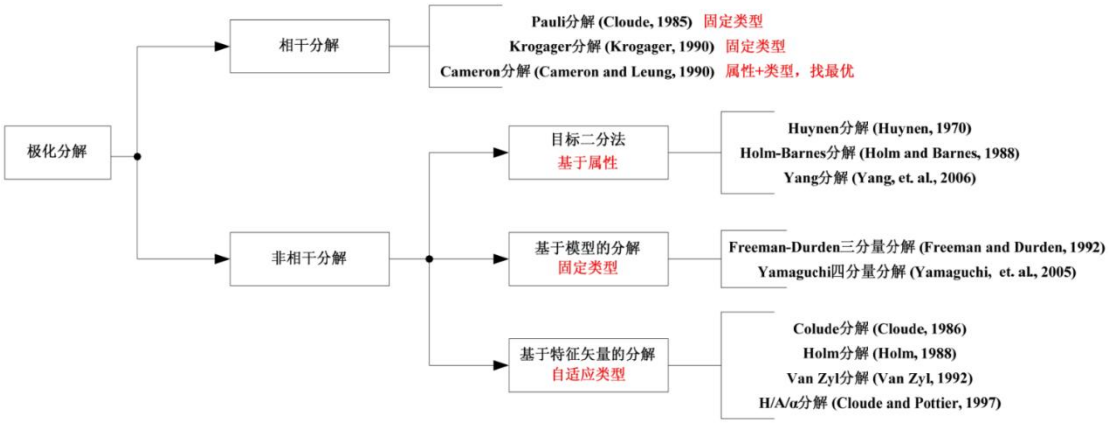
2.2 SAR 极化分解

2.2.1 极化分解概述

极化分解技术是研究和应用全极化 SAR 数据的主要方法，因为极化分解技术本质上分离了地物不同散射机制引|起的极化特征，利用各种极化分解方法就可以有效地提取目标地物的主要散射特征，从而实现提取目标地物信息的目的。其中,极化分解包含了基于散射矩阵的相干目标极化分解和基于散射矩阵的二阶矩(协方差矩阵或相干矩阵)的非相干目标极化分解。极化特征参数是基于极化分解技术得到反映目标散射电磁波特征的参数或在这些极化分解参数基础上组合形成的特定指数。利用这些特征参数可以相对容易地解释电磁波与地物之间的相互作用，简化了定量反演地物参数等有关信息的算法。

2.2.2 极化分解方法

雷达遥感应用中的自然目标大多为分布目标散射体,其属于非相干目标,可采用非相干极化分解方法对散射体进行统计性描述。常用的非相干分解方法有 Freeman-Durden 分解、基于特征矢量和特征值的分解、H/A/alpha 分解及 Van Zyl 分解等。采用这些分解方法,都可以在一定约束条件下得到反映目标散射特性的定量信息。这些分解方法分别从基本电磁散射物理模型、数学正交分解模型及地物散射模型等角度出发,反演出地物散射回波信息中的单次、偶次和随机散射的强度,以及与地物结构密切相关的参数。



2.2.3 boxcar 均值极化滤波

Boxcar 滤波是一种简单的信号处理技术，它通过在信号上滑动一个固定宽度的窗口，并计算窗口内的平均值来平滑信号。Boxcar 滤波器，也被称为滑动平均滤波器，因其形状类似于一个矩形（boxcar）而得名。

原理

Boxcar 滤波器的核心思想是将信号的每个点替换为该点及其周围点的平均值。这种方法可以减少噪声，但同时也会引入一定的延迟，因为输出信号的每个点都是基于输入信号的

多个点计算得到的。

主要步骤

1、确定窗口大小：选择一个固定的窗口大小（N），这个大小决定了滤波器平滑程度和延迟的大小。窗口越大，平滑效果越明显，但延迟也越大。

2、滑动窗口：将窗口从信号的开始位置开始滑动，每次移动一个点。

3、计算平均值：对于窗口覆盖的每个点，计算窗口内所有点的平均值。

4、替换信号点：将窗口中心的原始信号点替换为步骤 3 中计算出的平均值。

5、重复过程：继续滑动窗口，重复步骤 3 和 4，直到覆盖整个影像。

注意事项

● Boxcar 滤波器对信号的边缘点处理较为简单，通常采用复制边缘点的方法来计算边缘的平均值，但这可能会导致边缘效应。

● Boxcar 滤波器对信号的高频成分有较大的衰减，因此它不适合用于需要保留高频信息的场合。

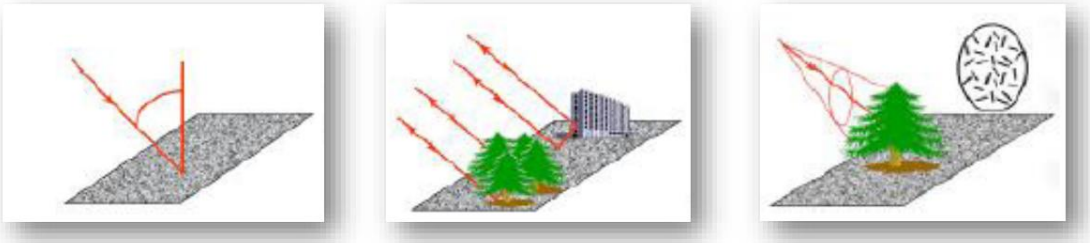
● Boxcar 滤波器是一种线性滤波器，它不改变信号的直流分量。

2.2.4 Freeman-Durden 分解

freeman 分解是 1998 年 Freeman 和 Durden 在 van Zyl 的研究基础上提出的一种非相干矩阵分解方法，其中主要思路是将极化协方差矩阵分解为三种主要的散射机理，即面散射、体散射、二面角散射。其表现形式

Freeman 分解公式为：

$$[C]=\begin{bmatrix} |S_{HH}|^2 & \sqrt{2}S_{HH}S_{HV}^* & S_{HH}S_{VV}^* \\ \sqrt{2}S_{HH}^*S_{HV} & 2|S_{HV}|^2 & \sqrt{2}S_{HV}S_{VV}^* \\ S_{HH}^*S_{VV} & \sqrt{2}S_{HV}^*S_{VV} & |S_{VV}|^2 \end{bmatrix}=\begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{12}^* & C_{22} & C_{23} \\ C_{13}^* & C_{23}^* & C_{33} \end{bmatrix}$$



(a) 奇次散射机制 Cs　　(b) 偶次散射机制 Cd　　(c) 体散射机制 Cv

所以，Freeman 分解将测量到的协方差矩阵 $[C_3]$ 与总的后向散射模型表示如下：

$$[C_3]=\langle[C_3]\rangle_v+\langle[C_3]\rangle_d+\langle[C_3]\rangle_s$$

$$\begin{cases} \langle|S_{HH}|^2\rangle=f_s|\beta|^2+f_d|\alpha|^2+f_v \\ \langle|S_{VV}|^2\rangle=f_s+f_d+f_v \\ \langle S_{HH}S_{VV^*}\rangle=f_s\beta+f_d\alpha+f_v/3 \\ \langle|S_{HV}|^2\rangle=f_v/3 \end{cases}$$

$$C_s = f_s \begin{pmatrix} |\beta|^2 & 0 & \beta \\ 0 & 0 & 0 \\ \beta^* & 0 & 1 \end{pmatrix} \quad C_d = f_d \begin{pmatrix} |\alpha|^2 & 0 & \alpha \\ 0 & 0 & 0 \\ \alpha^* & 0 & 1 \end{pmatrix} \quad C_v = f_v \begin{pmatrix} 1 & 0 & 1/3 \\ 0 & 2/3 & 0 \\ 1/3 & 0 & 1 \end{pmatrix}$$

其中，f_v，f_s，f_d分别对应着体散射、面角散射和二面角散射分量，对应着的散射功率表示如下：

$$\text{面散射功率} \quad P_s = f_s * (1 + \beta^2)$$
$$\text{二面角散射功率} \quad P_d = f_d * (1 + \alpha^2)$$
$$\text{体散射功率} \quad P_v = 8 * f_v/3$$

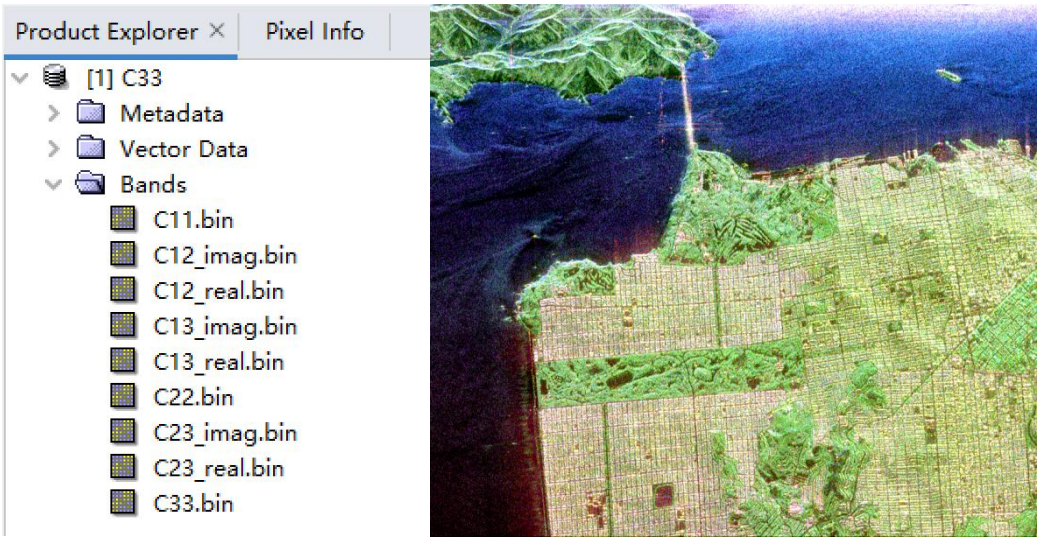
与其他一些分解算法一样，Freeman-Durden 分解的各参量也可以用来形成一幅彩色的 RGB 图像来表示极化信息。在这种情况下，考虑分解的各分量散射能量，有：

$$P_s \rightarrow Blue$$
$$P_d \rightarrow Red$$
$$P_v \rightarrow Green$$

三、实习数据与环境

3.1 旧金山湾极化 SAR 数据 C3 文件

旧金山湾地区 AIRSAR 全极化数据，已转换为 C3 格式，其数据组织格式及 C11、 C22、C33 通道 RGB 合成结果如下图：



C 矩阵与各极化通道关系如下：

$$[C] = \begin{bmatrix} |S_{HH}|^2 & \sqrt{2}S_{HH}S_{HV}^* & S_{HH}S_{VV}^* \\ \sqrt{2}S_{HH}^*S_{HV} & 2|S_{HV}|^2 & \sqrt{2}S_{HV}S_{VV}^* \\ S_{HH}^*S_{VV} & \sqrt{2}S_{HV}^* & |S_{VV}|^2 \end{bmatrix}$$

3.2 Eigen 库

实现 H-A-α 分解程序的编写，需要进行矩阵的操作及特征值分解等，为了方便需要用到 Eigen 库。

Eigen 是一个高层次的 C ++库，有效支持线性代数，矩阵和矢量运算，数值分析及其相关的算法。Eigen 目前最新的版本是 3.4，除了 C++标准库以外，不需要任何其他的依赖包。Eigen 使用的 CMake 建立配置文件和单元测试，并自动安装。如果使用 Eigen 库，只需包特定模块的的头文件即可。Eigen 适用范围广，支持包括固定大小、任意大小的所有矩阵操作，甚至是稀疏矩阵；支持所有标准的数值类型，并且可以扩展为自定义的数值类型；支持多种矩阵分解及其几何特征的求解；它不支持的模块生态系统提供了许多专门的功能，如非线性优化，矩阵功能，多项式解算器，快速傅立叶变换等。

3.3 SNAP 平台

ESA SNAP (Sentinel Application Platform) 哨兵数据应用平台，是欧空局针对哥白尼计划中哨兵数据任务开发的免费开源的软件。SNAP 软件中集成了多种算法的实现。

四、实习内容

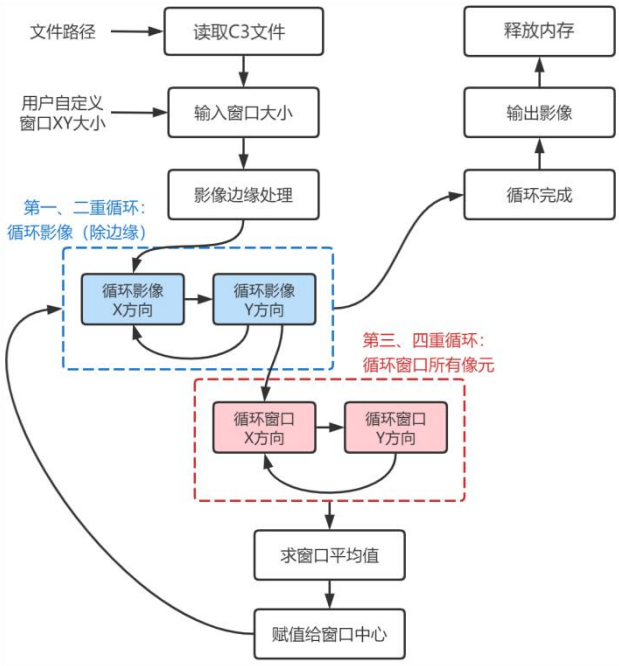
4.1 C3 格式二进制文件读取

C3 文件的存储由 9 个同样大小的二进制文件组成，分别存储 C 矩阵的上三角矩阵的所有实部和虚部，因此对于 C3 文件的读取应使用 fread 函数三次循环（文件、行、列）进行读取。

```
/* 输入/输出文件指针数组  */
FILE* in_file[16], * out_file_boxcar[16], * out_file_FD[16];

/* Strings */
char file_name[1024], in_dir[1024], out_dir_bx[1024], out_dir_FD[1024];
const char* file_name_in_out[16] =
{ "C11.bin", "C12_real.bin", "C12_imag.bin",
"C13_real.bin", "C13_imag.bin", "C22.bin",
"C23_real.bin", "C23_imag.bin", "C33.bin" };
char PolarCase[20], PolarType[20];

/* Input variables */
int Nlig = 900, Ncol = 1024;/* Initial image nb of lines and rows */
int Off_lig, Off_col; /* Lines and rows offset values */
int Sub_Nlig, Sub_Ncol;  /* Sub-image nb of lines and rows */
int Nwin = 5;                /* Filter window width */
```

```
int Npolar = 9;

// 文件路径
strcpy(in_dir, "D:\\gdal\\Freeman\\");

float*** M_in = new float** [Npolar];

for (int i = 0; i < Npolar; i++)
{
    M_in[i] = new float* [Nlig];
    for (int j = 0; j < Nlig; j++)
    {
        M_in[i][j] = new float[Ncol];
    }
}

float*** M_out = new float** [Npolar];
for (int i = 0; i < Npolar; i++)
{
    M_out[i] = new float* [Nlig];
    for (int j = 0; j < Nlig; j++)
    {
        M_out[i][j] = new float[Ncol];
    }
}

//设置输入输出文件路径及文件名
for (int Np = 0; Np < Npolar; Np++)
{
    sprintf(file_name, "%s%s", in_dir, file_name_in_out[Np]);
    //printf("file_name: %s\n", file_name);
    if ((in_file[Np] = fopen(file_name, "rb")) == NULL)
        printf_s("Could not open input file :%s\n ", file_name);

    sprintf(file_name, "%s%s", out_dir_bx, file_name_in_out[Np]);
    if ((out_file_boxcar[Np] = fopen(file_name, "wb")) == NULL)
        printf_s("Could not open output file : %s\n", file_name);
}

//M_in 读入 C33 原始数据
for (int i = 0; i < Npolar; i++)
    for (int j = 0; j < Nlig; j++)
        for (int k = 0; k < Ncol; k++)
            fread(&M_in[i][j][k], sizeof(float), 1, in_file[i]);
```

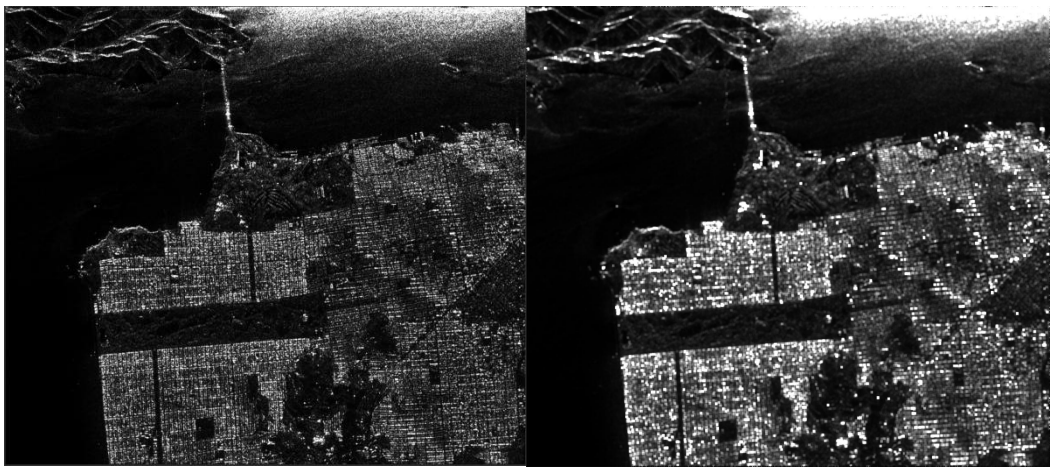
4.2 boxcar 均值滤波算法




```
//设置 5*5 滤波窗口
int xOff = (Nwin - 1) / 2;
int yOff = (Nwin - 1) / 2;
float* win = new float[Nwin * Nwin];

// 影像边缘直接复制原数据
for (int i = 0; i < Npolar; i++)
    for (int j = 0; j < Nlig; j++)
        for (int k = 0; k < Ncol; k++)
        {
            if (j < yOff || k < xOff || j > Nlig - 1 - yOff || k > Ncol - 1 - xOff)
                M_out[i][j][k] = M_in[i][j][k];
        }

//boxcar 均值滤波
for (int i = 0; i < Npolar; i++)
{
    for (int j = yOff; j <= Nlig - 1 - yOff; j++)
    {
        for (int k = xOff; k <= Ncol - 1 - xOff; k++)
        {
            // 循环窗口内每个像元
            int count = 0; float sum = 0;
            for (int x = -xOff; x <= xOff; x++)
            {
                for (int y = -yOff; y <= yOff; y++)
```

```
                    {
                        win[count] = M_in[i][j + x][k + y]; // 取出窗口内每个像元
                        sum += win[count]; // 求和
                        count++;
                    }
                }
                // 灰度赋值
                M_out[i][j][k] = (sum / count); // 赋平均值给窗口中心
            }
        }
    }

    // 输出滤波结果
    for (int i = 0; i < Npolar; i++)
        for (int j = 0; j < Nlig; j++)
            for (int k = 0; k < Ncol; k++)
            {
                //逐个文件写入 float 数据
                fwrite(&M_out[i][j][k], sizeof(float), 1, out_file_boxcar[i]);
            }
```

C3 文件滤波前后效果对比：



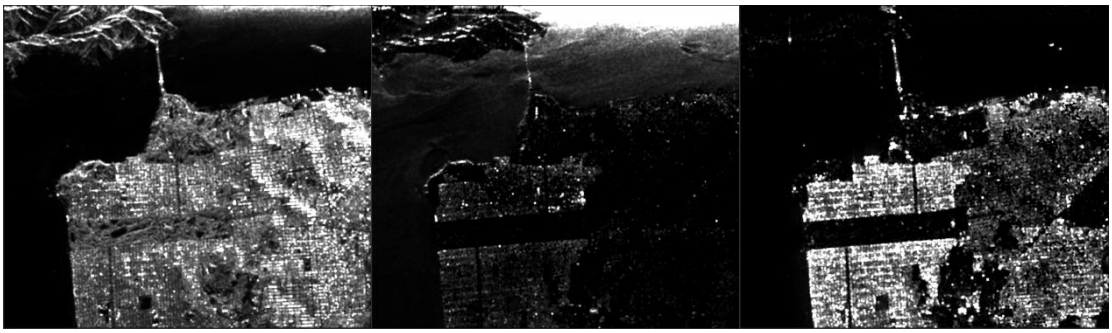
boxcar 均值滤波后进行假彩色合成：

4.3 Freeman-Durden 分解

```
float*** M_in_FD = new float** [Npolar];

for (int i = 0; i < Npolar; i++)
{
    M_in_FD[i] = new float* [Nlig];
    for (int j = 0; j < Nlig; j++)
    {
        M_in_FD[i][j] = new float[Ncol];
    }
}

float*** M_out_FD = new float** [Npolar];
for (int i = 0; i < Npolar; i++)
{
    M_out_FD[i] = new float* [Nlig];
    for (int j = 0; j < Nlig; j++)
    {
        M_out_FD[i][j] = new float[Ncol];
    }
}

for (int Np = 0; Np < Npolar_out; Np++) {
    sprintf(file_name, "%s%s", out_dir_FD, file_name_in_out[Np]);
    if ((out_file_FD[Np] = fopen(file_name, "wb")) == NULL)
        printf_s("Could not open output file :%s\n ", file_name);
}

float FS, FD, FV, ALP, BET, CC11, CC13_re, CC13_im, CC22, CC33;
float rtemp;
```

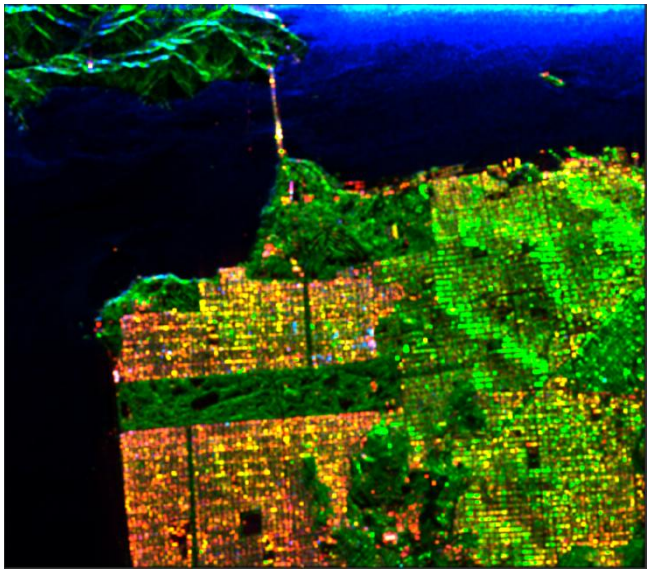
```
float eps = 1e-8;
FS = FD = FV = ALP = BET = 0;
CC11 = CC13_re = CC13_im = CC22 = CC33 = 0;

for (int j = 0; j < Nlig; j++)
{
    for (int k = 0; k < Ncol; k++)
    {
        CC11 = M_out[C11][j][k];
        CC13_re = M_out[C13_re][j][k];
        CC13_im = M_out[C13_im][j][k];
        CC22 = M_out[C22][j][k];
        CC33 = M_out[C33][j][k];
        FV = 3. *CC22 / 2.;
        CC11 = CC11 - FV;
        CC33 = CC33 - FV;
        CC13_re = CC13_re - FV / 3.;
        /*Case 1: Volume Scatter > Total*/
        if ((CC11 <= eps) || (CC33 <= eps))
        {
            FV = 3. * (CC11 + CC22 + CC33 + 2 * FV) / 8.;
            FD = 0.;
            FS = 0.;
        }
        else
        {
            /*Data conditionning for non realizable ShhSvv* term*/
            if ((CC13_re * CC13_re + CC13_im * CC13_im) > CC11 * CC33) {
                rtemp = CC13_re * CC13_re + CC13_im * CC13_im;
                CC13_re = CC13_re * sqrt(CC11 * CC33 / rtemp);
                CC13_im = CC13_im * sqrt(CC11 * CC33 / rtemp);
            }
            /*Odd Bounce*/
            if (CC13_re >= 0.) {
                ALP = -1.;
                FD = (CC11 * CC33 - CC13_re * CC13_re - CC13_im * CC13_im) / (CC11 + CC33 + 2
* CC13_re);
                FS = CC33 - FD;
                BET = sqrt((FD + CC13_re) * (FD + CC13_re) + CC13_im * CC13_im) / FS;
            }
            /*Even Bounce*/
            if (CC13_re < 0.) {
                BET = 1.;
                FS = (CC11 * CC33 - CC13_re * CC13_re - CC13_im * CC13_im) / (CC11 + CC33 - 2
```

```
* CC13_re);
                        FD = CC33 - FS;
                        ALP = sqrt((FS - CC13_re) * (FS - CC13_re) + CC13_im * CC13_im) / FD;
                    }
                }

                M_out_FD[ODD][j][k] = FS * (1 + BET * BET);
                M_out_FD[DBL][j][k] = FD * (1 + ALP * ALP);
                M_out_FD[VOL][j][k] = 8. * FV / 3.;

                if (M_out_FD[ODD][j][k] < 0) M_out_FD[ODD][j][k] = 0;
                if (M_out_FD[DBL][j][k] < 0) M_out_FD[DBL][j][k] = 0;
                if (M_out_FD[VOL][j][k] < 0) M_out_FD[VOL][j][k] = 0;
            }
        }

        //输出 FD 分解后的影像
        for (int i = 0; i < Npolar_out; i++)
            for (int j = 0; j < Nlig; j++)
                for (int k = 0; k < Ncol; k++)
                    fwrite(&M_out_FD[i][j][k], sizeof(float), 1, out_file_FD[i]);

        //释放内存
        delete[]M_in;
        delete[]M_out;
        delete[]M_in_FD;
        delete[]M_out_FD;
```



VOL 分量　　ODD 分量　　DBL 分量

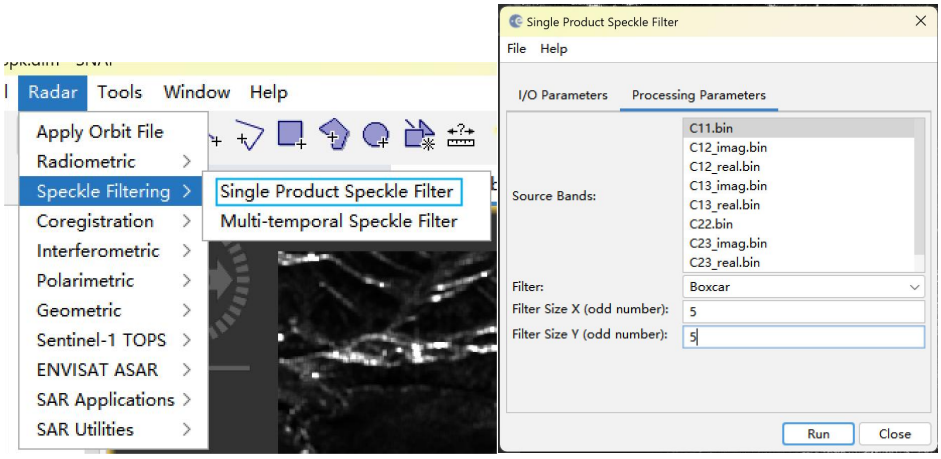
Freeman-Durden 分解后根据标准进行三通道的 RGB 假彩色合成：



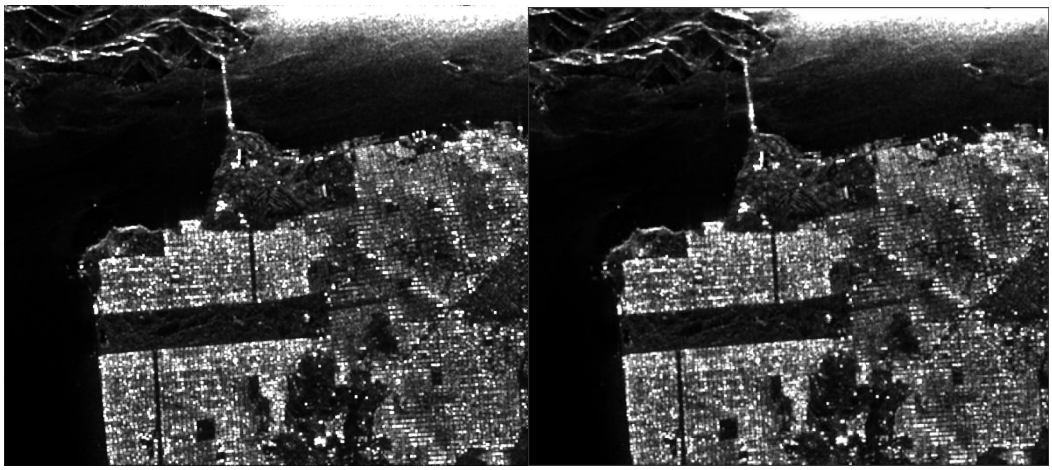
五、滤波结果与分解结果分析

5.1 程序与 SNAP 运行结果对比

SNAP 中进行 boxcar 滤波需要依次点击“Rader-Speckle Filtering-Single Product Speckle Filter”，在“processing parameters”中的“filter”选择“boxcar”，“Filter Size X”和“Filter Size Y”均为 5，然后点击 run。

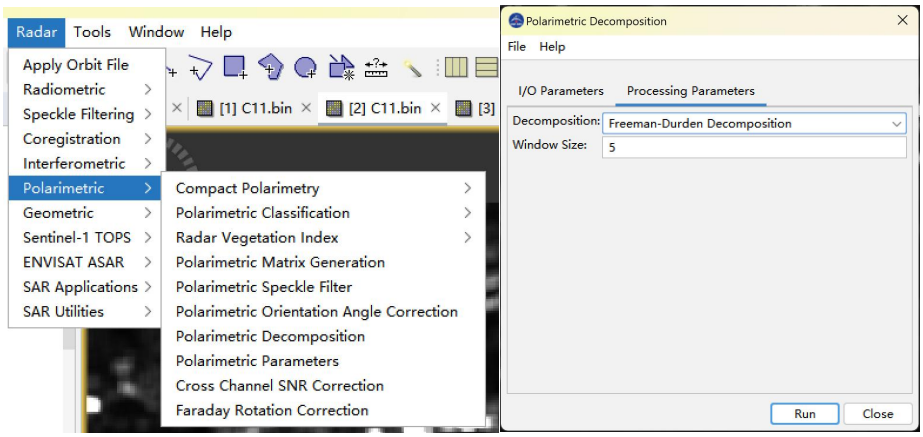


C++程序与 SNAP 滤波结果对比如下（以 C11.bin 示例），肉眼查看无明显差异，查看像元详细信息后也相同。



C++程序滤波结果　　SNAP 滤波结果

SNAP 中进行 Freeman-Durden 分解需要依次点击“Rader-Polarimetric-Polarimetric Decomposition”，在“processing parameters”中的“Decomposition”选择“Freeman-Durden Decomposition”，“Window Size X”为 5，然后点击 run。



C++程序与 SNAP 滤波结果对比如下（以 DBL.bin 示例），肉眼查看无明显差异，查看像元详细信息后细节处有细微不同。

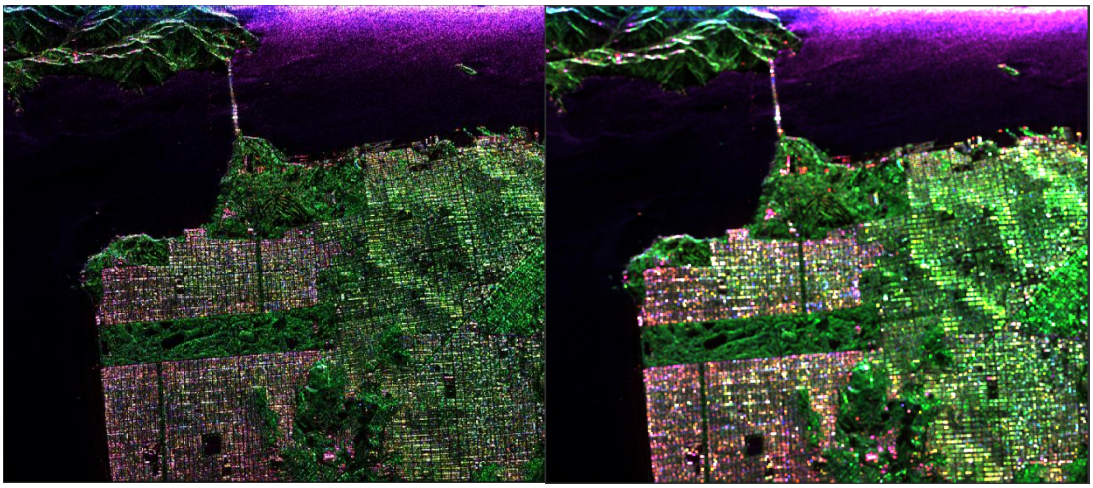
C++程序滤波结果　　SNAP 滤波结果

5.2 滤波结果分析

Boxcar 均值极化滤波是一种简单的图像处理技术，用于减少图像中的噪声。

原图由于噪声的存在，图像中的细节可能会被掩盖，使得小的地物特征难以辨认。同时噪声会降低图像的局部对比度，使得不同地物之间的界限变得模糊。

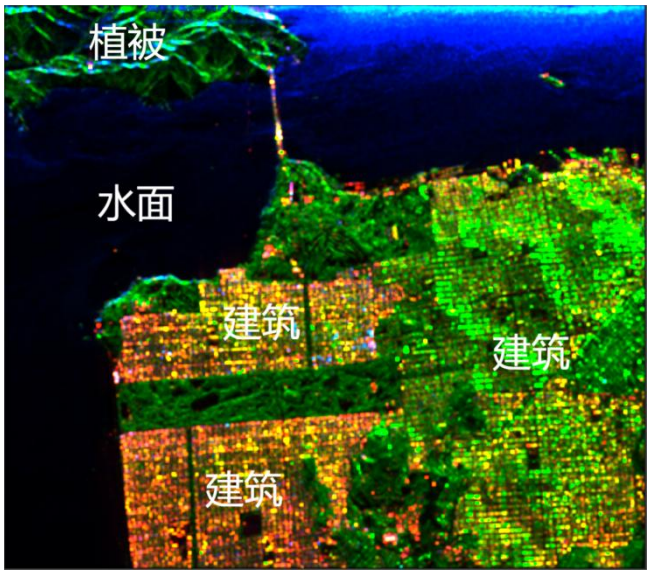
滤波后影像与原图相比，滤波后的结果斑点噪声有减少，Boxcar 滤波通过计算局部区域的平均值，有效减少了散斑噪声，使图像更加平滑。但均值滤波也使得滤波后的结果相较于原图失去了很多边缘和细节，本质上是因为其不适应斑点噪声乘性噪声的性质。与此同时放大可以发现原本同一地物的各部分显示出多种极化特征，但在均值滤波后这种情况被削弱，同一地物显得更同质化。由于 Boxcar 滤波是一种低通滤波器，它可能会使图像的边缘稍微模糊，但这可以通过选择合适的窗口大小来控制。减少噪声后，图像分类、目标检测等分析任务的准确性可能会提高。总而言之，Boxcar 均值极化滤波是一种简单且有效的降噪方法，可以显著改善 SAR 影像的质量，提高后续分析的准确性。但 Boxcar 滤波可能会引入一些模糊，特别是对于边缘和高频细节，这可能会影响某些特定分析任务的性能。



原图　　Boxcar 滤波结果

5.3 Freeman 分解地物目视解译

据以下 RGB 合成标准可知，蓝色主要反映奇次散射，绿色主要反映体散射，红色主要反映偶次散射。



1. 海洋地物较为容易解译，因水体散射特性决定其主要发生奇次散射，呈现为蓝黑色连续区域（黑色表明回波强度较低）；

2. 植被覆盖的山地主要通过山地纹理和植被体散射所呈现的绿色进行解译；

3. 部分建筑区可通过形成二面角产生偶次散射所呈现红色和矩形形状进行解译；

4. 另一部分建筑区可通过形状为矩形，以及排列整齐的纹理判断，其显示出绿色的主要原因是与其方向与电磁波入射并不垂直，使得其产生回波极化模式发生改变的情况。

六、实习心得

通过本次《微波遥感》课程的集中实习，我深刻体会到了理论知识与实践技能相结合的重要性。实习的主要目的是利用全极化 SAR 数据进行滤波和 Freeman-Durden 极化目标分解，这不仅加深了我对极化 SAR 遥感原理的理解，也锻炼了我的数据处理和编程能力。

在实习过程中，根据老师提供的示例代码，我学习了如何使用 SNAP 软件将旧金山湾的极化 SAR 数据转换为 C3 文件，这一步骤为我后续的数据处理打下了基础。然后我编程实现了 BOXCAR 均值极化滤波算法，通过编写代码对数据进行平滑处理，有效降低了图像噪声，提高了图像质量。最后编写了 Freeman-Durden 极化分解的程序，并将其应用于极化 SAR 数据，成功提取了地物的散射特性。

在实习中，我遇到了不少挑战，尤其是在编程实现滤波算法和极化分解时，需要不断调试和优化代码以获得最佳结果。这些经历教会了我在面对问题时如何保持耐心，并通过查阅资料和反复实践来寻找解决方案。

通过本次实习，我不仅掌握了 SAR 数据处理的关键技术，还学会了如何使用专业软件进行图像分析和解译。这些技能对我未来的学术研究和职业发展都具有重要意义。我深刻认识到，理论知识的学习需要通过实践来巩固和深化，而实际操作中遇到的问题和挑战则是提升自己解决问题能力的宝贵机会。在未来，我希望能够将实习中学到的知识和技能应用到更广泛的遥感领域中，不断探索和学习新的数据处理方法，为遥感科学的发展贡献自己的力量。