模式识别
遥感影像 KL 变换
实习报告

（遥感信息工程学院 2012 级）

班　　级：1201

姓　　名：陈卉君

学　　号：2012302590011

指导教师：马洪超

一、 算法原理与步骤

KL 变换是建立在统计特性基础上的一种具有去相关性能的线性可逆变换，也称为主成分分析或主分量分析。它是从图像统计特性出发,用一组不相关的系数来表示连续信号，实现正交变换，因而在均方误差准则下，它是失真最小的一种变换，故称作最佳变换。与其他正交变换相比，它的能量最集中，误差最小。变换之后数值较大的方差仅存于少数系数中，这样就有可能在允许的失真度下，把图像数据压缩到最小其定义为：以矢量信号X的协方差矩阵Φ的归一化正交特征矢量q所构成的正交矩阵Q，来对该矢量信号X做正交变换$Y = QX$，则称此变换为 KL 变换。可见，要实现 KL 变换，首先要从信号求出其协方差矩阵Φ，再由Φ求出正交矩阵Q。

KL 变换的有如下几点特性：

(1) 去相关特性。K L 变换是变换后的矢量信号的分量互不相关。

(2) 能量集中性。所谓能量集中性，是指对 N 维矢量信号进行 KL 变换后，最大的方差集中在前 M 个低次分量之中(M < N)。

(3) 最佳特性。KL 变换是在均方误差测度下，失真最小的一种变换。由于这一特性，KL 变换被称为最佳变换。许多其他变换都将 KL 变换作为性能上比较的参考标准。

(4) 无快速算法，且变换矩阵随不同的信号样值集合而不同。这是 KL 变换的一个缺点，是变换实际应用中的一个巨大障碍。

KL 变换的具体实现过程为：

设给定的随机矢量$X = [x_1, x_2, \dots, x_N]^T$，现在我们的目的是寻找$x_N$的线性函数$L^T x_j$($L$为正交矩阵，即$L^T L = I$)，使得$L^T x_j$的方差尽可能的大，即最大限度地解释变量变化的原因。因此在数学上就是求L，使得$L^T X$的协方差矩阵最大，其协方差矩阵为$cov(L^T X)$，有

$$cov(L^T X) = E\big(L^T X - E(L^T X)\big)\big(L^T X - E(L^T X)\big)^T = L^T \sigma L$$

容易知道，$\sigma = E\big(X - E(X)\big)\big(X - E(X)\big)^T$是对称正定的。假设它的特征根为$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_m \geq 0$，相应的特征向量记为$v_1, v_2, \dots, v_m$，令$V = (v_1, v_2, \dots, v_m)$，则$V$是一个正交阵，即$VV^T = V^T V = I$，则

$$\sigma = V \begin{bmatrix} \lambda_1 & & 0 \\ & \lambda_2 & \\ & \ddots & \\ 0 & & \lambda_m \end{bmatrix} V^T = \sum_{i=1}^{m} \lambda_i v_i {v_i}^T$$

所以，

$$L^T \sigma L = \sum_{i=1}^{m} \lambda_i L^T v_i {v_i}^T L = \sum_{i=1}^{m} \lambda_i \, (L^T v_i)^2$$

$$\leq \sum_{i=1}^{m} \lambda_1 (L^T v_i)^2 = \lambda_1 L^T V V^T L = \lambda_1 L^T L = \lambda_1$$

当$L = v_1$时，有：

$$\mathrm{cov}({v_1}^T \mathrm{X}) = {v_1}^T \sigma v_1 = {v_1}^T \sum_{i=1}^{m} \lambda_i v_i {v_i}^T v_1 = \lambda_1 ({v_1}^T v_1)^2 = \lambda_1$$

同理，

$$\mathrm{cov}({v_i}^T \mathrm{X}) = {v_i}^T \sigma v_i = {v_i}^T \sum_{j=1}^{m} \lambda_j v_j {v_j}^T v_i = \lambda_i ({v_i}^T v_i)^2 = \lambda_i$$

只要$i \neq j$

$$cov({v_i}^T X, {u_j}^T X) = {v_i}^T \sigma v_j = 0$$

由上可知，如果把${y_1} = {v_1}^T X, {y_2} = {v_2}^T X, \dots, {y_m} = {v_m}^T X$看成新的随机向量，则$y_1, y_2, \dots, y_m$互相独立，且有$cov(y_i) = \lambda_i, i = 1,2, \dots, m$。写成矩阵形式，则有：

$$Y = V^T X$$

这就是通常所说的 KL 变换。

二、 代码实现

```
void CBayesView::OnKl()
{
    //线性存储
    CBayesDoc *pDoc=GetDocument();
    DWORD lHeight = pDoc->lHeight;
    DWORD lWidth = pDoc->lWidth;
    DWORD bandnum = pDoc->nbands;

    BYTE** data = (BYTE**)malloc(sizeof(BYTE*)*bandnum);
    double *Result = (double*)malloc(sizeof(double)
        *bandnum*lHeight*lWidth);

    for(DWORD i=0;i<bandnum;i++)
        data[i] = (BYTE*)malloc(sizeof(BYTE)*lHeight*lWidth);

    for (DWORD j=0;j<lHeight;j++)
    {
        for (DWORD i=0;i<lWidth;i++)
        {
            //读取每个像素点的每个波段DN值
            for(DWORD k=0;k<bandnum;k++)
                data[k][i+j*lWidth] = *(pDoc->pData+i+
                j*lWidth+k*lWidth*lHeight);
        }
```

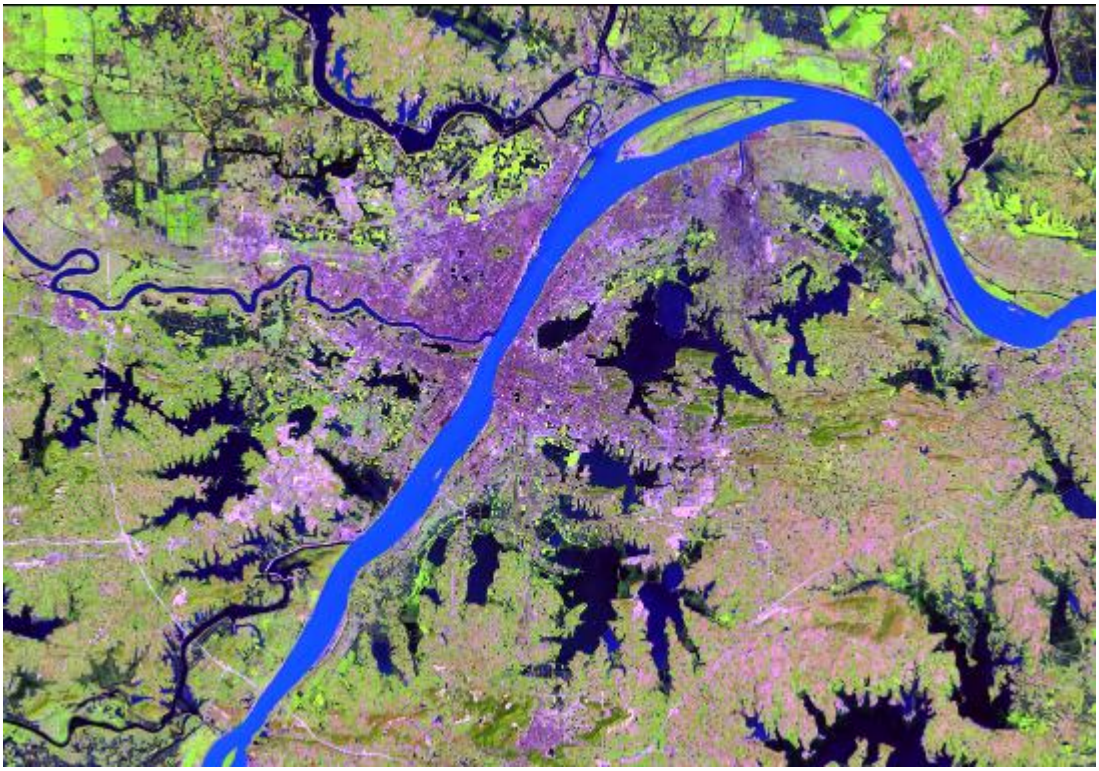
```
}

//计算均值、方差、特征值、特征向量

double *avg = (double*)malloc(sizeof(double)*bandnum);
//计算各波段均值
for(DWORD k=0;k<bandnum;k++)
{
    avg[k] = 0;
    for (i=0;i<lHeight*lWidth;i++)
    {
        avg[k] += (double)data[k][i];
    }
    avg[k] /= (double)(lHeight*lWidth);
}
//计算波段之间的协方差矩阵
double **cov = create_array(bandnum,bandnum);

for(i=0;i<bandnum;i++)
{
    for(DWORD j=0;j<bandnum;j++)
    {
        cov[i][j] = 0.0;

        double a = 0,b = 0;
        for (DWORD k=0;k<lHeight*lWidth;k++)
        {
            a += data[i][k]*data[j][k];
        }
        a /= (lWidth*lHeight);
        b = avg[i]*avg[j];
        cov[i][j] = a - b;
    }
}
//计算协方差矩阵的特征值

 int maxI=60;
 double **q,*b,*c;
 b = (double*)malloc(sizeof(double)*bandnum);
 c = (double*)malloc(sizeof(double)*bandnum);
 q = create_array(bandnum,bandnum);
const double eps=0.000001;
strq(cov,bandnum,q,b,c);
sstq(bandnum,b,c,q,eps,maxI);

//特征值降序排序
for (j=0;j<bandnum;j++)
{
    for (int i=0;i<bandnum-j;i++)
```

```
        {
            if (b[i]<b[i+1])
            {
                double t;
                t=b[i];b[i]=b[i+1];b[i+1]=t;
                liehuhuan(q,bandnum,bandnum,i,i+1);
            }
        }
    }

    //进行变换
    double *t=(double*)malloc(sizeof(double)*bandnum);
    for (j=0;j<lHeight;j++)
    {
        for (DWORD i=0;i<lWidth;i++)
        {
            for(DWORD k=0;k<bandnum;k++)
            {
                t[k]=*(pDoc->pData+i+j*lWidth+k*lWidth*lHeight)-avg[k];
                *(Result+i+j*lWidth+k*lWidth*lHeight) = 0.0;
            }

            for(k=0;k<bandnum;k++)
            {
            {
                for(DWORD num=0;num<bandnum;num++)
                {
                    *(Result+i+j*lWidth+k*lWidth*lHeight) +=
                        q[num][k]*t[num];
                }
            }
        }
    }
    delete[] t;

    //显示图像
    for (j=0;j<lHeight;j++)
    {
        for (DWORD i=0;i<lWidth;i++)
        {
            for(DWORD k=0;k<bandnum;k++)
            {
                *(pDoc->pData+i+j*lWidth+k*lWidth*lHeight) =
                    *(Result+i+j*lWidth+k*lWidth*lHeight);
            }
        }
    }

    //输出图像
```

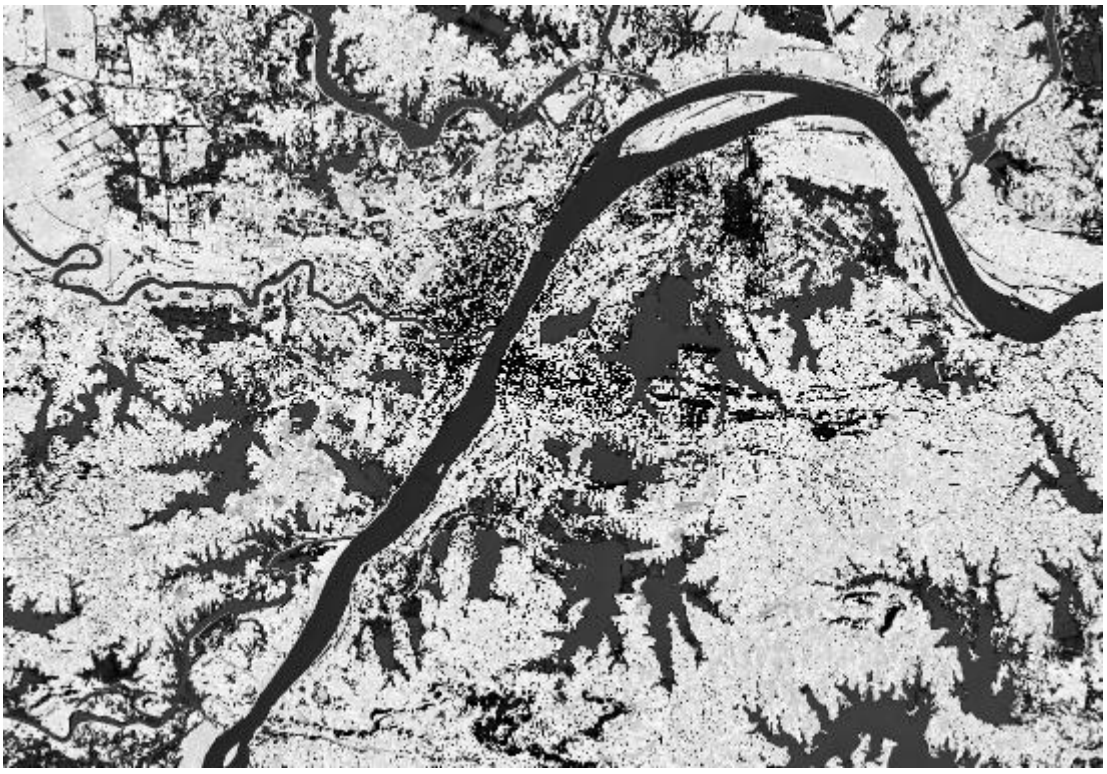
三、 结果分析

用于分类的影像是在地理空间数据云上下载的武汉市的 TM 影像，使用 ERDAS

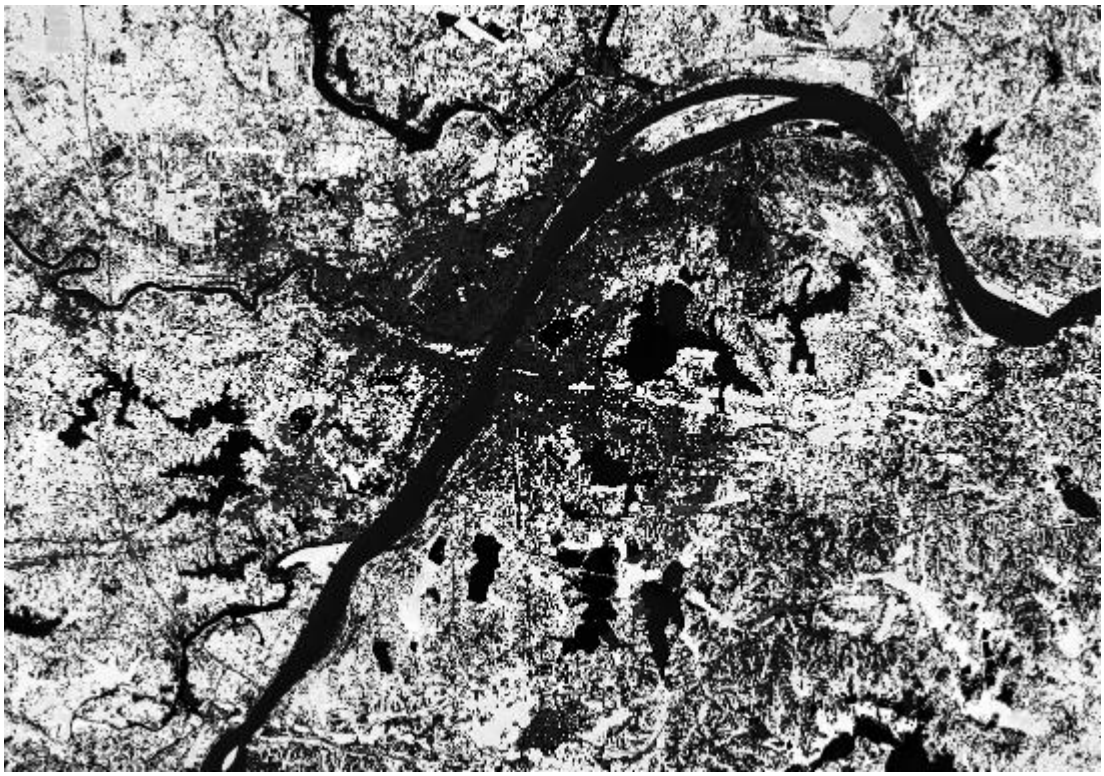
软件将 TM 的 1 到 5 波段以及第 7 波段进行合成，合成后影像格式为 img，下图为 543 波段作为 RGB 显示的影像。



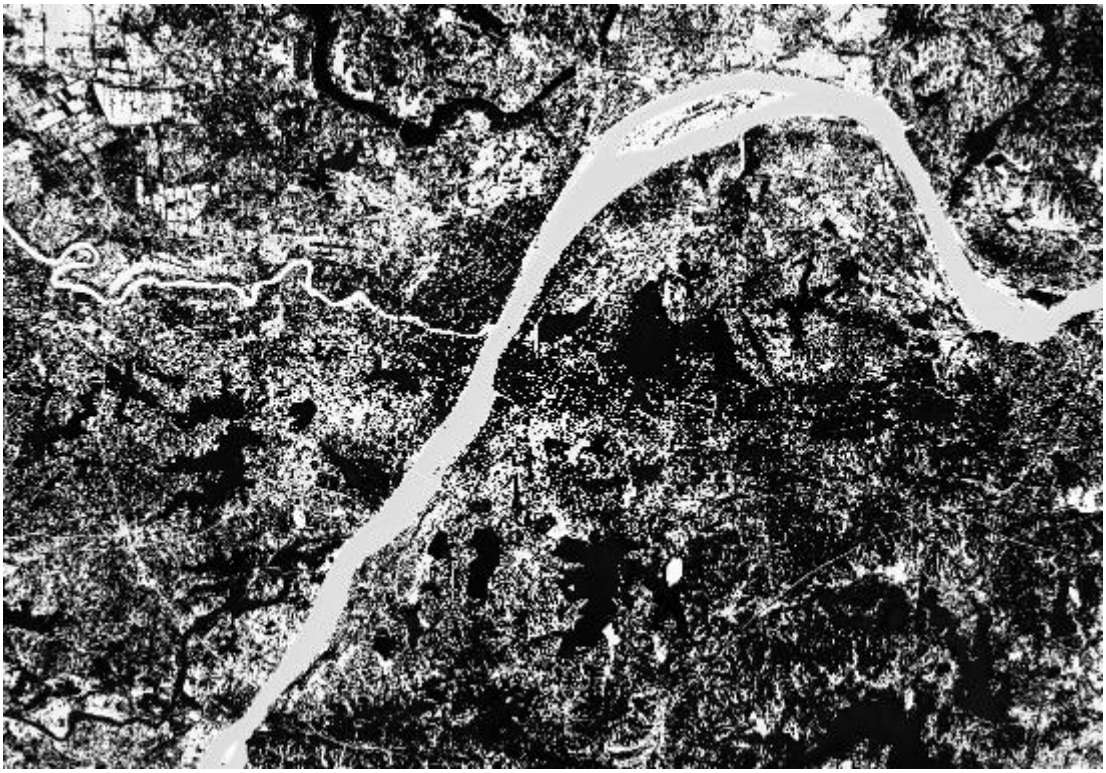
6 个分量图如下，下面注明了熵和特征值



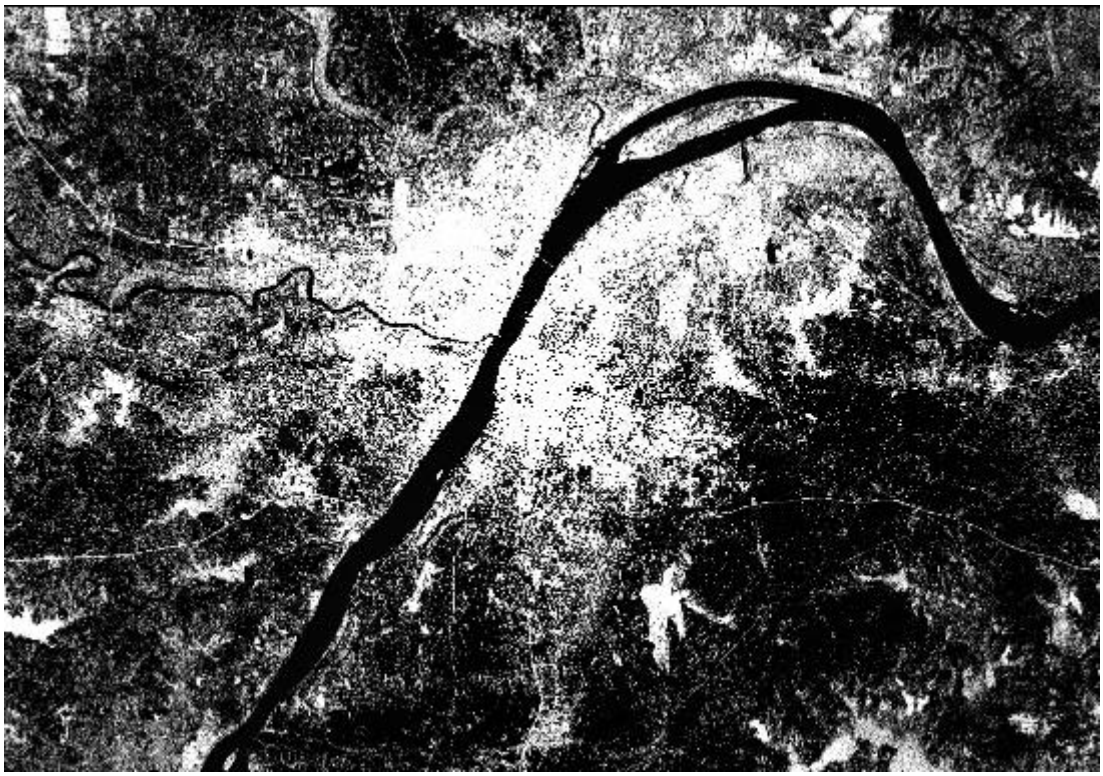
第 1 主分量 熵=6.8798 特征值= 1431.59



第 2 主分量 熵=5.9789 特征值= 262.70

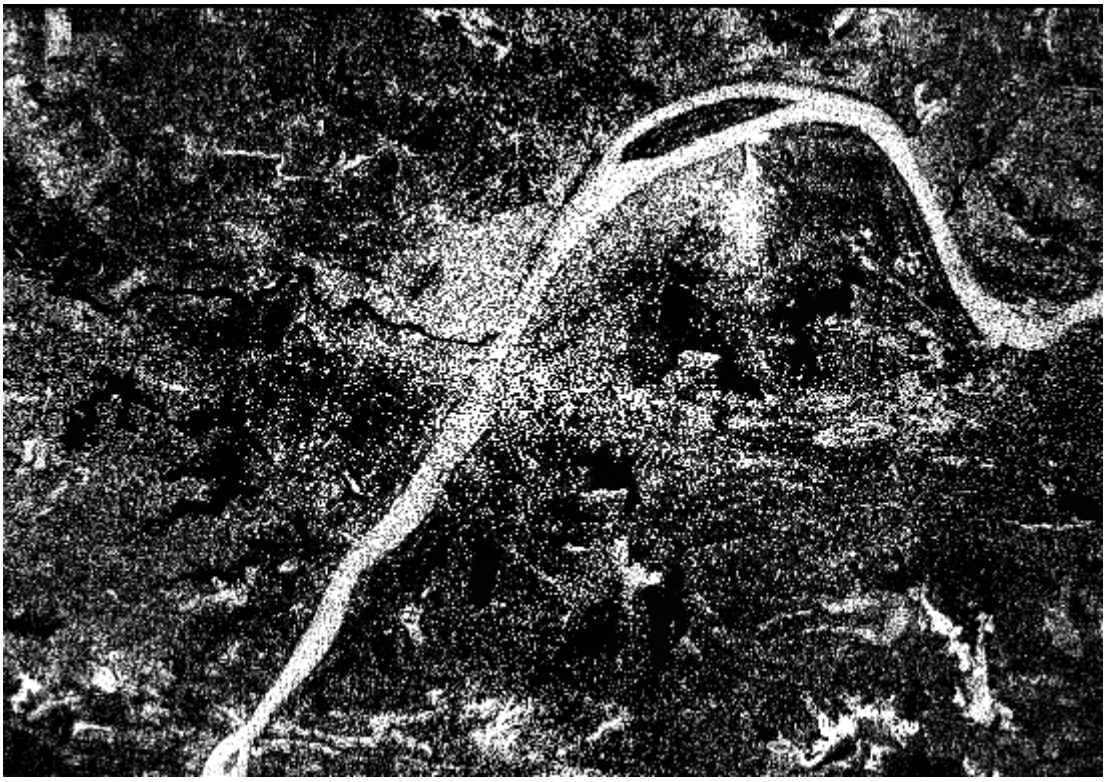


第 3 主分量 熵=4.9668 特征值= 99.97



第 4 主分量 熵=3.1612 特征值= 8.41

第 5 主分量 熵=2.9361 特征值= 5.93



第 6 主分量 熵=1.5937 特征值= 1.13

可以看出，熵是递减的，说明信息量第 1 主分量最多，第 6 主分量少；而特征值也是递减的，而且第 1 主分量特征值远大于其他特征值，最后 3 个主分量特征值则非常小，可知用 K-L 变换来降维可以使得损失的信息比较少。

四、心得体会

1. 对算法的理解是写好程序的基础和关键，对算法理解透了才能更有效率地设计程序
2. 需要了解一下遥感图像处理中常用的开源库，如 GDAL
3. 对于误操作要有相应处理方法和警告。
4. 把程序分成几个功能块，写一步测试一步调试效率较高