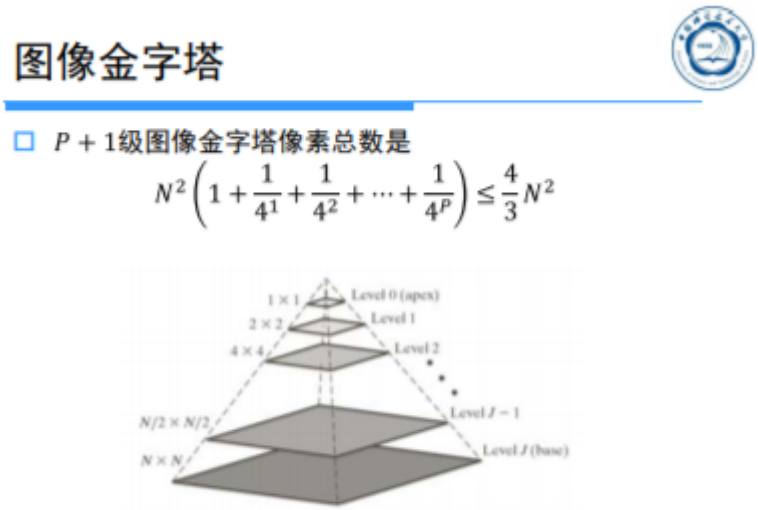
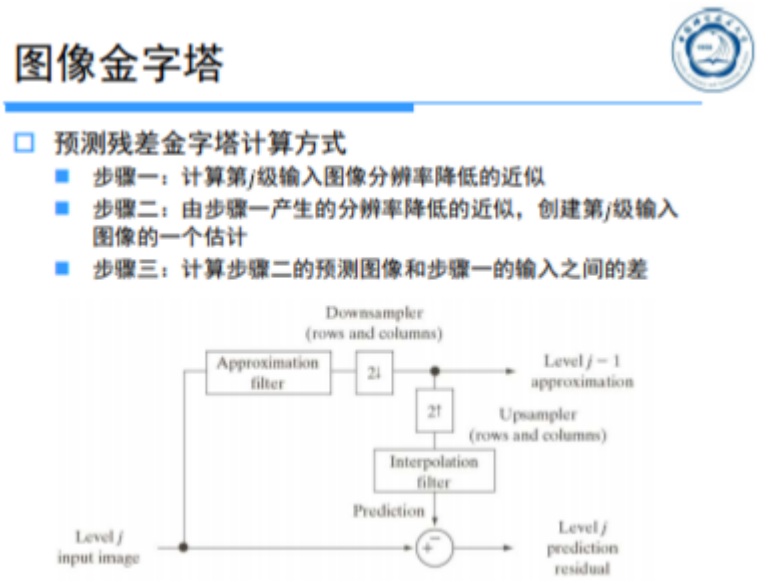
第四次数字图像处理作业

林静雯 PB16060913

实验原理

实验一：近似金字塔和预测残差金字塔

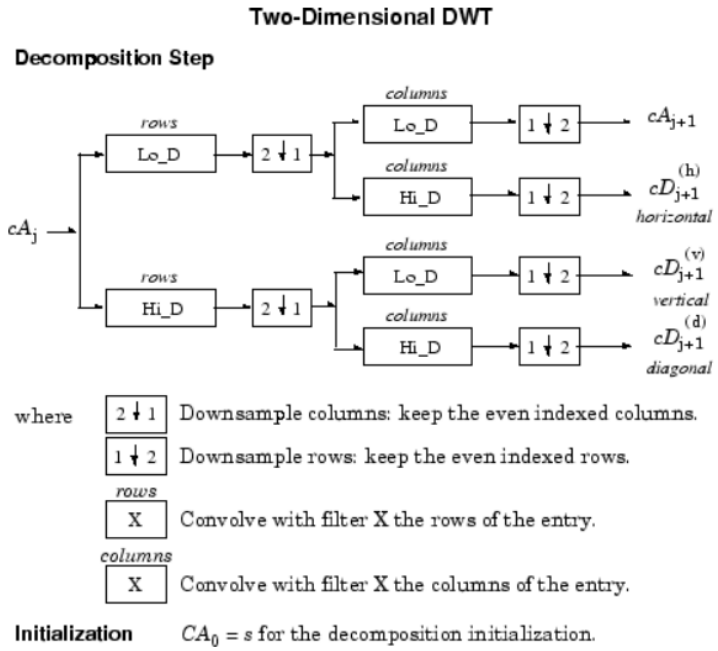




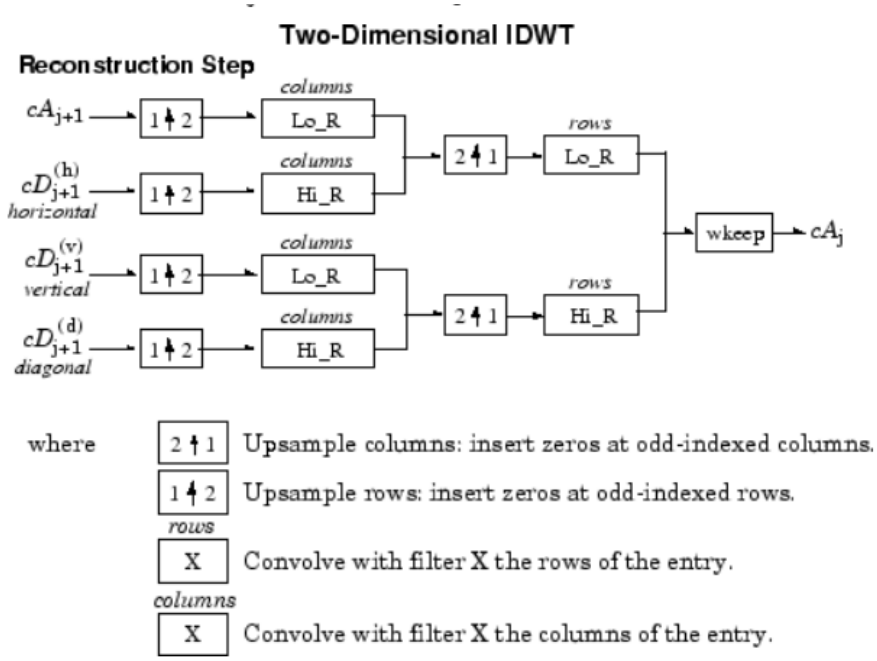
- 由于高频信息在残差图中，所以去噪可以针对残差图进行。

- 由于残差图是系数矩阵，故可以压缩保存

实验二： 快速小波变换以及图像复原



其中Lo_D是h0：尺度函数系数,Hi_D是h1： 小波函数系数



其中Lo_R是g0：尺度函数系数,Hi_R是g1： 小波函数系数

PSNR:峰值信噪比（通常缩写为PSNR）是信号的最大可能功率与影响其表示保真度的破坏噪声功率之间的比率的工程术语。

计算重构图像的PSNR (Peak signal-to-noise ratio) $PSNR$=10×log(255^2/MSE)
MSE=1/mn ∑∑[$I(i,j)$−$K(i,j)$]^2 其中 I 为尺寸为 m×n 的原始图像， K 为重构图像

实验三：利用小波变换进行边缘检测

将离散小波变换的最低尺度的近似分量置零，剩下的是高频信息。而边缘信息就在高频分量中，所以可以增强边缘。如果再将水平分量置零，则可以突出垂直边缘。

实验步骤

实验一：近似金字塔和预测残差金字塔

- 对j级图片进行高斯低通滤波
- 进行下采样 得到第j+1级图片
- 对j+1及图片进行双线性插值得到第j级图片的预测，然后和第j级图片做差得到残差
- 循环

实验二：快速小波变换以及图像复原

FWT:

- 对行卷积，对列进行下采样
- 对列卷积，对行进行下采样
- 得到四个子图，分别是近似子带，垂直子带，水平子带，低频子带
- 将低频子带作为输入，进行下一级分解

IFWT:

- 对近似子带，垂直子带，水平子带，低频子带进行行上采样，对列卷积
- 对列上采样，对行进行卷积
- 得到原图

实验三：利用小波变换进行边缘检测

- 将最低尺度的近似分量置零，然后进行IFWT
- 将最低尺度的近似分量和水平分量置零，然后进行IFWT

代码分析

实验一：近似金字塔和预测残差金字塔

```
im_GLPF = glpf(im_double);              %高斯低通滤波
im_down = downsample(im_GLPF);          %下采样
im_predic = myBilinear(im_down, 2);             %双线性插值
im_residual = residual(im_double, im_predic);     %计算残差
```

```
%%高斯滤波
D0 = 60;                                %滤波半径
D2 = (u-M/2).^2+(v-N/2).^2;
H=exp(-(D2)/(2*D0*D0));
im_F = fftshift(fft2(img));       % 空域 > 频域
im_GLPF = im_F .* H;                %高斯滤波
im_GLPF = real(ifft2(ifftshift(im_GLPF)));     % 频域 > 空域
```

```
%%下采样
down_im = zeros(M/2,N/2);
for i = 1:M/2
    for j = 1:N/2
        down_im(i,j) = im_glpf(2*i, 2*j);           %下采样
    end
end
```

双线性插值不再重复说明。

实验二：快速小波变换以及图像复原

```
%%根据g0计算g1,h1,h0
g0 =[0.0322, -0.0126, -0.0992, 0.2979, 0.8037,0.4976,
-0.0296, -0.0758];
s = (-1).^(0: 7);
g1 = s.*g0(end:-1:1);
h0 = g0(end:-1:1);
h1 = g1(end:-1:1);
```

```
%%分析
%一级分析
[img_low1, img_hori1, img_verti1, img_d1, pic1] =
approxi_filter(im_double);
%二级分析
[img_low2, img_hori2, img_verti2, img_d2, pic2] =
approxi_filter(img_low1);
%三级分析
[img_low3, img_hori3, img_verti3, img_d3, pic3] =
approxi_filter(img_low2);
```

```
%%重建
%一级重建
img_pre1 = synthesis_filter(img_low1, img_hori1,
img_verti1, img_d1);
PSNR1 = 10*log10(1/immse(img_pre1,im_double));
%二级重建
img_low_1 = synthesis_filter(img_low2, img_hori2,
img_verti2, img_d2);
img_pre2 = synthesis_filter(img_low_1, img_hori1,
img_verti1, img_d1);
PSNR2 = 10*log10(1/immse(img_pre2,im_double));
%三级重建
img_low_2 = synthesis_filter(img_low3, img_hori3,
img_verti3, img_d3);
img_low_1 = synthesis_filter(img_low_2, img_hori2,
img_verti2, img_d2);
img_pre3 = synthesis_filter(img_low_1, img_hori1,
img_verti1, img_d1);
PSNR3 = 10*log10(1/immse(img_pre3,im_double));
```

```
%%重建滤波器
function [img_low, img_hori, img_verti, img_d,pic] =
approximation_filer(img)
%对行用h1卷积，对列下采样 得到ROW_H1
%对行用h0卷积，对列下采样 得到ROW_H0
for i = 1 : M
   W1(i,:) = conv(img(i,:),h1,'same');
   W0(i,:) = conv(img(i,:),h0,'same');
   for j = 1 : N/2
        ROW_H1(i, j) = W1(i, 2*j);
        ROW_H0(i, j) = W0(i, 2*j);
   end
end
FN = N/2;
%对行卷积后
%对列向量卷积，对行下采样
for j = 1 : FN
    W_low(:,j) = conv(ROW_H0(:,j),h0,'same');
    W_hori(:,j) = conv(ROW_H0(:,j),h1,'same');
    W_verti(:,j) = conv(ROW_H1(:,j),h0,'same');
    W_d(:,j) = conv(ROW_H1(:,j),h1,'same');
    for i = 1 :M/2
        img_low(i,j) = W_low(2*i, j);
        img_hori(i,j) = W_hori(2*i, j);
        img_verti(i,j) = W_verti(2*i, j);
        img_d(i,j) = W_d(2*i, j);
    end
```

```
end
```

```
%%重建滤波器
function [img_pre] = synthesis_filter(img_low, img_hori,
img_verti, img_d)
%对行上采样 insert zeros at odd-indexed rows
mid_low(2:2:end,:) = img_low(1:end,:);
mid_hori(2:2:end,:) = img_hori(1:end,:);
mid_verti(2:2:end,:) = img_verti(1:end,:);
mid_d(2:2:end,:) = img_d(1:end,:);
%对列进行卷积
for j = 1:N
    mid_d(:,j) = conv(mid_d(:,j),g1,'same');
    mid_verti(:,j) = conv(mid_verti(:,j),g0,'same');
    mid_hori(:,j) = conv(mid_hori(:,j),g1,'same');
    mid_low(:,j) = conv(mid_low(:,j),g0,'same');
end
%对列进行上采样
ori_low_hori = zeros(M*2, N*2);
ori_verti_d = zeros(M*2, N*2);
ori_low_hori(:,2:2:end) = mid_low(:,1:end) +
mid_hori(:,1:end);
ori_verti_d(:,2:2:end) = mid_verti(:,1:end) +
mid_d(:,1:end);
%对行进行卷积
img_pre =  zeros(M*2, N*2);
for i = 1:M*2
    img_pre(i,:) = conv(ori_low_hori(i,:), g0,'same')+
conv(ori_verti_d(i,:), g1,'same');
end
```

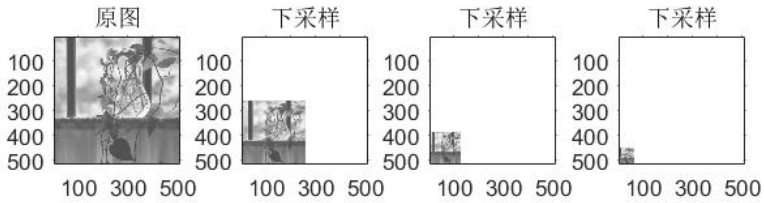
实验三：利用小波变换进行边缘检测

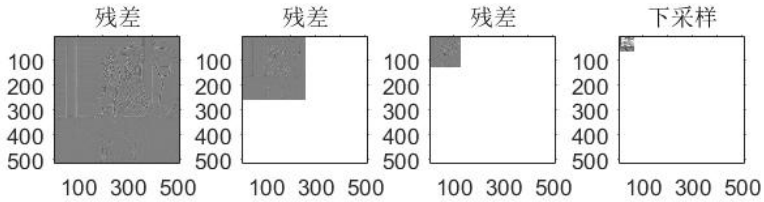
```
% 将最低尺度的近似分量置零，然后进行IFWT

img_low_2 =zeros(M/4,N/4);
img_low_1 = synthesis_filter(img_low_2, img_hori2,
img_verti2, img_d2);
img_edge = synthesis_filter(img_low_1, img_hori1,
img_verti1, img_d1);
```

```
% 将最低尺度的近似分量和水平分量置零，然后进行IFWT
img_low_2 =zeros(M/4,N/4);
img_hori_2 =zeros(M/4,N/4);
img_low_1 = synthesis_filter(img_low_2, img_hori_2,
img_verti2, img_d2);
img_edge = synthesis_filter(img_low_1, img_hori1,
img_verti1, img_d1);
```

实验结果展示

实验一：金字塔.jpg

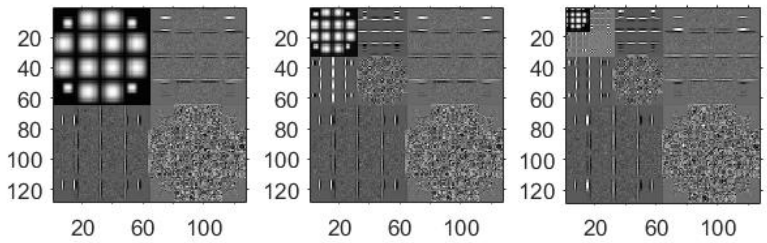




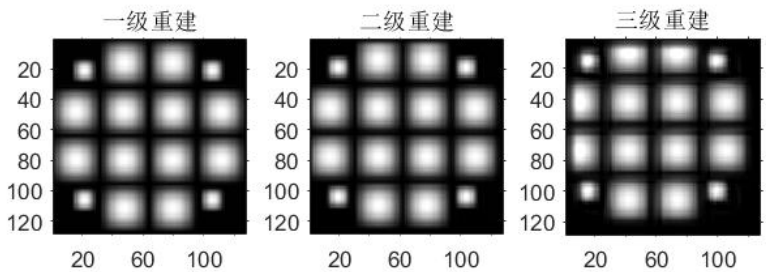
实验二：FWT.jpg, IFWT.jpg

原图：

FWT：



IFWT:

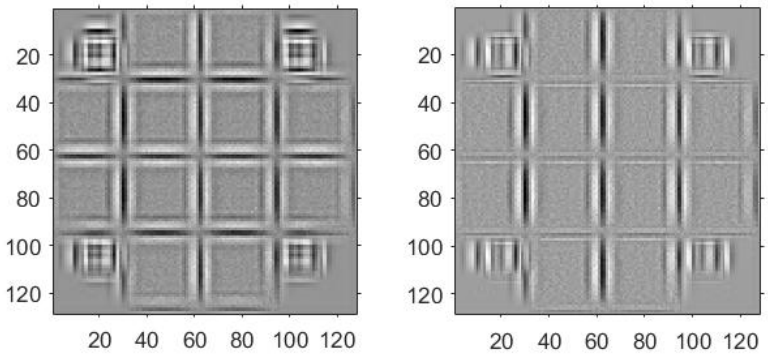


PSNR1: 23.2850

PSNR2 : 14.3430

PSNR3 : 8.7217

实验三： 边缘检测.jpg



实验分析与心得体会

- 实验一：残差图中存在更多的细节，证明高频分量基本都在残差图中，所以去噪可以针对残差图去噪。
- 实验一：残差图在mat2gray之前，基本上都是0，也就是稀疏矩阵，所以可以压缩存储。
- 实验二：经过小波变换可以得到四个子带，并可以较好得恢复出原图。
- 实验二：可以看到第一级重建PSNR是最高的，也就是噪声是最少的，恢复的最好。分解级数越多，噪声越强，PSNR越低。
- 实验三：小波辩护这些子带储存了不同的信息，可以利用这四个子带做边缘检测。可以看到高频信息主要储存在对角子带，水平子带和垂直子带中，并且边缘信息属于高频信息，所以尽管去掉了低频子带，其边缘信息仍很好的保存下来。
- 要注意mat2gray的使用，该函数的作用是将任意矩阵压缩到0-1之间。
- mid_low(2:2:end,:) = img_low(1:end,:);学会了matlab的快速表达。