数字图像处理作业四

夏厚 PB18051031

2021 年 5 月 8 日

1 实验原理

1.1 图像金字塔

以多个分辨率来表示图像的一种结构时图像金字塔，这种结构非常有效，且概念简单。图像金字塔最初用于机器视觉和图像压缩，是一系列以金字塔形状排列的、分辨率逐渐降低的图像集合。金字塔底部是待处理图像的高分辨率表示，顶部是一个低分辨率近似。向金字塔上层移动，尺寸和分辨率逐步降低。基础图像大小为 $N \times N$，P+1 级金字塔中的像素总数是

$$N^2\left(1+\frac{1}{(4)^1}+\frac{1}{(4)^2}+\cdots+\frac{1}{(4)^p}\right) \leqslant \frac{4}{3}N^2$$

近似和预测残差金字塔都以一种迭代方式进行计算。

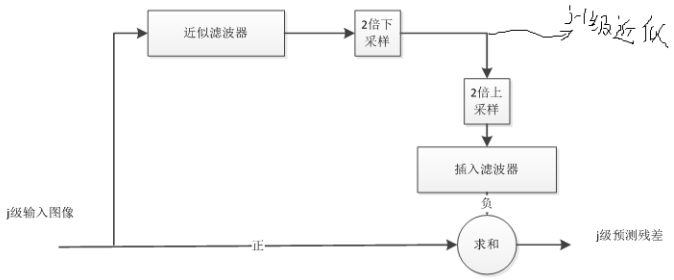

图 1: 创建近似和预测残差金字塔的简单系统

- **步骤 1** 计算第 j 级输入图像分辨率降低的近似。这可以通过滤波并对滤波后的结果以 2 为基进行下采样来完成。得到近似金字塔的第 $j-1$ 级。

- **步骤 2** 由步骤 1 产生的分辨率降低的近似，创建第 j 级输入图像的一个估计。对产生的近似与第 j 级图像进行上采样和滤波。
- **步骤 3** 计算步骤 2 的预测图像和步骤 1 的输入之间的差。结果便是预测残差金字塔的第 j 级。

其中上采样序列 $f_{2\uparrow}(n)$ 如下式定义:

$$f_{2\uparrow}(n) = \begin{cases} f(n/2), & n \quad is \quad even \\ 0, & else \end{cases}$$

下采样的互补操作定义为:

$$f_{2\downarrow}(n) = f(2n)$$

上采样可以视为在序列中的每个样本后插入以 0；下采样可以视为每隔一个样本就丢弃一个样本。

1.2 二维快速小波变换

类似于一维离散小波变换，二维 DWT 可以用数字滤波器和下取样器来实现。利用可分的二维尺度函数和小波函数，可以先简单地取 $f(x,y)$ 的行的一维 FWT，然后，取结果列的一维 FWT。如下图：

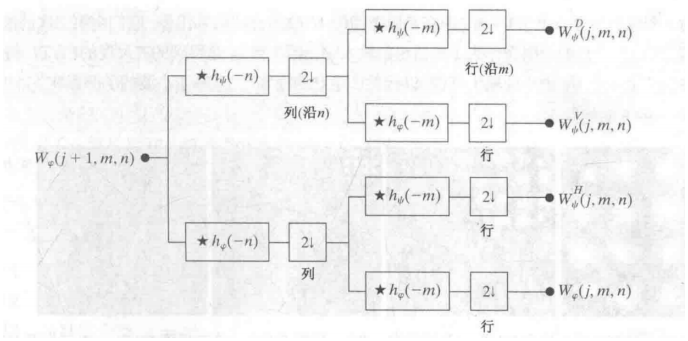

图 2: 分析滤波器组

$f(x,y)$ 被用作 $W_\varphi(J,m,n)$ 的输入。得到四幅 1/4 大小的输出子图像——W_φ,W_ψ^H,W_ψ^V,W_φ^D 如图 3：
其中 W_φ 是对原图的近似；W_ψ^H,W_ψ^V,W_φ^D 分别是原图水平、垂直和对角方向的细节。反向处理的综合滤波器组如图 4：

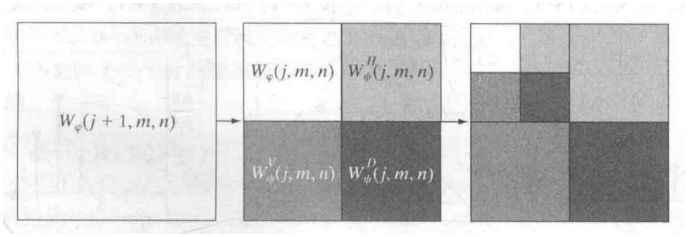

图 3: 分解结果

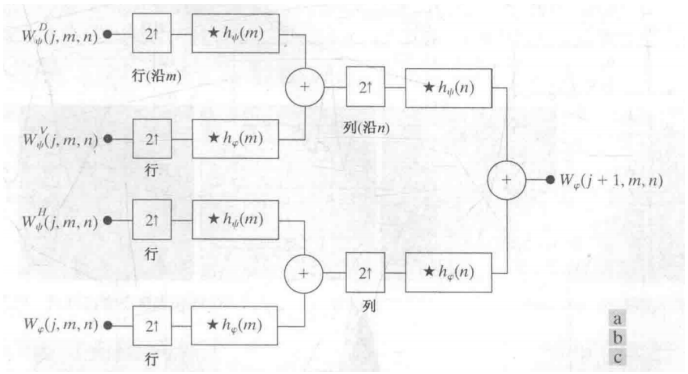

图 4: 综合滤波器组

2 实验内容

2.1 构建图像金字塔，复现图 7.3

产生花瓶图像的 4 级近似金字塔时，使用低通高斯平滑滤波器，并进行下取样操作。由于类似高斯滤波器对图像进行滤波，在前面的实验中有所涉及，此处采用 MATLAB 的内置函数进行高斯滤波和下采样。得到的金字塔包含 512×512 分辨率的原图，和 3 个低分辨率的近似图像。

可以看到，金字塔的分辨率越低，伴随的细节就越少。通常，金字塔的低分辨率级别用于分析较大的结构或图像的整体内容；而高分辨率图像适合于分析单个物体的特性。

使用双线性内插滤波器用于产生预测残差金字塔。此处使用的双线性内插滤波器是直接调用前面实验写过的双线性内插函数。

代码：

```
1          clc;
```

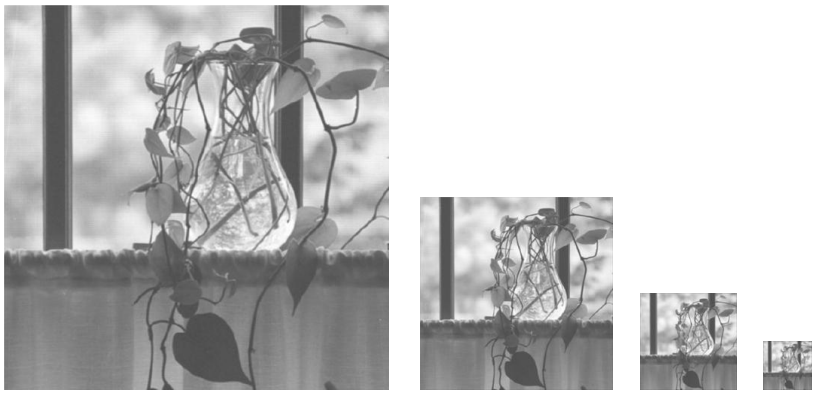

图 5: 近似金字塔

图 6: 预测残差金字塔

```
clear;
close all;
filename = 'demo-1'; %测试图像2
img1 = uint8(imread([filename, '.jpg']));
img1 = im2gray(img1);
[M,N]=size(img1);
w=fspecial('gaussian',[3 3]);
img2=imresize(imfilter(img1,w),[M/2 N/2]);
img3=imresize(imfilter(img2,w),[M/4 N/4]);
img4=imresize(imfilter(img3,w),[M/8 N/8]);
```

```
imshow(img1);
imwrite((img1), sprintf('result/%s.jpg', 'img1'));
figure,imshow(img2);
imwrite((img2), sprintf('result/%s.jpg', 'img2'));
figure,imshow(img3);
imwrite((img3), sprintf('result/%s.jpg', 'img3'));
figure,imshow(img4);
imwrite((img4), sprintf('result/%s.jpg', 'img4'));
% imhist(img2); title('近似图像直方图');
img13=(img3-myBilinear(img4,2)+128);
figure,imshow(uint8(img13));
imwrite((img13), sprintf('result/%s.jpg', 'img13'));
img12=(img2-myBilinear(img3,2)+128);
% imhist(img3-myBilinear(img4,2)+128); title('近似图像直方图');
figure,imshow(img12);
imwrite((img12), sprintf('result/%s.jpg', 'img12'));
img11=img1-myBilinear(img2,2)+128;
figure,imshow(img11);
imwrite((img11), sprintf('result/%s.jpg', 'img11'));
```

2.2 计算二维快速小波变换，并完成基于小波的边缘检测

实验中用到的 $h_\varphi(n)$ 正交归一化四阶对称小波滤波器系数如图 7：

n	$h_\varphi(n)$
0	0.0322
1	−0.0126
2	−0.0992
3	0.2979
4	0.8037
5	0.4976
6	−0.0296
7	−0.0758

图 7: $h_\varphi(n)$ 滤波器系数

原图像作为实验原理中所述分析滤波器的输入，输出 4 个 1/4 大小的分解输出（即近似、水平、垂直和对角线)。类似的过程产生二尺度 FWT，此时滤波器的输入变为一尺度输出的左上角近似图像。同理得到三尺度 FWT 结果。每次通过滤波器组，便会产生 4 个大小为 1/4 的输出图像。

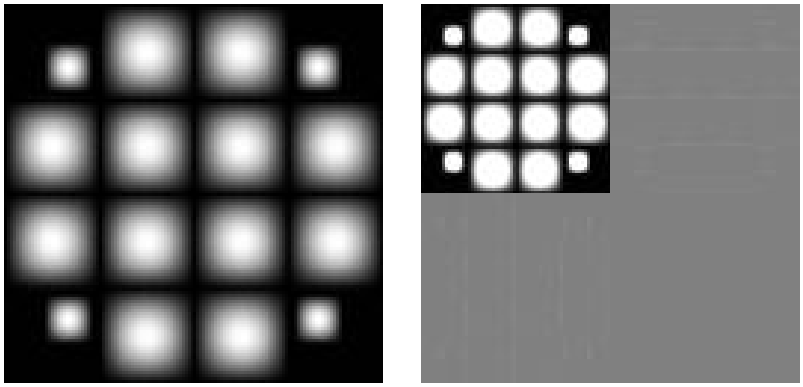

图 8: 教材图 7.25 复现结果 (依次为原图、一尺度 FWT)

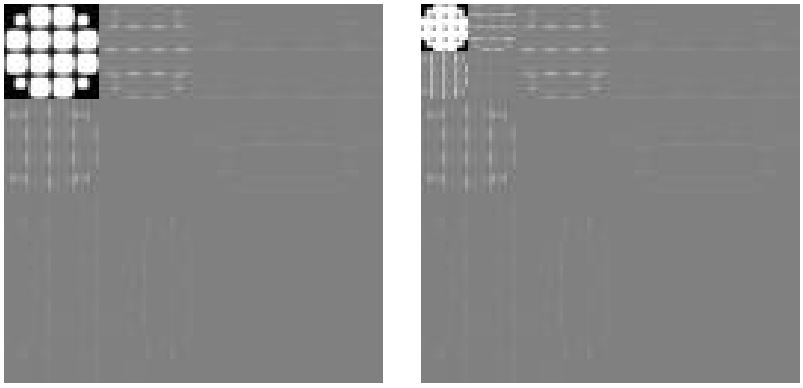

图 9: 教材图 7.25 复现结果 (依次为、二尺度 FWT、三尺度 FWT)

基于小波的边缘检测将前面得到的离散小波变换的近似分量置零，并通过实验原理中所述的综合滤波器组进行反变换，最终效果是边缘增强。进一步将水平细节置零，可以孤立出垂直边缘。

代码分为三个 MATLAB 文件，其中主函数位于 $picture\_7\_25$，分析滤波器组与综合滤波器组独立成函数，分别位于 $tranfor$,$intranfor$ 文件

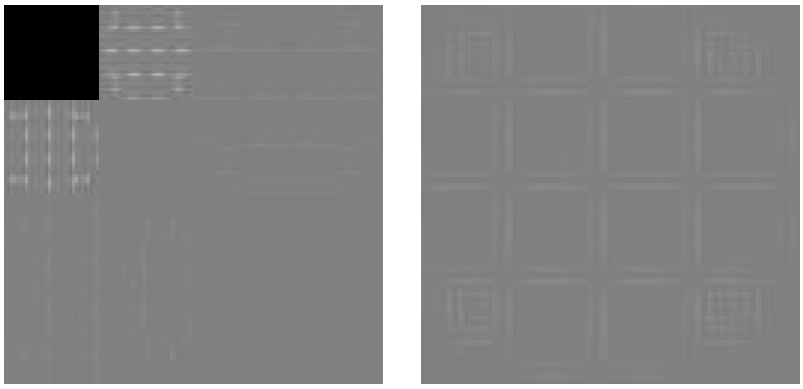

图 10: 教材图 7.27 复现结果 (近似分量置零并反变换)

图 11: 教材图 7.27 复现结果 (进一步将水平细节置零并反变换)

中。下附主函数代码：

```
clc;
clear;
close all;
filename = 'demo-2'; %测试图像2
img1 = uint8(imread([filename, '.tif']));
img1 = im2gray(img1);
[M,N]=size(img1);

%%%%%
```

```
%以下块代码对图7.25进行复现
%%%%%
% imwrite((img1), sprintf('result/%s.jpg', '7_25_img1'));
[img_1 img_2 img_3 img_4]=transfor(img1,M,N);
img2=[img_1 img_2+128;img_3+128 img_4+128];
imshow(img2);
% imwrite((img2), sprintf('result/%s.jpg', '7_25_img2'));
[img_5 img_6 img_7 img_8]=transfor(img_1,M/2,N/2);
img33=[img_5 img_6+128;img_7+128 img_8+128];
img3=[img33 img_2+128;img_3+128 img_4+128];
figure;imshow(img3);
% imwrite((img3), sprintf('result/%s.jpg', '7_25_img3'));
[img_9 img_10 img_11 img_12]=transfor(img_5,M/4,N/4);
img444=[img_9 img_10+128;img_11+128 img_12+128];
img44=[img444 img_6+128;img_7+128 img_8+128];
img4=[img44 img_2+128;img_3+128 img_4+128];
figure;imshow(img4);
% imwrite((img4), sprintf('result/%s.jpg', '7_25_img4'));

%%%%%%
%复现图7.27第1.2图像请取消以下块代码注释
%%%%%%
% img_5=zeros(M/4,N/4);
% img33=[img_5 img_6+128;img_7+128 img_8+128];
% img3=[img33 img_2+128;img_3+128 img_4+128];
% figure;imshow(img3);
% % imwrite((img3), sprintf('result/%s.jpg', '7_27_img1'));
% im7_27_img22=intransfor(img_5,img_6,img_7,img_8,M/4,N/4);
% im7_27_img2=intransfor(im7_27_img22,img_2,img_3,img_4,M/2,N/2);
% figure;imshow(uint8(im7_27_img2)+128);
% % imwrite((uint8(im7_27_img2)+128), sprintf('result/%s.jpg', '7_27

%%%%%%
```

```
%复现图7.27第3.4图像请取消以下块代码注释
%%%%%
% img_5=zeros(M/4,N/4);
% img_6=zeros(M/4,N/4);
% img_2=zeros(M/2,N/2);
% img33=[img_5 img_6;img_7+128 img_8+128];
% img3=[img33 img_2;img_3+128 img_4+128];
% figure;imshow(img3);
% % imwrite((img3), sprintf('result/%s.jpg', '7_27_img3'));
% im7_27_img44=intransfor(img_5,img_6,img_7,img_8,M/4,N/4);
% im7_27_img4=intransfor(im7_27_img44,img_2,img_3,img_4,M/2,N/2);
% figure;imshow(uint8(im7_27_img4)+128);
% % imwrite((uint8(im7_27_img4)+128), sprintf('result/%s.jpg', '7_27
```

3 总结

- 第一部分实验中涉及高斯滤波器，由于空域滤波器在前面的实验中有涉及，本实验中为了提高程序运行效率，使用了 MATLAB 内置的滤波器。
- 第二部分实验分析滤波器与综合滤波器由于都是自己手写的，还有很大的优化空间。例如代码中使用了很多数组保存中间变量，出于时间原因，我没有进行更多的优化。